Secure Channel

Cryptography and Protocols

Andrei Bulatov

Secure Channel: Security Requirements

- Alice sends a sequence of messages $P_1, P_2, \ldots$
- Bob receives (after removing those failed authentication) a sequence of messages $P_1', P_2', \ldots$
- **R1**. Eve does not learn anything about messages P_i except for their timing and size
- **R2**. The sequence $P_1', P_2', \ldots$ is a subsequence of $P_1, P_2, \ldots$ and Bob knows for sure which subsequence he receives.

Secure Channel: Key, Message Number, …

- **Key**. Alice and Bob have a shared key. How they get it will be discussed later. Choose the key length to be 256
- **Message number**. Bob needs to recover the order of messages, so we need a message number. Besides it increases strength of the scheme. Choose 4-byte message number.
- **Authentication**. Use HMAC-h, where h is, say, SHA-256.

  For each i let P_i be the corresponding message and X_i certain auxiliary data about the message. Then the tag we compute is

  $$T_i = MAC(i \parallel \ell(X_i) \parallel X_i \parallel P_i)$$

- **Encryption**. Use AES in the counter mode. Include the block counter into the message

$$k_i = E_k(0 \parallel i \parallel 0)||E_k(1 \parallel i \parallel 1) \ldots \quad C_i = i \parallel (k_i \oplus (P_i \parallel T_i))$$

Secure Channel: Initialization

- **Input**: k key of the channel, 256 bits
  Output: S state for the channel
- **Algorithm**

  KeySendEnc := h(k||`Enc Alice to Bob')
  KeyRecEnc := h(k||`Enc Bob to Alice')
  KeySendAuth := h(k||`Auth Alice to Bob')
  KeyRecAuth := h(k||`Auth Bob to Alice')
  MsgCntSend, MsgCntRec := 0
  S := (KeySendEnc , KeyRecEnc , KeySendAuth , KeyRecAuth ,
  MsgCntSend, MsgCntRec)
  return S

For security we use 4 different keys to encrypt and authenticate by Alice and Bob

Secure Channel: Sending a Message

- **Input**: S channel state, P message, X extra data
  Output: C data to transmit
- **Algorithm**

MsgCntSend := MsgCntSend + 1 i := MsgCntSend

/* Authentication

T := HMAC-h(KeySendAuth||i||l(X)||X||P)

C := P||T

/* Encryption

K := KeySendEnc

$k := E_K(0 \parallel i \parallel 0) \parallel E_K(1 \parallel i \parallel 1) \parallel \cdots$

C := i||(C $\oplus$ k)

return C

Secure Channel: Receiving a Message

- **Input**: S channel state, C ciphertext, X extra data
  Output: P original message
- **Algorithm**

  i || C := C

  K := KeySendEnc $k := E_K(0 \parallel i \parallel 0) \parallel E_K(1 \parallel i \parallel 1) \ldots$

  P || T := C ⊕ k /* Decrypt

  T' := HMAC-h(KeySendAuth||i||l(X)||X||P) /* Check authent.

  if T ≠ T' **then** **destroy** k, P **return** AuthenticationFailure

  else if i ≤ MsgCntRec **then**

  destroy k, P **return** MessageOrderError

  MsgCntRec := i

  return P

SSL, TSL

- SSL – Secure Socket Layer, TSL – Transport Secure Layer
- Integrated into IPv6, can be used with IPv4
- Supports many ciphers and MACs
- Structure:

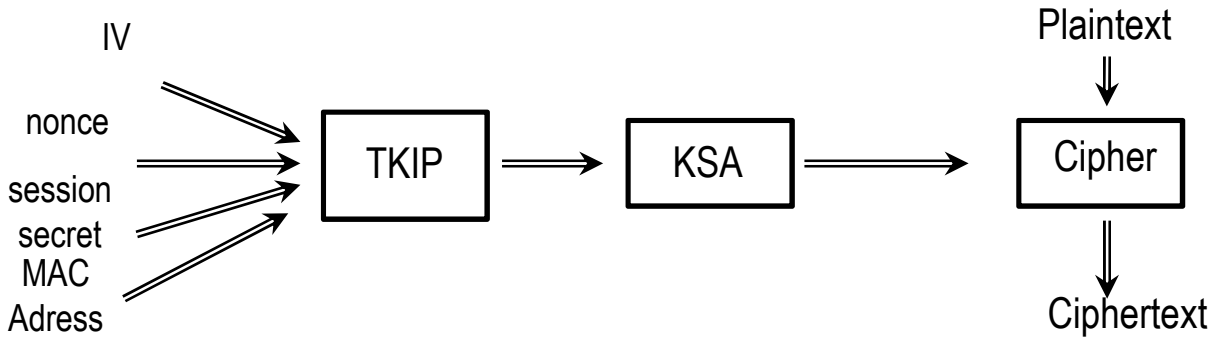
  Handshaking (to be discussed later) Produces the type of cipher and MAC used, key for encryption and authentication, verifies the server, etc.

  Encryption /Authentication

and then encrypt

WPA2

- General scheme



- More details