51.505 – Foundations of Cybersecurity

Week 5 - Symmetric Encryption

Created by **Pawel Szalachowski** (2017)
Modified by **Jianying Zhou** (2018)

Last updated: 23 Sept 2018

Review of Exercises

- Mid-term exam (Week 6): Fri 19 Oct, 7:30 PM (covering Part I Foundations: Week 1 – Week 4)
- Let’s review some exercises from Week 1 to Week 4.

Cryptography

- Cryptography: art and science of encryption (ciphers)
- More than encryption (other primitives)
  - ✓ hash functions, MACs, (P)RNG, RSA, DH
- Higher-level constructions
  - ✓ secure channel, key server, PKI
- Real-world systems ?

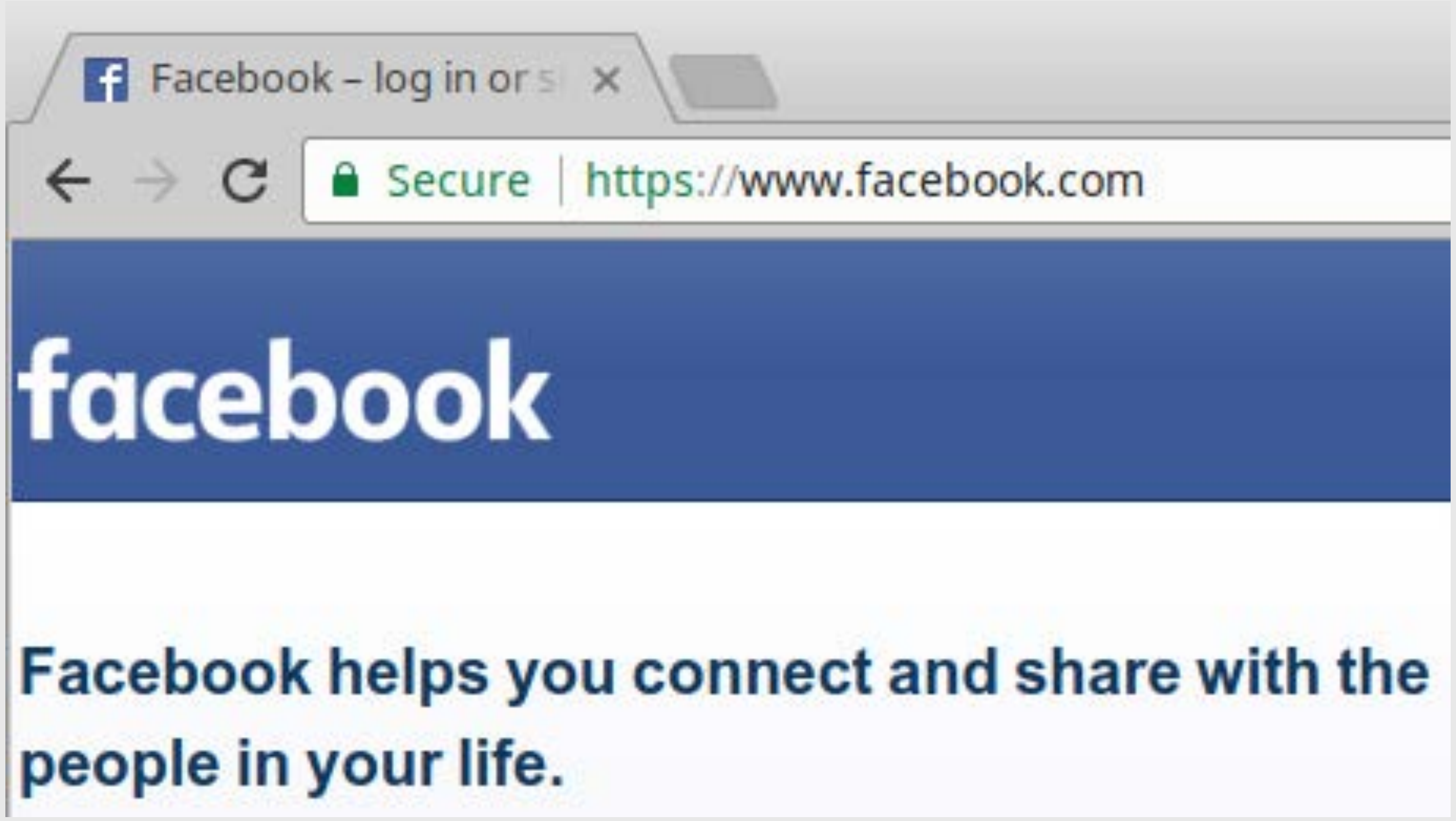
Cryptography

- Threat model
  - ✓ Understand what and against whom you are trying to protect
- Cryptography is very difficult.
  - ✓ Proofs but with many assumptions, implementation issues, side-channel attacks, security vs. performance
- Cryptography is the easy part.
  - ✓ Systems are very complex, while a cryptographic component has fairly well-defined boundaries and requirements.
- Cryptography only solves some security problems.

How it happens?

- Secure communication

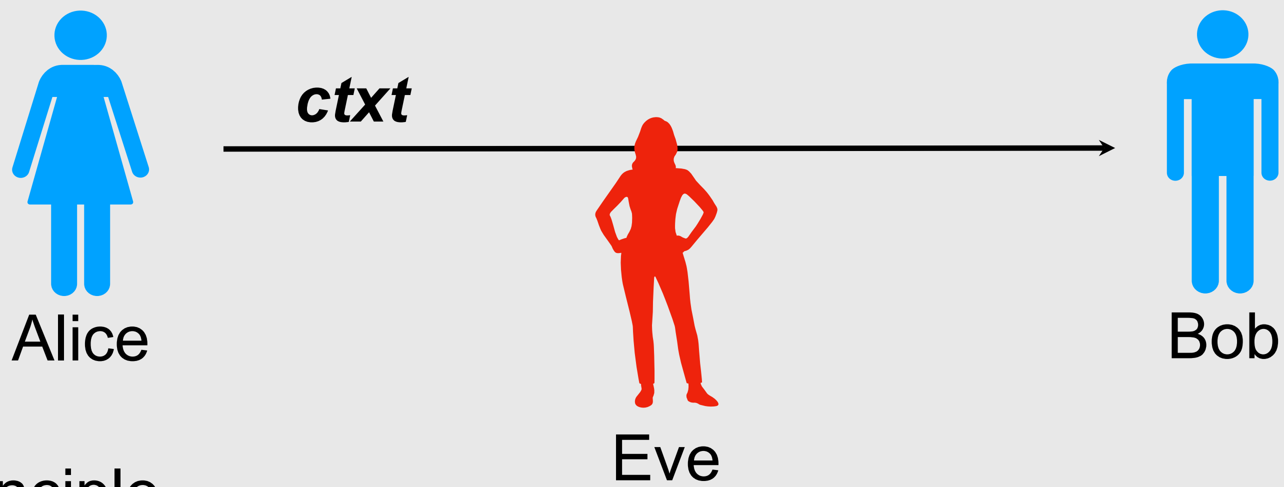
  ✓ Client-Server via HTTP**S**



Symmetric Encryption

- Encryption scheme
  - ✓ Encryption and decryption algorithms: ***E( )*** and ***D( )***
- Alice, Bob, and Eve
  - ✓ Alice and Bob share a secret (symmetric) key: ***K***
  - ✓ Eve sees all (encrypted) communication.

