Block Ciphers

Block Ciphers

- Modern version of a **codebook cipher**
- In effect, a block cipher algorithm yields a huge number of codebooks
  - Specific codebook determined by key
- It is OK to use same key for a while
  - Just like classic codebook
  - **Initialization vector** (IV) is like additive
- Change the key, get a new codebook

(Iterated) Block Cipher

- Plaintext and ciphertext "units" are fixed sized blocks
  - Typical block sizes: 64 to 256 bits
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of key and the output of previous round
- Most are designed for software

Multiple Blocks

- How to encrypt multiple blocks?
- A new key for each block?
  - As bad as (or worse than) a one-time pad!
- Encrypt each block independently?
- Make encryption depend on previous block(s), i.e., "chain" the blocks together?
- How to handle partial blocks?

Block Cipher Modes

- We discuss 3 (many others)
- Electronic Codebook (**ECB**) mode
  - Encrypt each block independently
  - There is a serious weakness
- Cipher Block Chaining (**CBC**) mode
  - Chain the blocks together
  - Better than ECB, virtually no extra work
- Counter Mode (**CTR**) mode
  - Like a stream cipher (random access)

ECB Mode

- Notation: $C=E(P,K)$
- Given plaintext $P_0,P_1,\ldots,P_m,\ldots$
- Obvious way to use a block cipher is

Encrypt

$C_0 = E(P_0, K),$
$C_1 = E(P_1, K),$
$C_2 = E(P_2, K),\ldots$

Decrypt

$P_0 = D(C_0, K),$
$P_1 = D(C_1, K),$
$P_2 = D(C_2, K),\ldots$

- For a fixed key K, this is an electronic version of a codebook cipher (no additive)
- A new codebook for each key

ECB Cut and Paste Attack

- Suppose plaintext is

  `Alice digs Bob. Trudy digs Tom.`

- Assuming 64-bit blocks and 8-bit ASCII:

  P_0 = "`Alice di`", P_1 = "`gs Bob. `",

  P_2 = "`Trudy di`", P_3 = "`gs Tom. `"

- Ciphertext: C_0, C_1, C_2, C_3
- Trudy cuts and pastes: C_0, C_3, C_2, C_1
- Decrypts as

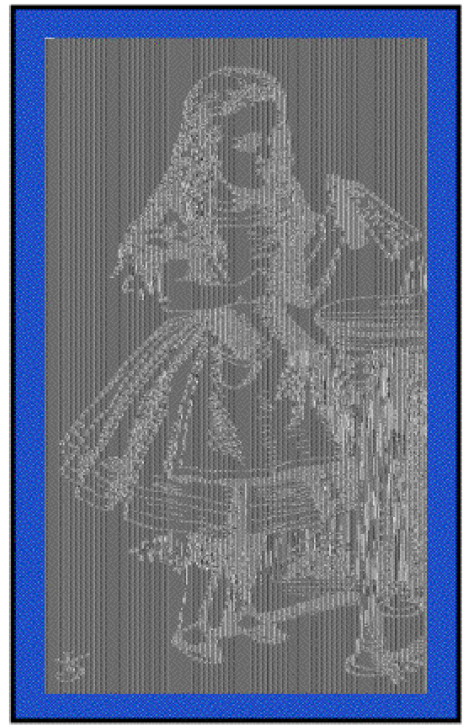
  `Alice digs Tom. Trudy digs Bob.`

ECB Weakness

- Suppose $P_i = P_j$
- Then $C_i = C_j$ and Trudy knows $P_i = P_j$
- This gives Trudy some information, even if she does not know P_i or P_j
- Trudy might know P_i
- Is this a serious issue?

Alice Hates ECB Mode

- Alice's uncompressed image, Alice ECB encrypted (TEA)



- Why does this happen?
- Same plaintext block ⇒ same ciphertext!

CBC Mode

- Blocks are "chained" together
- A random initialization vector, or IV, is required to initialize CBC mode
- IV is random, but need not be secret

Encryption

$C_0 = E(IV \oplus P_0, K),$
$C_1 = E(C_0 \oplus P_1, K),$
$C_2 = E(C_1 \oplus P_2, K),\ldots$

Decryption

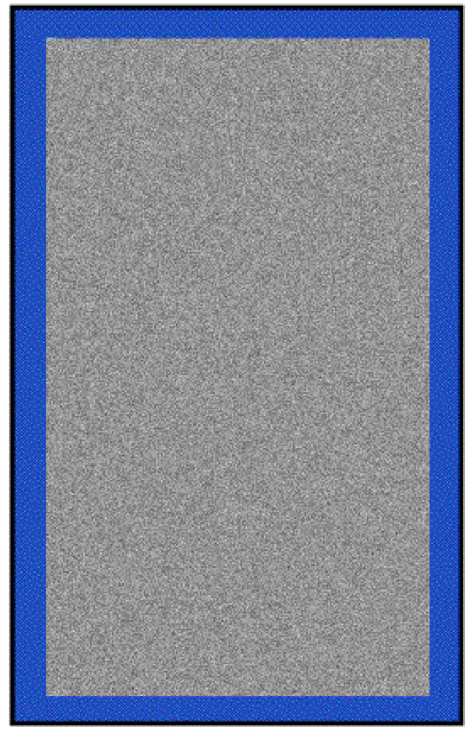
$P_0 = IV \oplus D(C_0, K),$
$P_1 = C_0 \oplus D(C_1, K),$
$P_2 = C_1 \oplus D(C_2, K),\ldots$

