The Secure Channel

Ramki Thurimella
Secure Channel Properties

- **Roles**
  - Alice—Bob (Server—Client)
  - Direction of flow
  - Eve
    - Delete, Insert, or modify data in transit

- **Can be created for storage**
  - Think of it as transmitting it into the future
  - Both Alice & Bob the same person who knows the key
Key

- Only Alice and Bob know the key $K$
- Every time the secure channel is created, a *session* key is created
- Session key creation is discussed later
- $K$ is 256 bits long
Messages or Streams

- Messages
- Because streams can be broken down into messages, transmitted, and reassembled by the receiver
- Assume unreliable communication
- But lost messages to be dealt by the underlying transport layer
Security Requirements

- Messages are kept confidential from Eve.
- Eve can drop some messages while in transit (That is, Bob receives a subsequence).
- Eve can see the following about messages that are being exchanged by Alice and Bob:
  - How many
  - Size
  - When
- Eve can do *traffic analysis* with this data (For some applications this compromises privacy).
Order of Authentication & Encryption

- Encrypt-then-authenticate
- Authenticate-then-encrypt (recommended by the authors)
- Encrypt-and-authenticate
Encrypt-Then-Authenticate

1. Compute $E(P)$
2. Compute $MAC(E(P))$
3. Send $E(P)$ and $MAC(E(P))$

Receiver
1. Compute $MAC(E(P))$
2. If doesn’t match the MAC that was received
   - Discard $E(P)$
3. Else
   - Decrypt $E(P)$
Encrypt-Then-Authenticate (cont.)

- Advantages
  1. Inverting MAC, only gives $E(P)$ and NOT the message $P$ itself
  2. Efficient: Bob doesn’t have to decrypt, if MACs don’t agree
Encrypt-And-Authenticate

1. Compute $E(P)$ and (possibly simultaneously) $MAC(P)$
2. Send $E(P)$, $MAC(P)$

Receiver
1. Decrypt $E(P)$ and compute $MAC(P)$ (possibly simultaneously)

Advantage:
   Possible parallelism

Disadvantage:
   Eve has access to $MAC(P)$ and could potentially find out $P$
Authenticate-Then-Encrypt (recommended method)

1. Compute MAC(P)
2. Compute E(P,MAC(P))
3. Send E(P,MAC(P))

Receiver

1. Decrypt E(P,MAC(P)), and unpack P and MAC(P)
2. Compute MAC(P)
3. If what was computed doesn’t match the MAC that was received
   Discard the message
Authenticate-Then-Encrypt (cont.)

- Advantages:
  - Eve has no access to MAC(P)
  - Follows Horton Principle (Authenticate what you mean, not what you say) whereas Encrypt-Then-Authenticate violates this principle because it attempts to compute MAC of ciphertext
Secure Channel Design

- **Message Numbers:**
  - Exchanging $N$ messages puts the communication channel in $N+1$ distinct states
    - 0 messages sent
    - 1 message sent
    - 2 messages sent
    - ...
    - $N$ messages sent
  - Using 32-bits, $2^{32}$ states can be recorded, i.e. $N+1 = 2^{32}$
  - Equivalently $N = 2^{32} - 1$
  - $1 \leq \text{Message number } i \leq 2^{32} - 1$
HMAC-SHA-256

- \( H(\cdot) \) be a cryptographic hash function (in this case SHA-256)
- \( K \) be a secret key padded to the right with extra zeros to the input block size of the hash function, or the hash of the original key if it's longer than that block size
- \( m \) be the message to be authenticated
- \(||\) denote concatenation
- \( \oplus \) denote exclusive or (XOR)
- \( a \) be the outer padding (0x5c5c5c...5c5c, one-block-long hexadecimal constant)
- \( b \) be the inner padding (0x363636...3636, one-block-long hexadecimal constant)
- Then \( \text{HMAC}(K,m) \) is mathematically defined by
  - \( \text{HMAC}(K,m) = H((K \oplus a) || H((K \oplus b) || m)). \)
Secure Channel Design (cont.)

- **Authentication**
  - HMAC-SHA-256
  - \[ a_i = \text{MAC} ( i \ || \ l(x_i) \ || \ x_i \ || \ m_i ) \]
  - \( l(.) \) length of a string in bytes
  - \( x \) is contextual data
    - Protocol identification
    - Protocol version
    - Negotiated field sizes
  - \( m_i \) message \( i \)
Secure Channel Design (cont.)

- Encryption
  - AES in CTR mode. Recall that this is
    - $K_i = E(K, \text{Nonce} \ || \ i)$ for $i = 1, 2, \ldots, k$
    - $C_i = P_i \oplus K_i$
  - Length of each message $\leq 16 \cdot 2^{32}$
  - Each message is broken down into blocks, and each block is numbered
  - Each block is 128 bits long
  - With 32-bit counter for block numbers, messages up to $2^{36}$ can be accommodated