$\boldsymbol{ctxt} = E_K(ptxt)$



$ptxt = D_K(\boldsymbol{ctxt})$

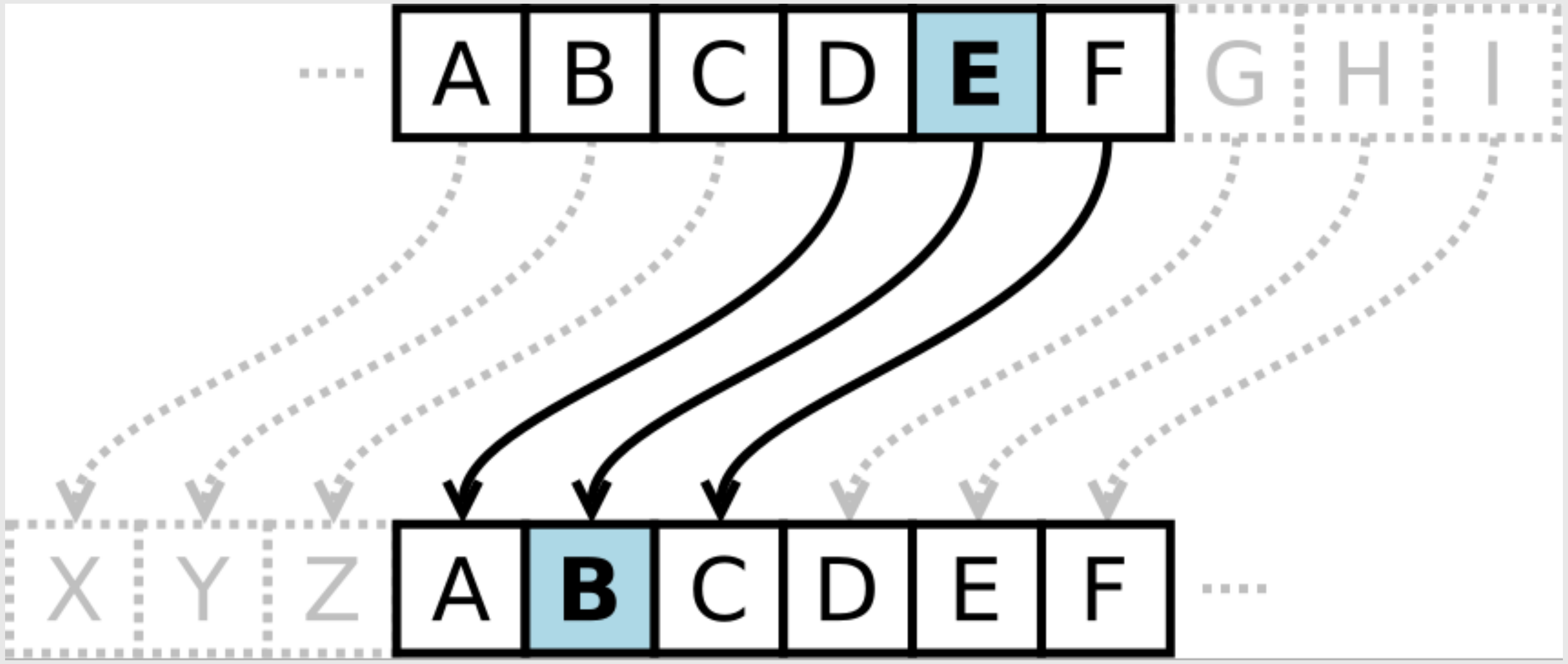
- <u>Kerckhoffs' Principle</u>
  - ✓ *The security of the encryption scheme must depend only on the secrecy of the key, and not on the secrecy of the algorithm.*

Caesar Cipher

- A substitution cipher, where each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.

  ✓ This fixed number of positions is the secret key.

  ✓ What’s the key in the following example?



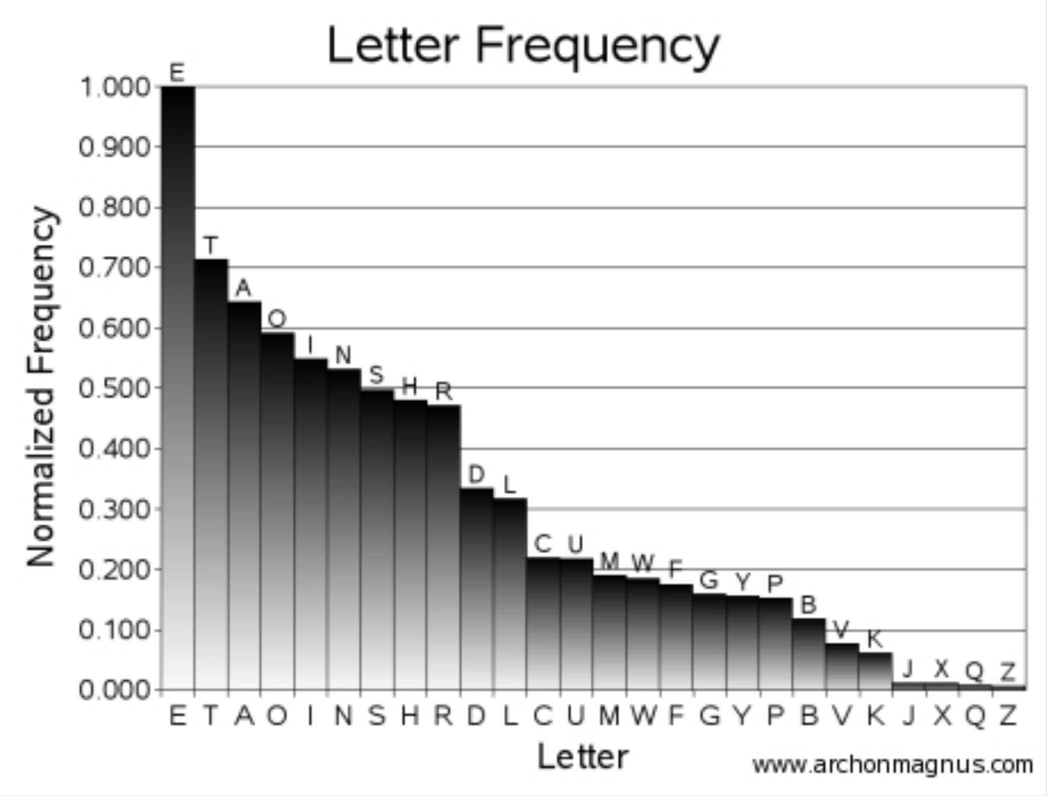
Other Substitution Ciphers

- Monoalphabetic

  Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

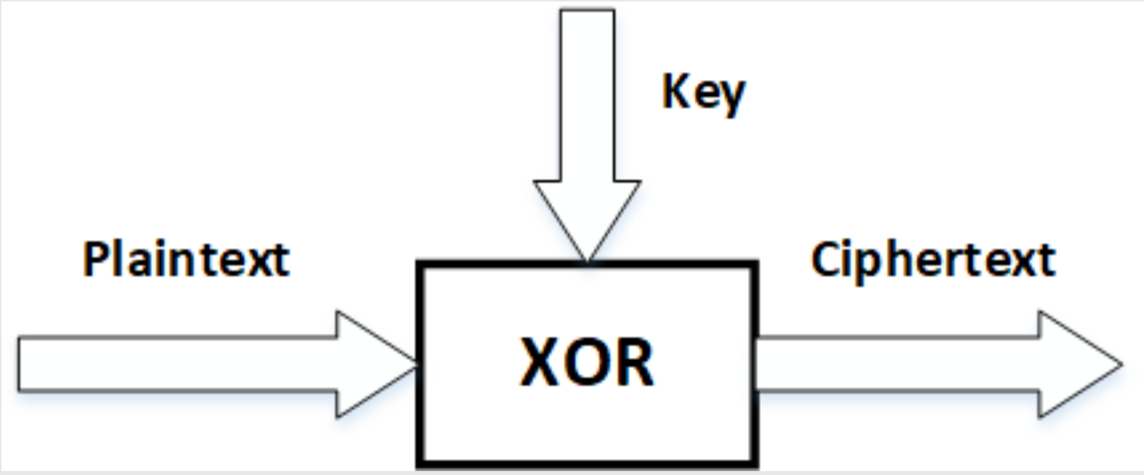
  Key: PDKIFMRBHSONCGXUTJWEYLQAZV

- Trivial to break with letter frequency



One-Time Pad

- Key is *random*, as long as plaintext (at least), and is used only once.



