Chapter 6 – Contemporary Symmetric Ciphers

"I am fairly familiar with all the forms of secret writings, and am myself the author of a trifling monograph upon the subject, in which I analyze one hundred and sixty separate ciphers," said Holmes.

—The Adventure of the Dancing Men, Sir Arthur Conan Doyle
Multiple Encryption & DES

• clearly a replacement for DES was needed
  – theoretical attacks that can break it
  – demonstrated exhaustive key search attacks

• AES is a new cipher alternative

• prior to this alternative was to use multiple encryption with DES implementations

• Triple-DES is the chosen form
Double-DES?

- could use 2 DES encrypts on each block
  \[ C = E_{K2}(E_{K1}(P)) \]
- issue of reduction to single stage
- and have “meet-in-the-middle” attack
  - works whenever use a cipher twice
  - since \( X = E_{K1}(P) = D_{K2}(C) \)
  - attack by encrypting \( P \) with all keys and store
  - then decrypt \( C \) with keys and match \( X \) value
  - can show takes \( O(2^{56}) \) steps
Triple-DES with Two-Keys

• hence must use 3 encryptions
  – would seem to need 3 distinct keys
• but can use 2 keys with E-D-E sequence
  – $C = E_{K_1} (D_{K_2} (E_{K_1} (P)))$
  – encrypt & decrypt equivalent in security
  – if $K_1 = K_2$ then can work with single DES
• standardized in ANSI X9.17 & ISO8732
• no current known practical attacks
Triple-DES with Three-Keys

• although are no practical attacks on two-key Triple-DES have some indications

• can use Triple-DES with Three-Keys to avoid even these

  \[ C = E_{K3} \left( D_{K2} \left( E_{K1} \left( P \right) \right) \right) \]

• has been adopted by some Internet applications, eg PGP, S/MIME
Modes of Operation

• **block ciphers encrypt fixed size blocks**
  – eg. DES encrypts 64-bit blocks with 56-bit key
• need some way to en/decrypt arbitrary amounts of data in practice
• **ANSI X3.106-1983 Modes of Use** (now FIPS 81) defines 4 possible modes
• subsequently 5 defined for AES & DES
• have **block** and **stream** modes
Electronic Codebook Book (ECB)

• message is broken into independent blocks which are encrypted
• each block is a value which is substituted, like a codebook, hence name
• each block is encoded independently of the other blocks
  \[ C_i = DES_{K_1}(P_i) \]
• uses: secure transmission of single values
Electronic CodeBook (ECB)

(a) Encryption

(b) Decryption
Advantages and Limitations of ECB

• message repetitions may show in ciphertext
  – if aligned with message block
  – particularly with data such as graphics
  – or with messages that change very little, which become a code-book analysis problem

• weakness is due to the encrypted message blocks being independent

• main use is sending a few blocks of data
Cipher Block Chaining (CBC)

- message is broken into blocks
- linked together in encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process
  \[ C_i = DES_{K_1}(P_i \ XOR \ C_{i-1}) \]
  \[ C_{-1} = IV \]
- uses: bulk data encryption, authentication
Cipher Block Chaining (CBC)
Message Padding

• at end of message must handle a possible last short block
  – which is not as large as blocksize of cipher
  – pad either with known non-data value (eg nulls)
  – or pad last block along with count of pad size
    • eg. [ b1 b2 b3 0 0 0 0 5]
      • means have 3 data bytes, then 5 bytes pad+count
  – this may require an extra entire block over those in message

• there are other, more esoteric modes, which avoid the need for an extra block
Advantages and Limitations of CBC

- a ciphertext block depends on all blocks before it
- any change to a block affects all following ciphertext blocks
- need **Initialization Vector (IV)**
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value (as in EFTPOS)
  - or must be sent encrypted in ECB mode before rest of message
Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feedback for next stage (hence name)
- standard allows any number of bits (1, 8, 64 or 128 etc) to be feedback
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- most efficient to use all bits in block (64 or 128)
  \[ C_i = P_i \ XOR \ DES_{K_1}(C_{i-1}) \]
  \[ C_{-1} = IV \]
- uses: stream data encryption, authentication
Cipher FeedBack (CFB)

(a) Encryption

(b) Decryption
Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in encryption mode at both ends
- errors propagate for several blocks after the error
Output FeedBack (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance
  \[ C_i = P_i \ XOR \ O_i \]
  \[ O_i = DES_{K_1}(O_{i-1}) \]
  \[ O_{-1} = IV \]
- uses: stream encryption on noisy channels
Output FeedBack (OFB)
Advantages and Limitations of OFB

