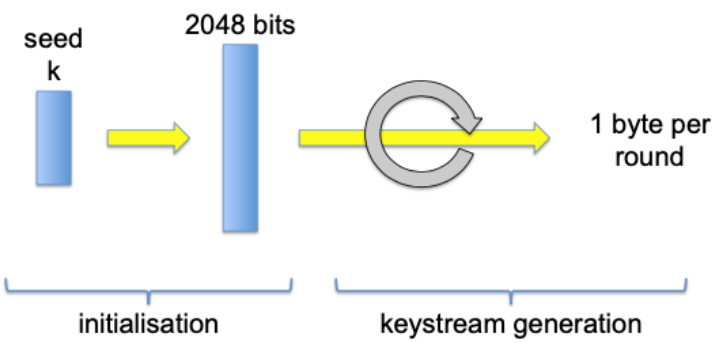
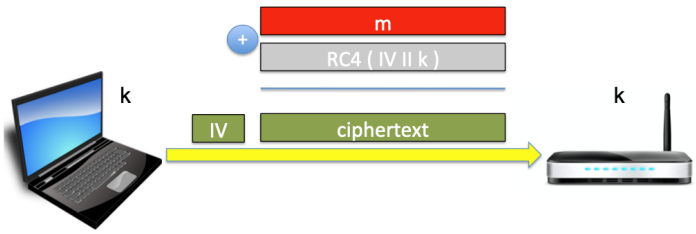
CS Revision Lecture 7

- ## Lecture 7 - Cryptography: Symmetric encryption
  - ### Stream Ciphers
    - Goal: make the OTP practical
    - Idea: use a pseudorandom key rather than a really random key
      - The key will not really be random, but will look random
      - The key will be generated from a key seed using a **Pseudo-Random Generator (PRG)**
        - $G : \{0,1\}^s \rightarrow \{0,1\}^n$ with s << n
    - Encrpytion using a PRG
      - G: $E(k, m) = G(k) \oplus m$
    - Decryption using a PRG
      - G: $D(k, c) = G(k) \oplus c$
    - **Stream ciphers are subject to two-time pad attacks**
    - **Stream ciphers are malleable**
    - RC4
      - https://www.youtube.com/watch?v=1UP56WM4ook (good explaination)
      - Stream cipher invented by Ron Rivest in 1987
      - Consists of 2 phases



- Main data structure: array S of 256 bytes
- Used in HTTPS and WEP(**W**ired **E**quivalent **P**rivacy)
- Weaknesses of RC4
  - first bytes are biased
    - solution: drop the first to 256 generated bytes
  - subject to related keys attacks
    - soulution: choose randomly generated keys as seeds

- WEP uses RC4



- Initialisation Vector (IV): 24-bits long string
- RC4 as all stream ciphers is subject to two time-pads attacks - that is the same key should not be used twice. So in turn you need a different seed to the RC4 PRNG everytime you encrypt a new message.
- To make RC4 practical in the context of WEP we use the IV (**a randomly sampled 24-bits bitstring**) for **randomising the seed to RC4.** The router needs to know the IV for decrypting though, and this is why it is appended to the ciphertext.

- Weaknesses of WEP
  - two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
    - **Solution: use longer IVs**
  - Fluhrer, Mantin and Shamir (FMS) attack (related keys attack):
    - the keys only differ in the 24 bits IV
    - first bytes of key stream known because standard headers are always sent

- for certain IVs knowing m bytes of key and keystream means you can deduce byte m + 1 of key
- **Solution: instead of using related IVs, generate IVs using a PRG**

- Modern stream ciphers
  - Project eStream: project to “identify new stream ciphers suitable forwidespread adoption”, organised by the EU ECRYPT network
    - HC-128, Rabbit, Salsa20/12, SOSEMANUK, Grain v1, MICKEY 2.0, Trivium
  - Conjecture
    - These eStream stream ciphers are “secure”
- Concluding remarks on Stream Ciphers
  - Perfect secrecy does not capture all possible attacks.
    - **need for different security definition**
  - Theorem (Shannon 1949) Let (E,D) be a cipher over (M,C,K). If(E,D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|M| \leq |K|$).
    - Stream ciphers do not satisfy perfect secrecy because the keys in K are smaller than the messages in M
      - **need for different security definition**
  - The design of crypto primitives is subtle and error prone.
    - **use standardised publicly know primitives**
  - Crypto primitives are secure under a precisely defined threat model.
    - **respect the security assumptions of the crypto primitives**
  - Many attacks dues to poor implementations of cryptography

