Processing with Block Ciphers

- Most ciphers work on blocks of fixed (small) size
- How to encrypt long messages?
- Modes of operation
  - ECB (Electronic Code Book)
  - CBC (Cipher Block Chaining)
  - OFB (Output Feedback)
  - CFB (Cipher Feedback)
  - CTR (Counter)

Issues for Block Chaining Modes

- Information leakage
  - Does it reveal info about the plaintext blocks?
- Ciphertext manipulation
  - Can an attacker modify ciphertext block(s) in a way that will produce a predictable/desired change in the decrypted plaintext block(s)?
  - Note: assume the structure of the plaintext is known, e.g., first block is employee #1 salary, second block is employee #2 salary, etc.
Issues… (Cont’d)

• **Parallel/Sequential**
  – Can blocks of plaintext (ciphertext) be encrypted (decrypted) in parallel?

• **Error propagation**
  – If there is an error in a plaintext (ciphertext) block, will there be an encryption (decryption) error in more than one ciphertext (plaintext) block?

Electronic Code Book (ECB)

- The easiest mode of operation; each block is independently encrypted

ECB Decryption

- Each block is independently decrypted
ECB Properties

- Does information leak?
- Can ciphertext be manipulated profitably?
- Parallel processing possible?
- Do ciphertext errors propagate?

Cipher Block Chaining (CBC)

- Chaining dependency: each ciphertext block depends on all preceding plaintext blocks

Initialization Vectors

- **Initialization Vector (IV)**
  - Used along with the key; not secret
  - For a given plaintext, changing either the key, or the IV, will produce a different ciphertext
  - Why is that useful?
- IV generation and sharing
  - Random; may transmit with the ciphertext
  - Incremental; predictable by receivers
CBC Decryption

• How many ciphertext blocks does each plaintext block depend on?

CBC Properties

• Does information leak?
  – Identical plaintext blocks will produce different ciphertext blocks

• Can ciphertext be manipulated profitably?
  – ???

• Parallel processing possible?
  – no (encryption), yes (decryption)

• Do ciphertext errors propagate?
  – yes (encryption), a little (decryption)

Output Feedback Mode (OFB)
**OFB Decryption**

**OFB Properties**

- Does information leak?
  - identical plaintext blocks produce different ciphertext blocks
- Can ciphertext be manipulated profitably?
  - ???
- Parallel processing possible?
  - no (generating pad), yes (XORing with blocks)
- Do ciphertext errors propagate?
  - ???

**OFB ... (Cont’d)**

- If you know one plaintext/ciphertext pair, can easily derive the one-time pad that was used
  - i.e., should not reuse a one-time pad!
- Conclusion: IV must be different every time
Cipher Feedback Mode (CFB)

- Ciphertext block \( C_j \) depends on all preceding plaintext blocks.

CFB Decryption

- No block decryption required!

CFB Properties

- Does information leak?
  - Identical plaintext blocks produce different ciphertext blocks
- Can ciphertext be manipulated profitably?
  - ???
- Parallel processing possible?
  - no (encryption), yes (decryption)
- Do ciphertext errors propagate?
  - ???
CTR Mode Properties

- Does information leak?
  - Identical plaintext block produce different ciphertext blocks
- Can ciphertext be manipulated profitably
  - ???
- Parallel processing possible
  - Yes (both generating pad and XORing)
- Do ciphertext errors propagate?
  - ???
- Allow decryption the ciphertext at any location
  - Ideal for random access to ciphertext
Stronger DES

- Major limitation of DES
  - Key length is too short
- Can we apply DES multiple times to increase the strength of encryption?

Double Encryption with DES

- Encrypt the plaintext twice, using two different DES keys
- Total key material increases to 112 bits
  - Is that the same as key strength of 112 bits?

Concerns About Double DES

- Wasn’t clear at the time if DES was a group (it’s not)
  - If it were, then $E_{k2}(E_{k1}(P)) = E_{k1}(P)$, for all $P$
    - Not good?
- Possible attack (better than brute force): meet-in-the-middle
  - A known-plaintext attack
The Meet-in-the-Middle Attack

1. Choose a plaintext $P$ and generate ciphertext $C$, using double-DES with $K_1 + K_2$
2. Then...
   a. encrypt $P$ using single-DES for all possible $2^{56}$ values $K_1$ to generate all possible single-DES ciphertexts for $P$: $X_1, X_2, \ldots, X_{2^{56}}$; store these in a table indexed by ciphertext values
   b. decrypt $C$ using single-DES for all possible $2^{56}$ values $K_2$ to generate all possible single-DES plaintexts for $C$: $Y_1, Y_2, \ldots, Y_{2^{56}}$; for each value, check the table

Steps … (Cont’d)

3. Meet-in-the-middle:
   - each match ($X_i = Y_j$) reveals a candidate keypair $K_i + K_j$
   - there should be approx. $(2^{112} / 2^{64}) = 2^{48}$ such pairs for one value of $(P, C)$
     - $2^{112}$ possible keys, but there are only $2^{64}$ X’s
   - Repeat the above, for a second plaintext/ciphertext pair $(P', C')$, and find those $2^{48}$ candidate keypairs $K_i' + K_j'$

Why $2^{48}$?
- The table contains only $2^{48}/2^{64}$ of all possible 64-bit values
- for each $X_i$, there is only 1/2 chance there is a matching $Y_j$