- This scheme **cannot** be *broken.*
    - ✓ No matter how strong an adversary is, she cannot learn anything about plaintext.
- Disadvantages?

Attacks

Goal: to discover the key.

- Ciphertext-Only Attack (COA)
  - ✓ Eve knows only ciphertexts (without the corresponding plaintexts).
- Known-Plaintext Attack (KPA)
  - ✓ Eve knows some (plaintext, ciphertext) pairs.
- Chosen-Plaintext Attack (CPA)
  - ✓ Eve can select plaintexts and obtain the corresponding ciphertexts.
- Chosen-Ciphertext Attack (CCA)
  - ✓ Eve can select plaintexts and/or ciphertexts and obtain the corresponding ciphertexts and/or plaintexts.

More powerful attacks

Security Level

- Exhaustive search (brute-force) attack: an adversary tries all possible values for some target object (like the key).

- If an attack requires 2^n steps of work, then it is corresponding to an exhaustive search for a n-bit value. Example via keylength.com:

Date	Symmetric	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash	
2017 - 2022	128	2000	250	2000	250	SHA-256 SHA-512/256 SHA-384 SHA-512	SHA3-256 SHA3-384 SHA3-512
> 2022	128	3000	250	3000	250	SHA-256 SHA-512/256 SHA-384 SHA-512	SHA3-256 SHA3-384 SHA3-512

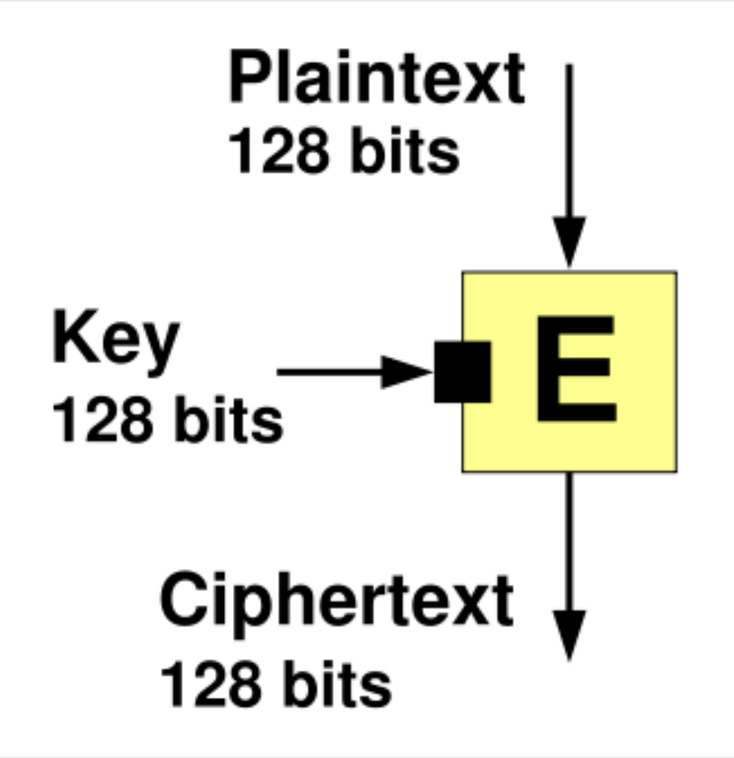
- The level of security is usually a function of the access of the adversary (e.g. how many encrypted messages she sees).

Modern Ciphers

- **Block ciphers**
  - ✓ Operate on data blocks
  - ✓ AES, DES, Serpent, ...
- Stream ciphers
  - ✓ Operate on data streams
  - ✓ RC4, Salsa20, ...

Block Cipher

- An encryption/deception function for fixed-size blocks of data.
  - ✓ Encryption function (E_K) for a secret key and a plaintext block returns the cipertext (one block).
  - ✓ Decryption function (D_K) for the secret key and the ciphertext block *reverts* the plaintext block.
  - ✓ Currently, 128 bits is the most common block size and key lengths are usually between 128 - 512 bits.



AES (Rijndael)

- The Advanced Encryption Standard (AES)
  - ✓ Standardized (2001) and the most popular
  - ✓ Hardware support in recent CPUs
  - ✓ Blocks are 128-bit long
  - ✓ Keys can be 128-, 192-, or 256-bit long

AES Internals

```
AddRoundKey(0)

for round in range(1, Nr):
     SubBytes()
     ShiftRows()
     MixColumns()
     AddRoundKey(round)

SubBytes()
ShiftRows()
AddRoundKey(Nr)
```

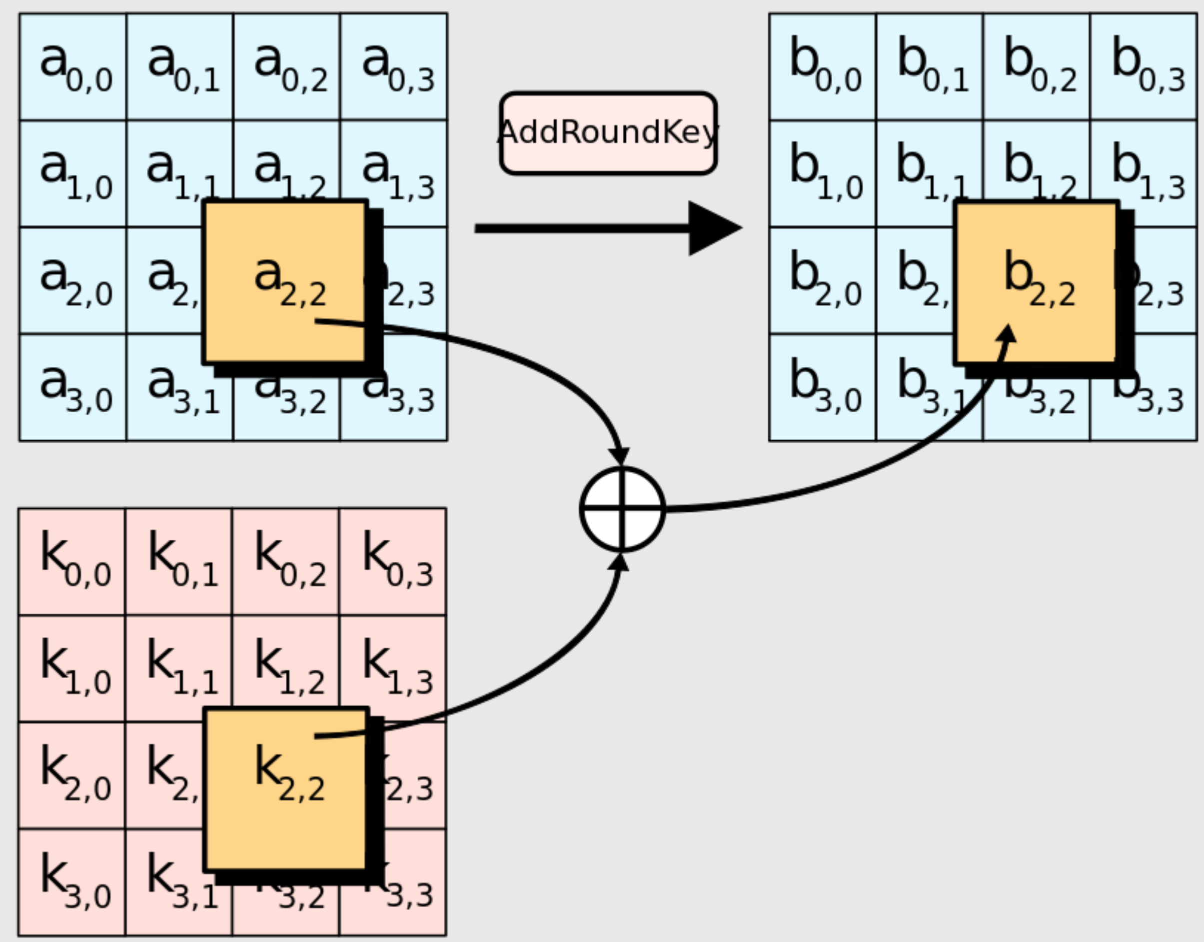
- Substitution-permutation network.
- Number of rounds (N_r) depends on key length.
  - ✓ $N_r = 10$ for 128-bit keys
  - ✓ $N_r = 12$ for 192-bit keys
  - ✓ $N_r = 14$ for 256-bit keys
- Before execution the key expansion procedure is called to derive N_{r+1} subkeys.
- A set of reverse rounds is applied to transform a ciphertext back into the plaintext.