CBC Mode

- Identical plaintext blocks yield different ciphertext blocks
- Cut and paste is still possible, but more complex (and will cause garbles)
- If C_1 is garbled to, say, G then $P_1 \neq C_0 \oplus D(G, K)$, $P_2 \neq G \oplus D(C_2, K)$
- But $P_3 = C_2 \oplus D(C_3, K)$, $P_4 = C_3 \oplus D(C_4, K)$,…
- Automatically recovers from errors!

Alice Likes CBC Mode

- Alice's uncompressed image, Alice CBC encrypted (TEA)



- Why does this happen?
- Same plaintext yields different ciphertext!

Counter Mode (CTR)

- CTR is popular for random access
- Use block cipher like stream cipher

Encryption

$C_0 = P_0 \oplus E(IV, K),$
$C_1 = P_1 \oplus E(IV+1, K),$
$C_2 = P_2 \oplus E(IV+2, K), \ldots$

Decryption

$P_0 = C_0 \oplus E(IV, K),$
$P_1 = C_1 \oplus E(IV+1, K),$
$P_2 = C_2 \oplus E(IV+2, K), \ldots$

- CBC can also be used for random access!!!

Integrity

Data Integrity

- **Integrity** — prevent (or at least detect) unauthorized modification of data
- Example: Inter-bank fund transfers
  - Confidentiality is nice, but integrity is critical
- Encryption provides **confidentiality** (prevents unauthorized disclosure)
- Encryption alone does **not** assure integrity (recall one-time pad and attack on ECB)

MAC

- Message Authentication Code (MAC)
  - Used for data **integrity**
  - Integrity **not** the same as confidentiality
- MAC is computed as **CBC residue**
  - Compute CBC encryption, but only save the final ciphertext block

MAC Computation

- MAC computation (assuming N blocks)

  $C_0 = E(IV \oplus P_0, K)$,
  $C_1 = E(C_0 \oplus P_1, K)$,
  $C_2 = E(C_1 \oplus P_2, K), \ldots$
  $C_{N-1} = E(C_{N-2} \oplus P_{N-1}, K) = \text{MAC}$

- MAC sent along with plaintext
- Receiver does same computation and verifies that result agrees with MAC
- Receiver must also know the key K

Why does a MAC work?

- Suppose Alice computes

  $C_0 = E(IV \oplus P_0, K)$, $C_1 = E(C_0 \oplus P_1, K)$,
  $C_2 = E(C_1 \oplus P_2, K)$, $C_3 = E(C_2 \oplus P_3, K) = MAC$

- Alice sends IV, P_0, P_1, P_2, P_3 and MAC to Bob
- Trudy changes P_1 to X
- Bob computes

  $C_0 = E(IV \oplus P_0, K)$, $\mathbf{C_1} = E(C_0 \oplus X, K)$,
  $\mathbf{C_2} = E(\mathbf{C_1} \oplus P_2, K)$, $\mathbf{C_3} = E(\mathbf{C_2} \oplus P_3, K) = \mathbf{MAC} \neq MAC$

- **Propagates** into ***MAC*** (unlike CBC decryption)
- Trudy can't change ***MAC*** to MAC without K

Confidentiality and Integrity

- Encrypt with one key, MAC with another
- Why not use the same key?
  - Send last encrypted block (MAC) twice?
  - Can't add any security!
- Use different keys to encrypt and compute MAC; it's OK if keys are related
  - But still twice as much work as encryption alone
- Confidentiality and integrity with one "encryption" is a research topic

Uses for Symmetric Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Integrity (MAC)
- Authentication protocols (later…)
- Anything you can do with a hash function (upcoming chapter…)

Feistel Cipher

- **Feistel cipher** refers to a type of block cipher design, not a specific cipher
- Split plaintext block into left and right halves: Plaintext = (L_0,R_0)
- For each round i=1,2,…,n, compute

  $L_i = R_{i-1}$

  $R_i = L_{i-1} \oplus F(R_{i-1},K_i)$

  where F is **round function** and K_i is **subkey**
- Ciphertext = (L_n,R_n)

Feistel Cipher

- Decryption: Ciphertext = (L_n, R_n)
- For each round i=n,n−1,…,1, compute

  $R_{i-1} = L_i$

  $L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$

  where F is round function and K_i is subkey
- Plaintext = (L_0, R_0)
- Formula "works" for any function F
- But only secure for certain functions F

Conclusions

- Block ciphers widely used today
- Fast in software, very flexible, etc.
- Not hard to design strong block cipher
- Tricky to design fast and secure block cipher
- Next: CMEA, Akelarre and FEAL