- Block Ciphers
  - A block cipher with parameters k and l is a pair of deterministic algorithms (E, D) such that

- Encryption E: $\{0,1\}^k$ x $\{0,1\}^l \rightarrow \{0,1\}^l$
- Decryption D: $\{0,1\}^k$ x $\{0,1\}^l \rightarrow \{0,1\}^l$

- **Examples:**
  - **3DES: l = 64, k = 168**
  - **AES: l = 128, k = 128, 192, 256**

- Data Encryption Standard (DES)
  - Early 1970s: Horst Feistel designs Lucifer at IBM

    k = 128 bits, l = 128 bits
  - 1973: NBS calls for block cipher proposals.
    - IBM submits a variant of Lucifer.
  - 1976: NBS adopts DES as a federal standard

    k = 56 bits, l = 64 bits
  - 1997: DES broken by exhaustive search
  - 2001: NIST adopts AES to replace DES

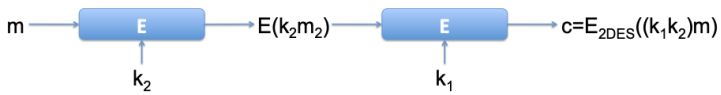
    k = 128, 192, 256 bits, l = 128 bits

- Attacks on DES
  - **Exhaustive search**: it takes 2^{56} to do an exhaustive search over the key space
    - COBACOBANA (120 FPGAs, ~ 10K$): 7 days

      FPGA (Field-programmable gate array)
  - **Linear cryptanalysis**: found affine approximations to DES
    - can find 14 key bits in time 2^{42}
    - brute force the remaining 56-14=42 in time 2^{42}
    - total attack time $\approx 2^{43}$
  - **DES is badly broken! Do not use it in new projects**

- Triple DES (3DES)

- Goal: build on top of DES a block cipher **resistant against exhaustive search attacks**
- Used in bank cards and RFID chips
  - RFID (Radio-frequency identification)
- Let *DES* = (E_{DES}, D_{DES}). We build *3DES* = (E_{3DES}, D_{3DES}) as follows
  - $E_{3DES} : ((\{0,1\}^k)^3 \times \{0,1\}^l \rightarrow \{0,1\}^l$
    - Encrypt plaintext using , Decrypt it using then Encrypt using
    - $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}((K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
      - if $K_1 = K_2 = K_3$ Then 3DES = DES
  - $D_{3DES} : ((\{0,1\}^k)^3 \times \{0,1\}^l \rightarrow \{0,1\}^l$
    - Decrypt plaintext using , Encrypt it using then Decrypt using
    - $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}((K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$
  - **3 times as slow as DES**
  - Key size = 3 x 56 = 168 bits
    - Exhaustive search attack in 2^{168}
  - simple (meet-in-the-middle) attack in time 2^{118}
- What about double DES (2DES)?
  - $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



    - For m and c such that $E_{2DES}((K_1, K_2), M)$ = c we have that $E_{DES}(K_2, M) = D_{DES}((K_1, C)$
  - 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to

2^{56}. Given $M = (m_1, \ldots, m_{10})$ and C = $(E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((*k_1, k_2), m_{10}))$

- For all possible k_2, $computeE_{DES}(k_2, M)$
- Sort table according to the resulting $E_{DES}(k_2, M)$

} $2^{56}\log(2^{56})$

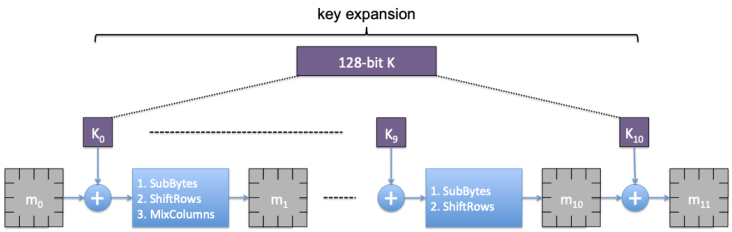
- For each possible k_1, $computeD_{DES}(k_1, C)$
- Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

} $2^{56}\log(2^{56})$

$\Rightarrow$ time $< 2^{63}$

- Similar attack on 3DES in time 2^{118}

- The Advanced Encryption Standard (AES)
  - Goal:
    - replace 3DES which is too slow
  - **Block size *l* = 128 bits, Key size *k* = 128, 192, 256 bits**
  - **Encryption circuit**