AES Internals

```
AddRoundKey(0)

for round in range(1, Nr):
     SubBytes()
     ShiftRows()
     MixColumns()
     AddRoundKey(round)

SubBytes()
ShiftRows()
AddRoundKey(Nr)
```



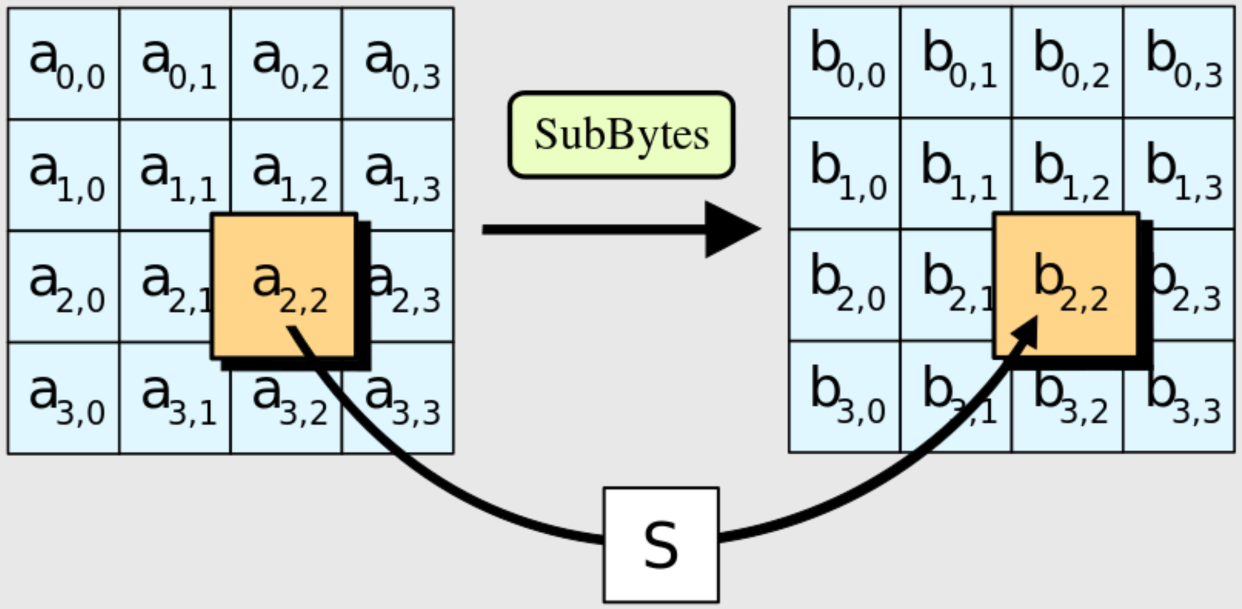
- Each byte of the state is combined (***XOR**ed*) with a byte of the round subkey.

AES Internals

```
AddRoundKey(0)

for round in range(1, Nr):
      SubBytes()
      ShiftRows()
      MixColumns()
      AddRoundKey(round)

SubBytes()
ShiftRows()
AddRoundKey(Nr)
```



- Each byte in the state is replaced with its entry in a fixed 8-bit lookup table S.
- The goal is to provide the *non-linearity* in the cipher.

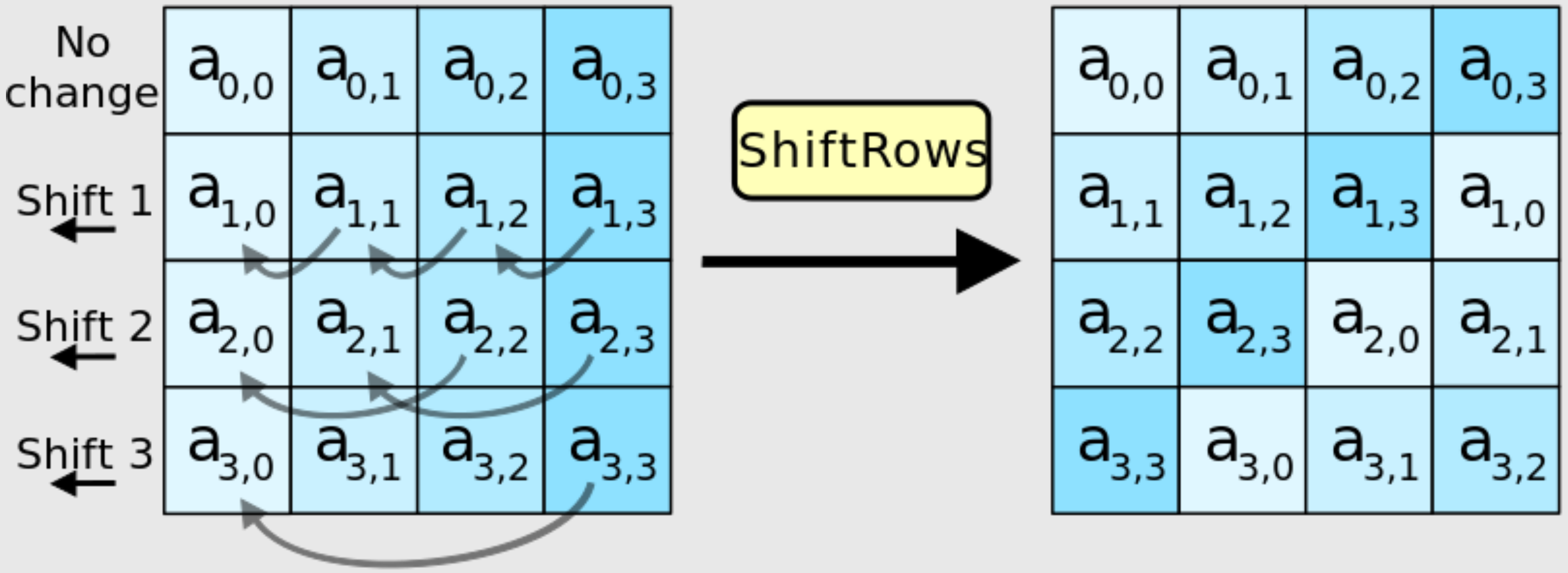
AES Internals

```
AddRoundKey(0)

for round in range(1, Nr):
    SubBytes()
    ShiftRows()
    MixColumns()
    AddRoundKey(round)

SubBytes()
ShiftRows()
AddRoundKey(Nr)
```

- Bytes in each row of the state are shifted cyclically to the left.
- The goal is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.