- bit errors do not propagate
- more vulnerable to message stream modification
- a variation of a Vernam cipher
  - hence must never reuse the same sequence (key+IV)
- sender & receiver must remain in sync
- originally specified with m-bit feedback
- subsequent research has shown that only full block feedback (ie CFB-64 or CFB-128) should ever be used
Counter (CTR)

- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

\[
C_i = P_i \oplus O_i \\
O_i = \text{DES}_{K1}(i)
\]

- uses: high-speed network encryptions
Counter (CTR)
Advantages and Limitations of CTR

• efficiency
  – can do parallel encryptions in hardware or software
  – can preprocess in advance of need
  – good for bursty high speed links
• random access to encrypted data blocks
• provable security (good as other modes)
• but must ensure never reuse key/counter values, otherwise could break (cf OFB)
Stream Ciphers

• process message bit by bit (as a stream)
• have a pseudo random keystream
• combined (XOR) with plaintext bit by bit
• randomness of stream key completely destroys statistically properties in message
  \[ C_i = M_i \text{ XOR StreamKey}_i \]
• but must never reuse stream key
  – otherwise can recover messages (see book cipher)
Stream Cipher Structure

\[ K \]

Pseudorandom byte generator
(key stream generator)

\[ k \]

Plaintext
byte stream
M

\[ + \]

Ciphertext
byte stream
C

\[ + \]

Plaintext
byte stream
M

Key
K

Encryption

Decryption
Stream Cipher Properties

• some design considerations are:
  – long period with no repetitions
  – statistically random
  – depends on large enough key
  – large linear complexity

• properly designed, can be as secure as a block cipher with same size key

• but usually simpler & faster
RC4

• a proprietary cipher owned by RSA (a private company)
• another Ron Rivest design, simple but effective
• variable key size, byte-oriented stream cipher
• widely used (web SSL/TLS, wireless WEP)
• key forms random permutation of all 8-bit values
• uses that permutation to scramble input info processed a byte at a time
RC4 Key Schedule

- starts with an array S of numbers: 0..255
- use key to well and truly shuffle
- S forms **internal state** of the cipher

for i = 0 to 255 do
  S[i] = i
  T[i] = K[i mod keylen])
j = 0
for i = 0 to 255 do
  j = (j + S[i] + T[i]) (mod 256)
  swap (S[i], S[j])
RC4 Encryption

- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value from permutation
- XOR S[t] with next byte of message to en/decrypt

\[
i = j = 0
\]
\[
\text{for each message byte } M_i
\]
\[
i = (i + 1) \pmod{256}
\]
\[
j = (j + S[i]) \pmod{256}
\]
\[
\text{swap}(S[i], S[j])
\]
\[
t = (S[i] + S[j]) \pmod{256}
\]
\[
C_i = M_i \text{ XOR } S[t]
\]
RC4 Overview

(a) Initial state of S and T

(b) Initial permutation of S

(c) Stream Generation
RC4 Security

• claimed secure against known attacks
  – have some analyses, none practical
• result is very non-linear
• since RC4 is a stream cipher, must never reuse a key
• have a concern with WEP, but due to key handling rather than RC4 itself
XTS-AES Mode for Block-Oriented Storage Devices

- P1619 was designed for the following characteristics

1. The ciphertext is freely available for an attacker. Among the circumstances that lead to this situation:
   a. A group of users has authorized access to a database. Some of the records in the database are encrypted so that only specific users can successfully read/write them. Other users can retrieve an encrypted record but are unable to read it without the key.
   b. An unauthorized user manages to gain access to encrypted records.
   c. A data disk or laptop is stolen, giving the adversary access to the encrypted data.

2. The data layout is not changed on the storage medium and in transit. The encrypted data must be the same size as the plaintext data.

3. Data are accessed in fixed sized blocks, independently from each other. That is, an authorized user may access one or more blocks in any order.

4. Encryption is performed in 16-byte blocks, independently from other blocks (except the last two plaintext blocks of a sector, if its size is not a multiple of 16 bytes).