Steps … (Cont’d)

4. Look for an identical candidate keypair that produces collisions for both $(P, C)$ and $(P', C')$
   - the probability the same candidate keypair occurs for both plaintexts, but is not the keypair used in the double-DES encryption: $2^{48} / 2^{64} = 2^{-16}$
   - An expensive attack (computation + storage)
     - still, enough of a threat to discourage use of double-DES

Why $2^{-16}$?
- there are about $2^{48}$ candidate keypairs $K_i + K_j$
  - at most one is $K_1 + K_2$, the rest are imposters
  - if $K_i + K_j$ is an imposter, the probability using $K_i + K_j$ that $E(P) = D(C)$ is $1/2^{64}$
**Triple Encryption (Triple DES-EDE)**

- Why not E-E-E?
  - again, wasn’t clear if DES was a group
- Apply DES encryption/decryption three times
  - why not 3 different keys?
  - why not the same key 3 times?

**Triple DES (Cont’d)**

- Widely used
  - equivalent strength to using a 112 bit key
  - strength about $2^{110}$ against M-I-T-M attack
- However: inefficient / expensive to compute
  - one third as fast as DES on the same platform, and DES is already designed to be slow in software
- Next question: how is block chaining used with triple-DES?

**3DES-EDE: Outside Chaining Mode**

- What basic chaining mode is this?
OCM Properties

- Does information leak?
  - identical plaintext blocks produce different ciphertext blocks
- Can ciphertext be manipulated profitably?
  - ???
- Parallel processing possible?
  - no (encryption), yes (decryption)
- Do ciphertext errors propagate?
  - ???

3DES-EDE: Inside Chaining Mode
Message Authentication

- Encryption easily provides confidentiality of messages
  – only the party sharing the key (the “key partner”) can decrypt the ciphertext
- How to use encryption to authenticate messages? That is,
  – prove the message was created by the key partner
  – prove the message wasn’t modified by someone other than the key partner

Approach #1

- The quick and dirty approach
- If the decrypted plaintext “looks plausible”, then conclude ciphertext was produced by the key partner
  – i.e., illegally modified ciphertext, or ciphertext encrypted with the wrong key, will probably decrypt to random-looking data
- But, is it easy to verify data is “plausible-looking”? What if all data is plausible?
Approach #2: Plaintext+Ciphertext

- Send plaintext and ciphertext
  - receiver encrypts plaintext, and compares result with received ciphertext
  - forgeries / modifications easily detected
  - any problems / drawbacks?

Approach #3: Use Residue

- Encrypt plaintext using DES CBC mode, with IV set to zero
  - the last (final) ciphertext output block is called the residue

Approach #3… (Cont’d)

- Transmit the plaintext and this residue
  - receiver computes same residue, compares to the received residue
  - forgeries / modifications highly likely to be detected
Message Authentication Codes

- **MAC**: a small fixed-size block (i.e., independent of message size) generated from a message using secret key cryptography
  - also known as *cryptographic checksum*

Requirements for MAC

1. Given M and MAC(M), it should be **computationally infeasible (expensive)** to construct (or find) another message M' such that MAC(M') = MAC(M)
2. MAC(M) should be uniformly distributed in terms of M
   - for randomly chosen messages M and M',
     \[ P(\text{MAC}(M) = \text{MAC}(M')) = 2^{-k}, \]
     where k is the number of bits in the MAC

Requirements ... (cont’d)

3. Knowing MAC(M1), MAC(M2), ... of some (known or chosen) messages M1, M2, ..., it should be **computationally infeasible** for an attacker to find the MAC of some other message M’
S.K. Crypto for Confidentiality AND Authenticity?

• So far we’ve got
  – confidentiality (encryption),
    or…
  – authenticity (MACs)

• Can we get both at the same time with one cryptographic operation?

Attempt #1

1. Sender computes an error-correcting code or Frame-Check Sequence (FCS) \( F(P) \) of the plaintext \( P \)
2. Sender concatenates \( P \) and \( F(P) \) and encrypts
   • i.e., \( C = E_K( P | F(P) ) \)
3. Receiver decrypts received ciphertext \( C' \) using \( K \), to get \( P'|F' \)
4. Receiver computes \( F(P') \) and compares to \( F' \) to authenticate received message \( P' = P \)
   • How does this authenticate \( P \)?

Attempt #1… (Cont’d)

• The order (1) FCS, then (2) encryption is critical
  – why not (2), then (1)?
• “Subtle weaknesses” known in this approach, so not preferred
Attempt #2

1. Compute **residue** (MAC) using key **K1**
2. Encrypt plaintext **message** **M** using key **K2** to produce **C**
3. Transmit **MAC | C** to receiver
4. Receiver decrypts received **C’** with **K2** to get **P’**
5. Receiver computes **MAC(P’)** using **K1**, compares to received **MAC’**

Attempt #2… (cont’d)

- Good (cryptographic) quality, but…
- Expensive! Two separate, full encryptions with different keys are required

Summary

1. ECB mode is not secure
   - CBC most commonly used mode of operation
2. Triple-DES (with 2 keys) is much stronger than DES
   - usually uses EDE in Outer Chaining Mode
3. MACs use crypto to authenticate messages at a small cost of additional storage / bandwidth
   - but at a high computational cost