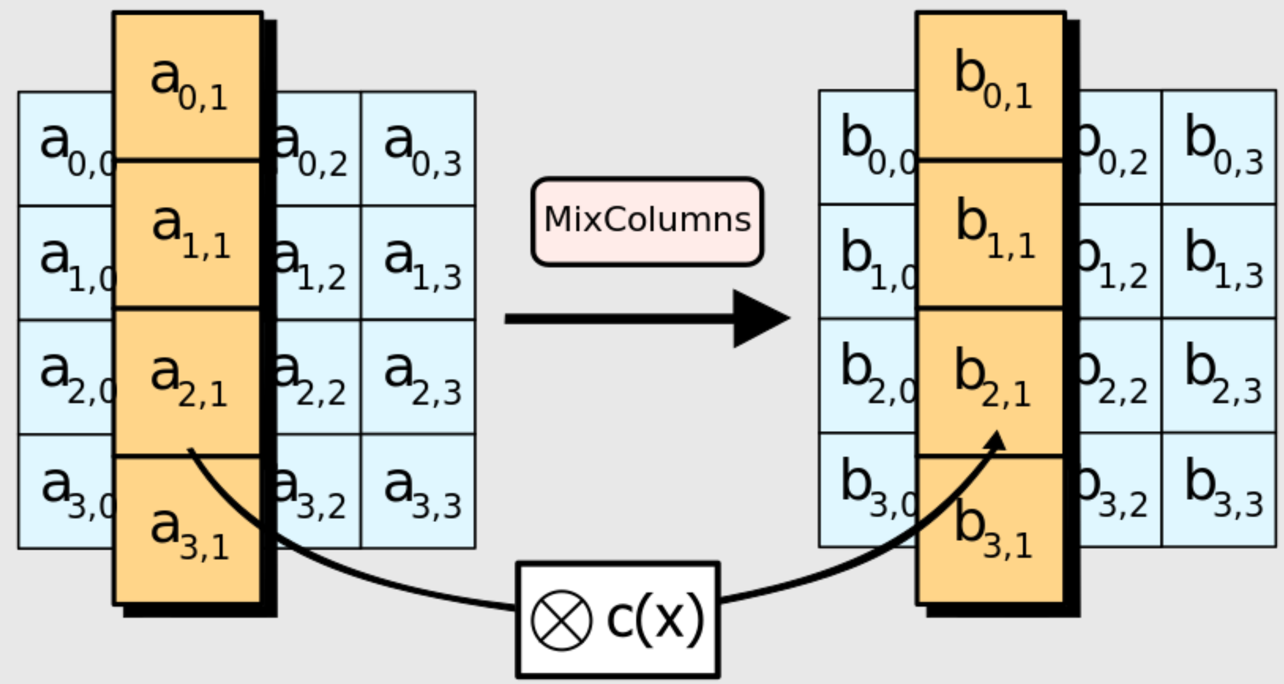


AES Internals

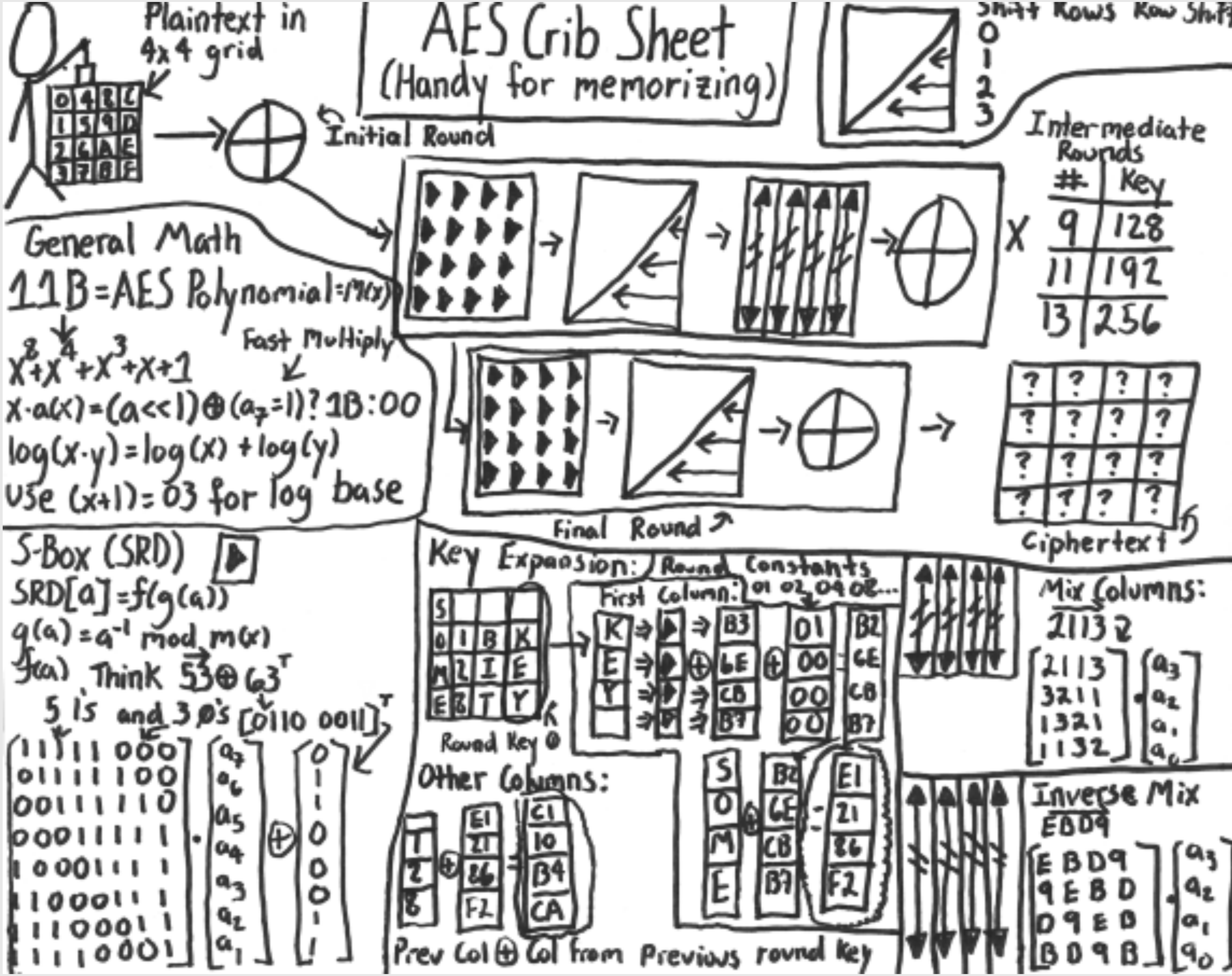
```
AddRoundKey(0)

for round in range(1, Nr):
    SubBytes()
    ShiftRows()
    MixColumns()
    AddRoundKey(round)

SubBytes()
ShiftRows()
AddRoundKey(Nr)
```



- Each column of the state is multiplied with a fixed polynomial *c(x)*.
- The goal is to provide *diffusion* in the cipher.