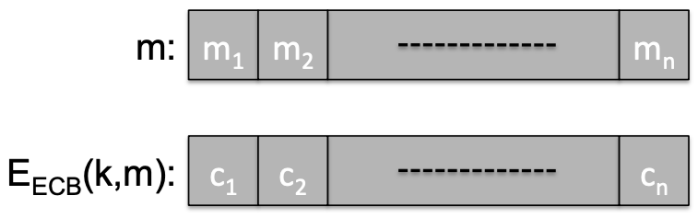


    - m_i : 4 x 4 byte matrix, K_i : 128-bit key
    - m_0 : plaintext, m_{11} : ciphertext
    - at the last round MixColumns is not applied
  - **Attacks on AES**
    - **Related-key attack**
      - on the 192-bit and 256-bit versions of AES:exploits the AES key schedule
      - Key recovery in time ~ 2^{99}
    - **First key-recovery attack** on full AES
      - 4 times faster than exhaustive search
    - Existing attacks on AES-128 are still not practical, but should use AES-192 and AES-256 in new projections
    - Additional protection against quantum computers
- Using block ciphers

- Goal
  - **Encrypt *M* using a block cipher operating on blocks of length *l* when $|M| \neq l$**
- Padding - $|M| \leqslant l$
  - **Bit padding**
    - **append a set bit ('1') at the end of message, and then append as many reset bits ('0') required**
    - Example:
      - padding a 52-bits message for a 64-bits block:
        - 11010011 01010110 10010000 00111010 10110101 01011010 1111**1000 00000000**
      - padding a 64-bits message M for 64-bits blocks requires adding a padding block:
        - M | **10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000**
  - **ANSI X.923**
    - **byte padding - pad with zeros, the last byte defines the number of padded bytes**
    - Example
      - padding a 4-bytes message for 8-bytes blocks:
        - DD DD DD DD **00 00 00 04**
      - padding a 8k-bytes messages for 8-bytes blocks requires adding a padding block:
        - DD DD DD DD DD DD DD DD | **00 00 00 00 00 00 00 08**
  - **PKCS#7**
    - byte padding - the value of each added byte is the total number of padding bytes. The padding will be 01, or 02 02, or 03 03 03, or 04 04 04 04, etc.
    - Example

- padding a 4-bytes message for 8-bytes blocks:
  - DD DD DD DD **04 04 04 04**
- padding a 8k-bytes messages for 8-bytes blocks requires adding a padding block:
  - DD DD DD DD DD DD DD DD | **08 08 08 08 08 08 08 08**

- Electronic Code Book (ECB) mode
  - (*E, D*) a block cipher
  - **To encrypt message *M* under key *K* using ECB mode:**
    - **M is padded:**
      - $\Longrightarrow$ $M' = M \,||\, P$ such that $|M'| = m \times l$
    - ***M'* is broken into m blocks of length *l***
      - $\Longrightarrow$ $M' = M_1 \,||\, M_2 \,||\, ... \,||\, M_m$
    - **Each block M_i is encrypted under the key *K* using the block cipher**
      - $\Longrightarrow$ $C_i = E\,(\,K, M_i\,)$ for all $i \in \{1, ..., m\}$
    - **The ciphertext corresponding to *M* is the concatenation of the C_is**
      - $\Longrightarrow$ $C = C_1 \,||\, C_2 \,||\, ... \,||\, C_m$
  - **Weakness of ECB**



    - Problem: $\forall\, i, j . m_i = m_j \implies c_i = E(k, m_i) = E(k, m_j) = c_j$
    - $\Longrightarrow$ **Malleable and weak to frequency analysis**
  - **Weakness of ECB in pictures**