5. There are no other metadata used, except the location of the data blocks within the whole data set.

6. The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again.

7. A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device.
Operations on a Single Block

- There are two keys and two AES algorithms

i is the sector number; it acts like the IV in CBC

j is the sequential number of the block within the sector; it acts like the counter in CTR mode

See next slide
Terminology

*Key* The 256 or 512 bit XTS-AES key; this is parsed as a concatenation of two fields of equal size called *Key*₁ and *Key*₂, such that *Key* = *Key*₁ || *Key*₂.

*P*ₖ The *j*th block of plaintext. All blocks except possibly the final block have a length of 128 bits. A plaintext data unit, typically a disk sector, consists of a sequence of plaintext blocks *P*₁, *P*₂, …, *P*ₘ.

*C*ₖ The *j*th block of ciphertext. All blocks except possibly the final block have a length of 128 bits.

*j* The sequential number of the 128-bit block inside the data unit.

*i* The value of the 128-bit tweak. Each data unit (sector) is assigned a tweak value that is a nonnegative integer. The tweak values are assigned consecutively, starting from an arbitrary nonnegative integer.

*α* A primitive element of GF(2¹²⁸) that corresponds to polynomial *x* (i.e., 0000…010₂).

*α*ʲ A multiplied by itself *j* times, in GF(2¹²⁸).

⊕ Bitwise XOR.

⊗ Modular multiplication of two polynomials with binary coefficients modulo *x*¹²⁸ + *x*⁷ + *x*² + *x* + 1. Thus, this is multiplication in GF(2¹²⁸).
Encryption and Decryption

<table>
<thead>
<tr>
<th>XTS-AES block operation</th>
<th>$T = E(K_2, i) \otimes \alpha^j$</th>
<th>$T = E(K_2, i) \otimes \alpha^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PP = P \oplus T$</td>
<td>$CC = C \oplus T$</td>
<td></td>
</tr>
<tr>
<td>$CC = E(K_1, PP)$</td>
<td>$PP = D(K_1, CC)$</td>
<td></td>
</tr>
<tr>
<td>$C = CC \oplus T$</td>
<td>$P = PP \oplus T$</td>
<td></td>
</tr>
</tbody>
</table>

To see that decryption recovers the plaintext, let us expand the last line of both encryption and decryption. For encryption, we have

$$C = CC \oplus T = E(K_1, PP) \oplus T = E(K_1, P \oplus T) \oplus T$$

and for decryption, we have

$$P = PP \oplus T = D(K_1, CC) \oplus T = D(K_1, C \oplus T) \oplus T$$

Now, we substitute for $C$:

$$P = D(K_1, C \oplus T) \oplus T$$

$$= D(K_1, [E(K_1, P \oplus T) \oplus T] \oplus T) \oplus T$$

$$= D(K_1, E(K_1, P \oplus T)) \oplus T$$

$$= (P \oplus T) \oplus T = P$$
Encryption of a Sector

- The input to the XTS-AES algorithm is $m$ 128 bit blocks, $P_0$, $P_1$, ..., $P_m$, where the last block which may be only partially filled (see last slide)
Decryption of a Sector

- Notice that each block, except possibly the last block, is treated independently and may be processed in parallel

(b) Decryption
## Handling the Final Block

- The last two blocks are encrypted and decrypted using a cipher stealing technique

| XTS-AES mode with null final block | $C_j = \text{XTS-AES-blockEnc}(K, P_j, i, j)$  $j = 0, \ldots, m - 1$
| | $P_j = \text{XTS-AES-blockEnc}(K, C_j, i, j)$  $j = 0, \ldots, m - 1$
| XTS-AES mode with final block containing $s$ bits | $C_j = \text{XTS-AES-blockEnc}(K, P_j, i, j)$  $j = 0, \ldots, m - 2$
| | $XX = \text{XTS-AES-blockEnc}(K, P_{m-1}, i, m - 1)$
| | $CP = \text{LSB}_{128-s}(XX)$
| | $YY = P_m \parallel CP$
| | $C_{m-1} = \text{XTS-AES-blockEnc}(K, YY, i, m)$
| | $C_m = \text{MSB}_s(XX)$
| | $P_j = \text{XTS-AES-blockEnc}(K, C_j, i, j)$  $j = 0, \ldots, m - 2$
| | $YY = \text{XTS-AES-blockDec}(K, C_{m-1}, i, m - 1)$
| | $CP = \text{LSB}_{128-s}(YY)$
| | $XX = C_m \parallel CP$
| | $P_{m-1} = \text{XTS-AES-blockDec}(K, XX, i, m)$
| | $P_m = \text{MSB}_s(YY)$