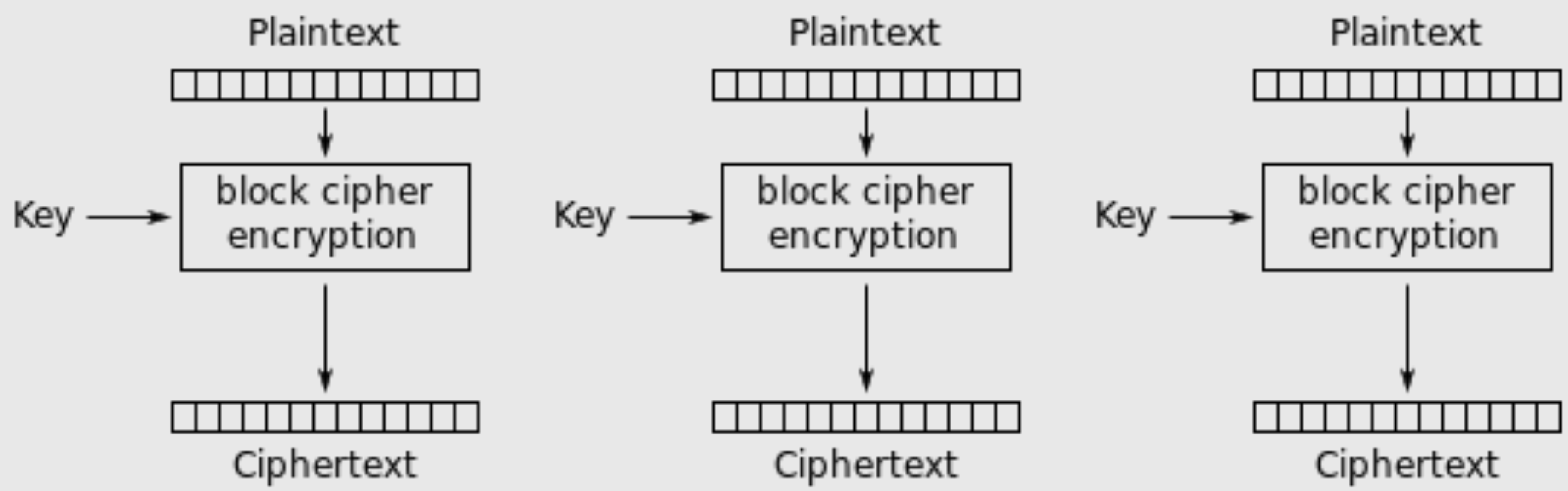

AES Crib Sheet
(Handy for memorizing)
Plaintext in 4x4 grid
0 4 8 C
1 5 9 D
2 6 A E
3 7 B F
Initial Round
Shift Rows Row Shift
0
1
2
3
Intermediate Rounds
Key
9 128
11 192
13 256
X
General Math
11B = AES Polynomial = m(x)
x^8+x^4+x^3+x+1
Fast Multiply
x·a(x) = (a<<1) ⊕ (a7=1)? 1B:00
log(x·y) = log(x) + log(y)
Use (x+1) = 03 for log base
Final Round
Ciphertext
? ? ? ?
S-Box (SRD)
SRD[a] = f(g(a))
g(a) = a^-1 mod m(x)
f(a) Think 53 ⊕ 63^T
5 1's and 3 0's [0110 0011]^T
11111000
01111100
00111110
00011111
10001111
11000111
11100011
11110001
a7 a6 a5 a4 a3 a2 a1
0 1 1 0 0 0 1 1
Key Expansion:
Round Constants
01 02 04 08...
First Column:
S
0 1 B K
M 2 I E
E 8 T Y
K
Round Key 0
K E Y
B3 6E CB B7
01 00 00 00
B2 6E CB B7
S O M E
B2 6E CB B7
E1 21 86 F2
Other Columns:
T 2 8
E1 21 86 F2
C1 10 B4 CA
Prev Col ⊕ Col from Previous round key
Mix Columns:
2113
2 1 1 3
3 2 1 1
1 3 2 1
1 1 3 2
a3 a2 a1 a0
Inverse Mix
EBD9
E B D 9
9 E B D
D 9 E B
B D 9 E
a3 a2 a1 a0

Block Cipher Modes

- How to encrypt variable-length messages with a block cipher?
- Naive approach:
  - ✓ Divide a message into blocks and encrypt each block.
- Padding: Encoding is up to the upper layer but must be <u>reversible</u>, e.g.
  - ✓ Add a single fixed byte (0x80) and pad the rest with 0x00, or
  - ✓ Determine number of padding bytes *n*, and pad with *n* bytes, each with value *n.*

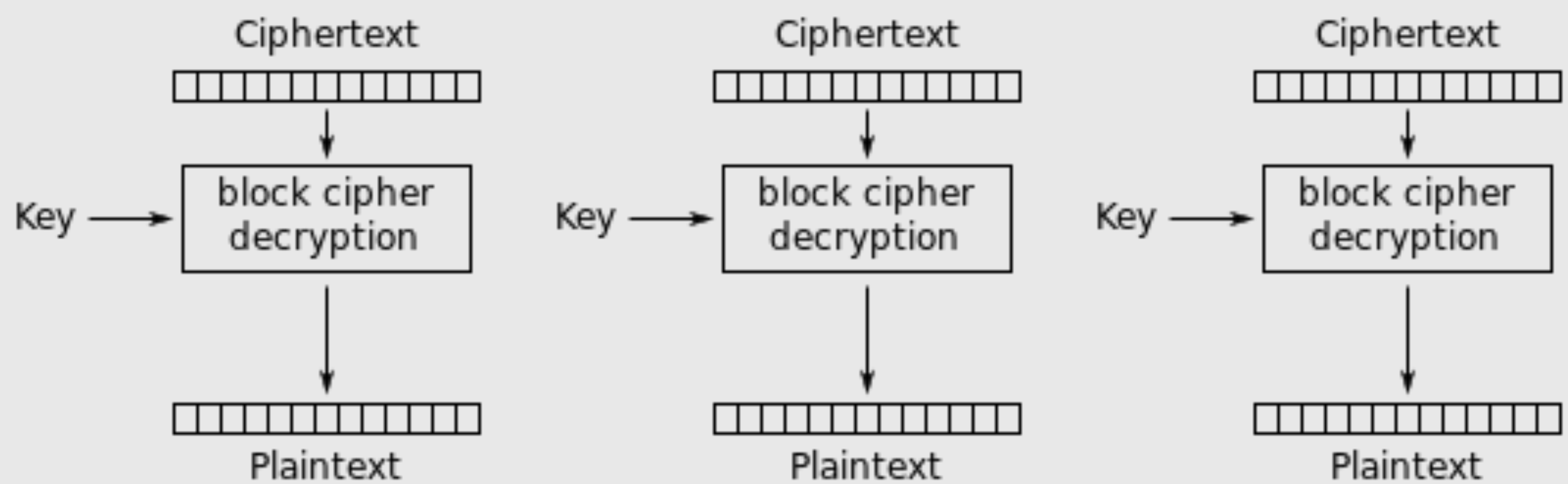
Electronic Codebook (ECB)

- $C_i = E_K(P_i)$



Electronic Codebook (ECB) mode encryption

- $P_i = D_K(C_i)$



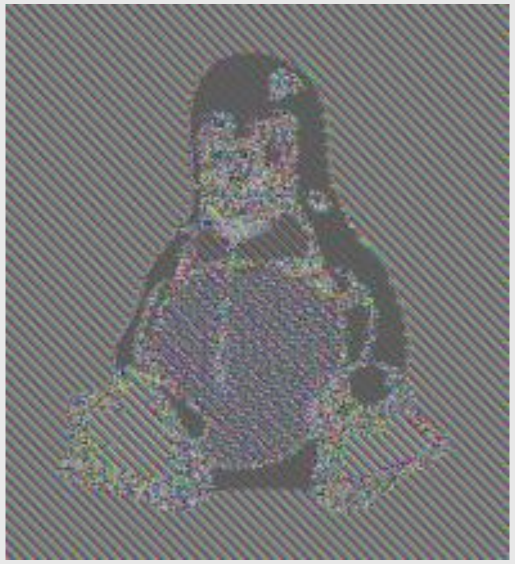
Electronic Codebook (ECB) mode decryption