Original image

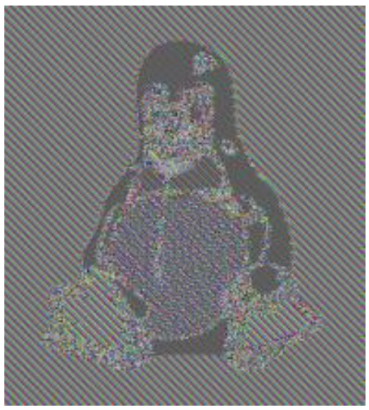
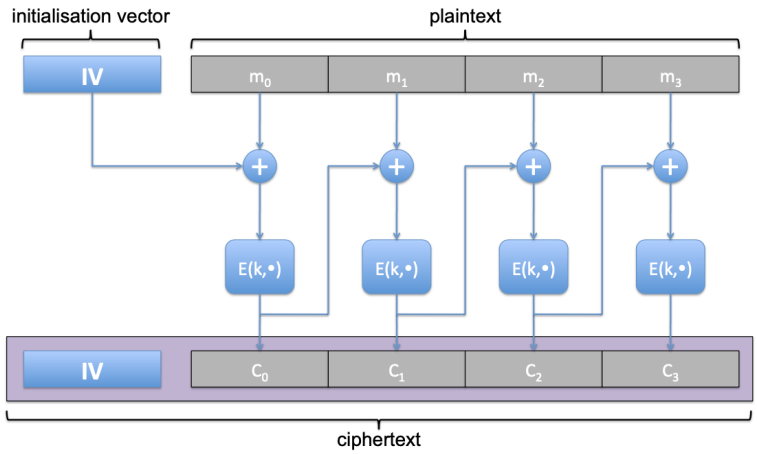
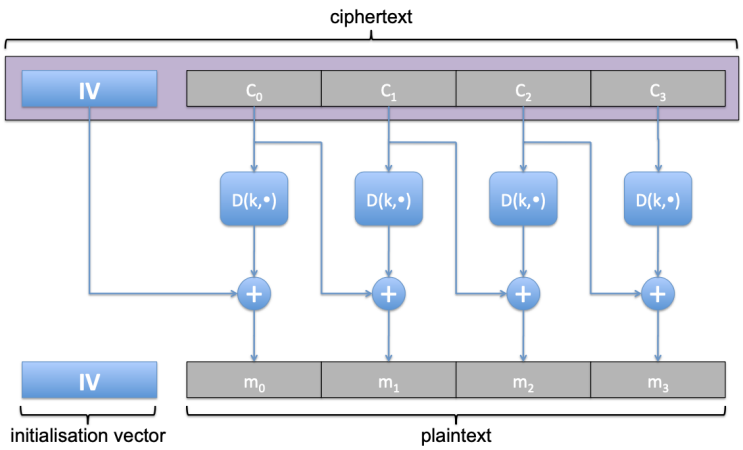


Image encrypted using ECB mode

- Cipher-block chaining (CBC) mode
  - **Encryption**
    - (E, D) a block cipher that manipulates blocks of size l



    - IV chosen at random in $\{0,1\}^l$
  - **Decryption**

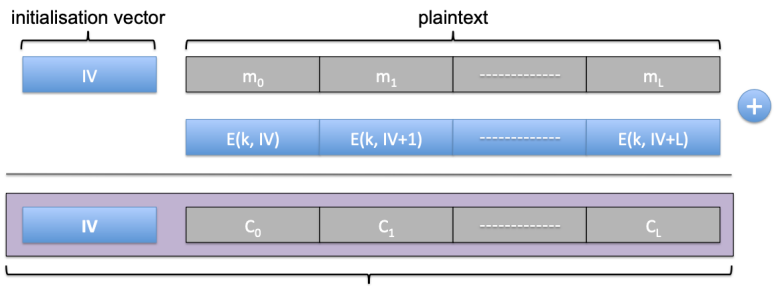


  - **Sony Playstation**
    - Prevent games being copied

- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game
- Sony PS used ECB full-disk encryption
- Hardware controlled user access to data

- **Sony Playstation disk encryption attack**
  - Remove disk and make copy
  - Put disk back in PlayStation
  - Copy a file to the disk
  - Remove disk and find area of disk that changed (that is the user encrypted file)
  - Copy target data to the user area
  - Put disk back in PlayStation and ask for user data
  - PlayStation decrypts the file and gives it to user

- Counter (CTR) mode
  - (E, D) a block cipher that manipulates blocks of size *l*



  - IV chosen at random in $\{0, 1\}^l$

- Block-size is also a problem
  - Sweet32 - birthday attacks on 64-bit block ciphers in TLS and openVPN
  - Attack due to block-size being too small

- The key management problem

- The confidentiality problem is now reduced to a key management problem:
  - Where are keys generated?
  - How are keys generated?
  - How are keys shared?
  - Where are keys stored?
  - Where are the keys actually used?
  - How are key revoked and replace?

- What we have learned on Symmetric Encryption
  - Frequency analysis as a cryptanalysis attack on classic encryption
  - Importance of randomness in cryptography
  - Stream ciphers
    - simple and efficient symmetric encryption schemes
    - use a random IV to thwart two-time pad attacks
    - good for random plaintexts otherwise subject to malleability attacks
  - Block ciphers - use AES not DES
  - CBC mode is more secure that ECB but less resilient to packets loss
  - CTR mode more secure than ECB and parallelisable
  - Keep up to date with cryptanalysis advances and standards
  - Do not implement crypto lightly - use public reference implementations

以上内容整理于 幕布文档