CS-4920: Lecture 8
Modes of operation

- Reading
  - Chapter 4
- Today's Outcomes
  - Explain various methods for applying secret key (block) encryption to a message stream
  - Using secret key techniques to generate MACs (message authentication codes)

1

Encrypting a long message

- Electronic Code Book (ECB)
  - Obvious approach, usually the worst choice
- Message is broken into 64-bit blocks
  - Each is encrypted independently
- Weaknesses
  - Information leakage
    - Identical ct blocks mean identical pt
  - Attacker can copy data at the block level (*e.g.*, boss' salary to his), even if he does not know the exact values encrypted

2

Cipher Block Chaining (CBC)

- ECB would be improved if we picked a (not necessarily secret) long random number and XORed the entire message with it
  - Benefit: solved ECB information leakage
    - But, I can still delete or rearrange blocks
    - And, if I know the pt, I can modify the random block to change the decrypted pt
  - Cost: Twice as much data to transmit!
- CBC addresses the cost problem

3

CBC

- Replace random number for each block with the ct of the previous block
- Still need an initial random number
  - IV – initialization vector, transmit to receiver
  - Otherwise, messages that begin the same still encrypt the same
- [encrypt] $c_n = K\{m_n \text{ XOR } c_{n-1}\}$
- [decrypt] $m_n = c_{n-1} \text{ XOR } (m_n \text{ XOR } c_{n-1})$
- Use IV in place of c_0
- Regularly changing IV is important
  - Otherwise, can detect identical messages

4

A CBC threat – modifying ciphertext blocks

- Can no longer swap/copy blocks
  - Because of the chaining
- But, can change particular bits
  - At the cost of mangling bits in previous block
  - Useful if you know some of the plaintext
- Want to change bit(s) in m_n
  - Toggle corresponding bit(s) in c_{n-1}
    - Damages m_{n-1}, forges m_n
  - Do not change c_n
    - Messages m_{n+1} and following are unaffected

5

Output feedback mode (OFB)

- Converts a block cipher (*e.g.*, DES) into a stream cipher
  - By generating a 1-time pad
- Start with IV = b_0
- $b_{n+1} = K\{b_n\}$
- XOR message with bits as needed before transmission to get ciphertext

6

OFB

- Advantages
  - Generate 1-time pad in advance
    - Encryption is a simple XOR
  - Garbling is restricted to individual bits, not blocks as in CBC
  - The transmitted block sizes can be smaller or larger than 64 bits
- Disadvantage
  - Known plaintext attack
    - Replace c with c′ = c XOR (m XOR m′) to cause recipient to receive m′

7

Generalization: k-bit OFB

- Start with IV = b_0
- $e_n = K\{b_{n-1}\}$
- a_n = k MSBs of e_n (discard the rest)
  - XOR this with message
- $b_{n+1} = (b_n << k) \mid a_n$
- *E.g.*, Size of e and b is 64 for DES
  - k might be 8

8

k-bit Cipher Feedback Mode (CFB)

- Like k-bit OFB, but the bits shifted in are the ciphertext bits, not the 'a' bits
- Can use as a stream cipher (k bits at a time)
  - But not precomputed, unlike OFB
- Advantages over OFB
  - Can resynchronize if M·k bits dropped from stream
    - | in the received c_n
    - not the computed a_n which are not synchronized between sender and receiver
  - Decryption uses only encryption operation
  - More robust to known pt substitution? Debatable
    - Modifying a byte garbles the next N/k bytes

9

Counter mode (CTR)

- Start with IV = b_0
- $a_{n+1} = K\{b_n\}$
  - XOR message with bits as needed before transmission to get ciphertext
- $b_{n+1} = b_n+1$
- Advantages
  - Can precompute a 1-time pad
  - Random access! (without repeated encryption)
- Disadvantages
  - Same key and IV reuses pad
  - Can get XOR of two plaintext blocks by XOR of ct blocks
  - Toggle specific bits by togging corresponding bits in ct

10

Generating MACs

- MAC – message authentication code
  - A cryptographic checksum
  - Can make with a secret key system
- Feedback/counter modes of encryption
  - Protect against deciphering
  - But not very well against undetected modification
- Standard MAC
  - Calculate CBC but send only the last block with the plaintext message (CBC residue)
    - Only someone who knows the secret key could forge (or calculate) this MAC
  - Works well for non-secret messages whose integrity must be ensured

11

What about confidentiality and integrity together?

- The residue is just the final CBC output, so does this mean we already have C and I?
  - No, if you have the whole CBC stream, integrity is lost – bits may be modified
- There are variations (checksum at end encrypted, …), but they all have practical or at least theoretical weaknesses
- Solutions
  - Separate keys for the CBC stream and residue
  - Related keys are believed secure as well (*e.g.*, XOR with 0xF0F0F0…)
  - Encrypt a cryptographic hash (different than CRC – hard to find messages that give the same hash)

12

Multiple encryption DES

- DES, at 56-bits, may be too weak
- So, do multiple DES passes
- Accepted method is
  - EDE (encrypt-decrypt-encrypt)
  - Also called 3DES
- Only two keys: K_1 and K_2
- Encrypting: Es with K_1, D with K_2

13

How many encryptions?

- Encrypting twice with the same key
  - Only doubles work for bad guy
- Encrypting twice with 2 keys
  - Weakness with known plaintext
    - Table with E(pt) and D(ct) for first block with each possible key
    - Find a match
    - 2^{64} blocks and 2^{56} keys, so 1/256 chance of a false alarm
      - Check next block
- Why not use 3 keys?
  - 112 bits is enough?
  - Forcing K1=K3 allows fallback to plain DES if we also have K1 = K2

14