ECB Properties

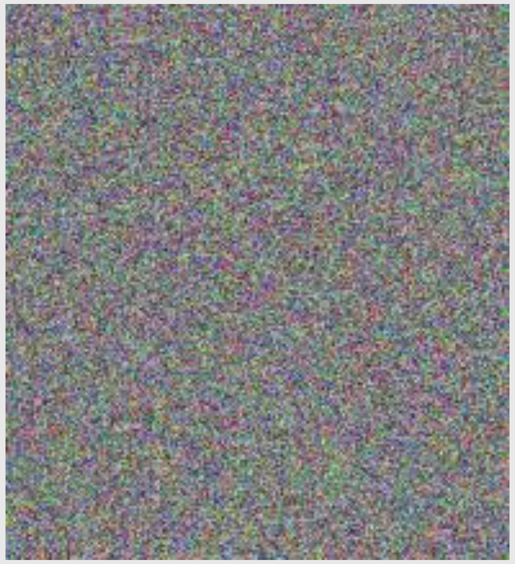
- Simple !
- Encryption/Decryption can be done in parallel.
- Padding is needed.
- Identical plaintext blocks are encrypted into identical ciphertext block.
  - ✓ if $P_i = P_j$ then $C_i = C_j$

Plaintext

ECB's ciphertext

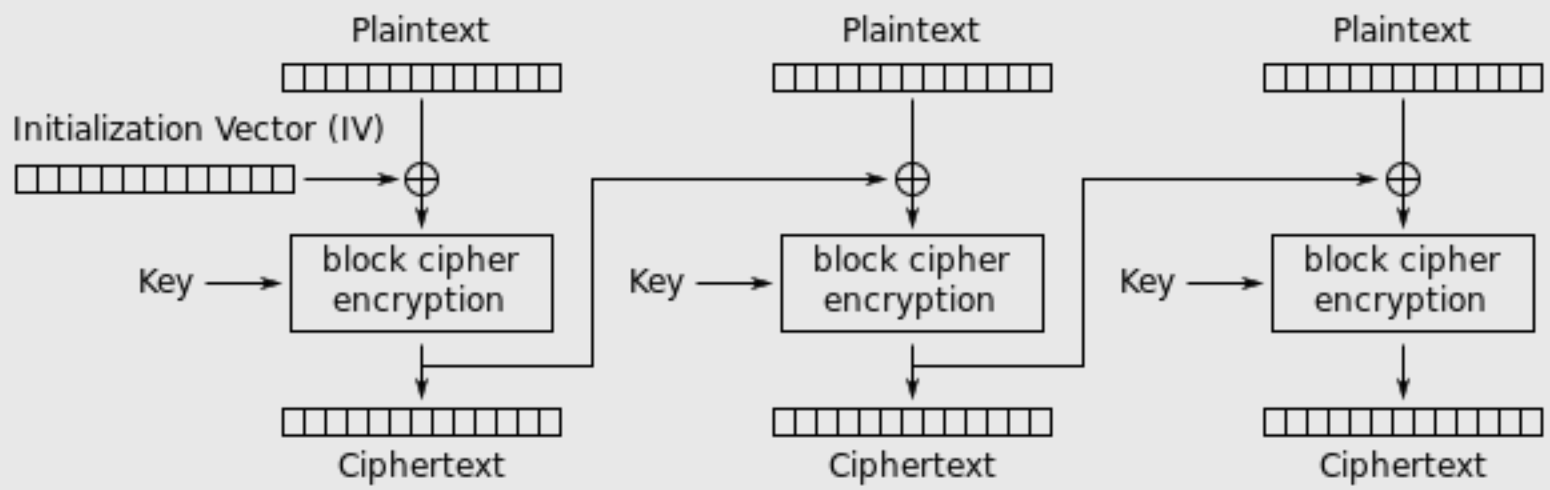


Expected



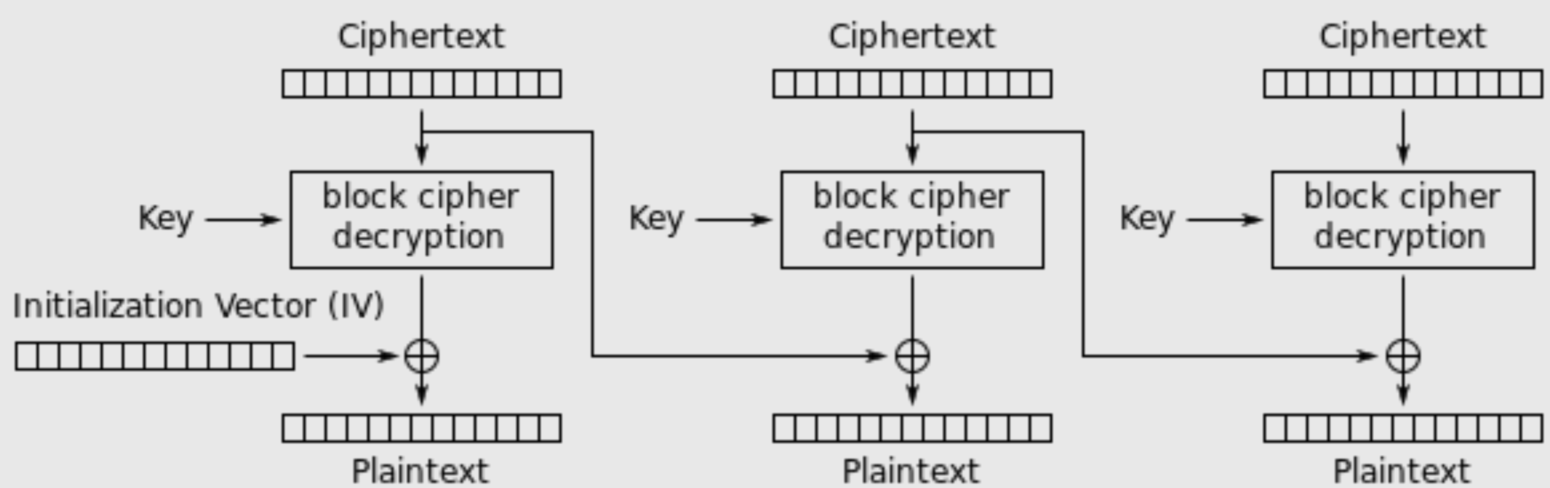
Cipher Block Chaining (CBC)

- $C_i = E_K(P_i \oplus C_{i-1})$
  $C_0 = IV$



Cipher Block Chaining (CBC) mode encryption

- $P_i = D_K(C_i) \oplus C_{i-1}$
  $C_0 = IV$



Cipher Block Chaining (CBC) mode decryption

CBC Properties

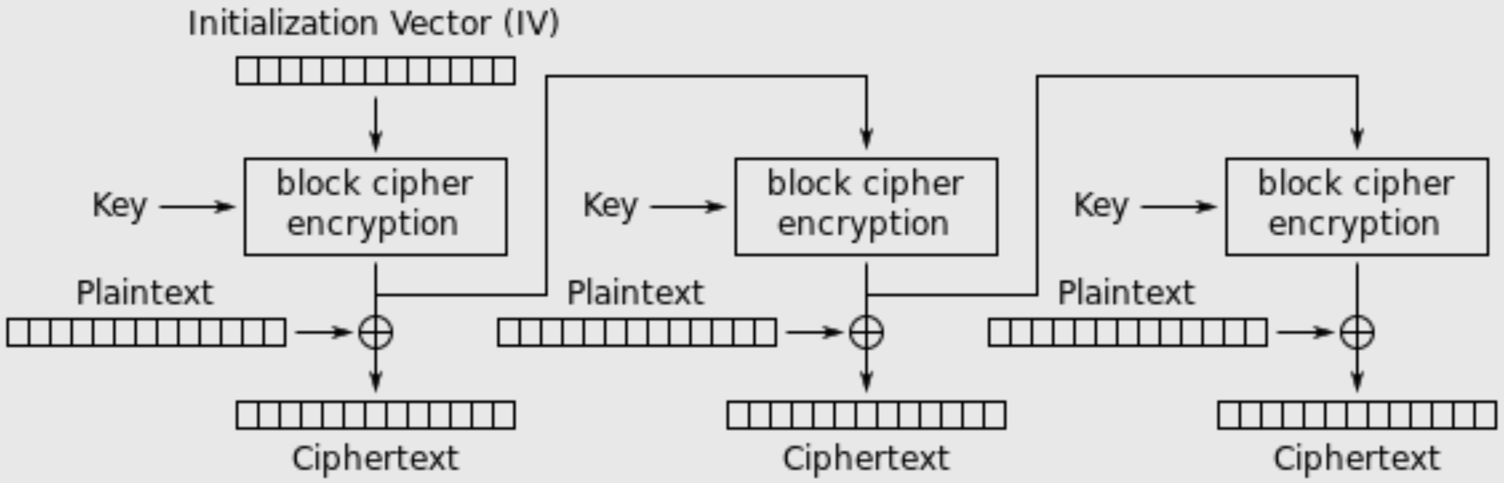
- Eliminates the problems of ECB.
  - ✓ Involves initialization vector (IV) to randomize inputs.
  - ✓ IV's length is the block size.
- Sequential encryption, **but decryption can be parallelized**.
- A receiver needs to know IV.
- If $C_i = C_j$ then $P_i \oplus P_j = C_{i-1} \oplus C_{j-1}$

Initialization Vector (IV)

- Fixed IV
  - ✓ CBC with a fixed IV has similar properties as ECB.
- Counter IV
  - ✓ $IV_{i+1} = IV_i$
  - ✓ Can reveal information about the plaintext (e.g., when the first plaintext blocks have small differences).
- **Random IV**
  - ✓ A random IV is generated for every message and sent with the ciphertext.
  - ✓ Increases communication overhead.
- **Nonce-generated IV**
  - ✓ Nonce (number used **once**) is used to generate an IV, e.g. $IV = E_K(nonce)$.
  - ✓ Nonce could be a message number (or any other unique number).
  - ✓ It can help to minimize the communication overhead of random IV.

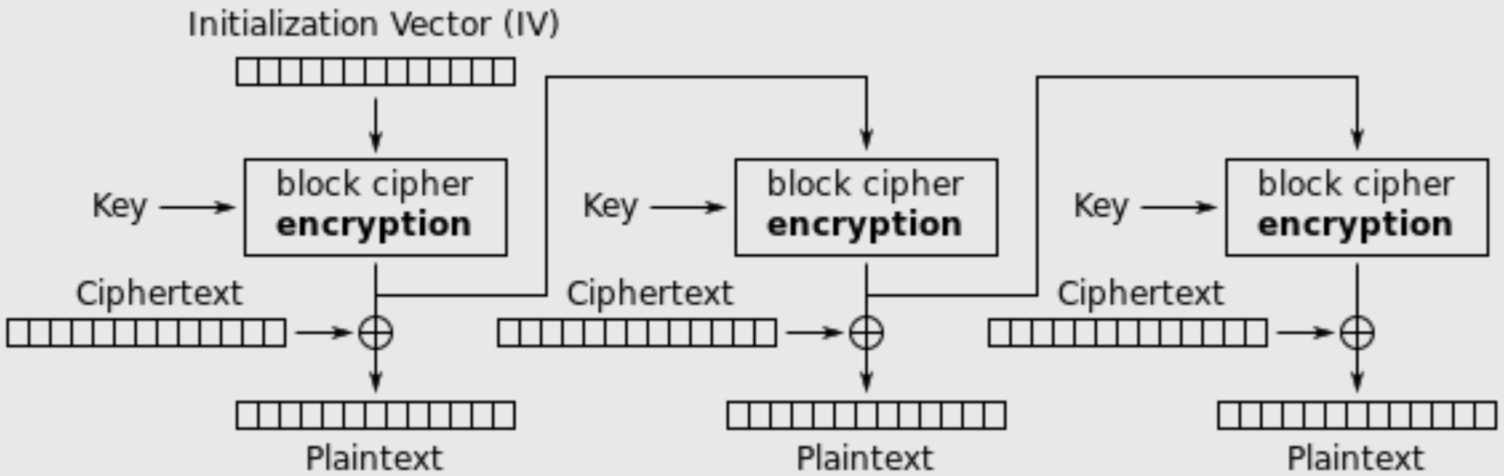
Output Feedback (OFB)

- $C_i = P_i \oplus T_i$
  $T_i = E_K(T_{i-1})$
  $T_0 = IV$



Output Feedback (OFB) mode encryption

- $P_i = C_i \oplus T_i$
  $T_i = E_K(T_{i-1})$
  $T_0 = IV$



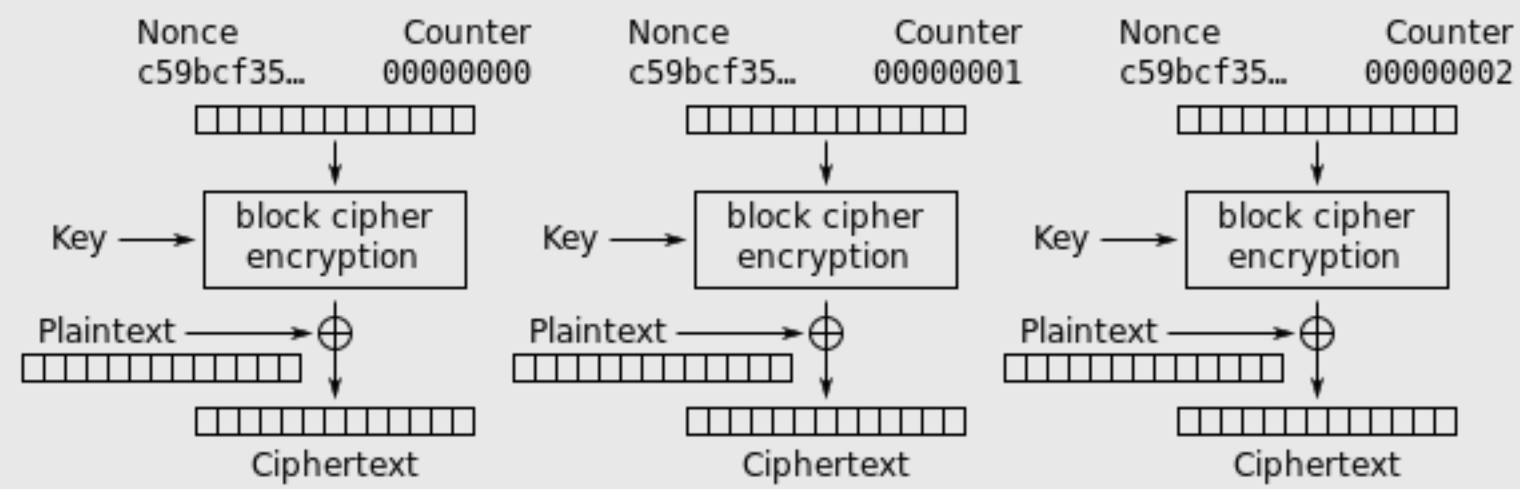
Output Feedback (OFB) mode decryption

OFB Properties

- Eliminates the problems of ECB by producing a **key stream.**
  - ✓ Involves IV to randomize key stream.
- Sequential encryption/decryption (cannot be parallelized).
- Only encryption operation is used (no decryption operation).
- **No padding is needed.**
- A receiver needs to know IV (as in CBC).
- Reused IV is very dangerous.
  - ✓ If IV = IV', then $T_i = T_i'$, $C_i \oplus C_i' = (P_i \oplus T_i) \oplus (P_i' \oplus T_i) = P_i \oplus P_i'$
  - ✓ If C_i, C_i', P_i are known, it is trivial to find P_i'.
- Cycles in key streams are possible (although, not very likely).
  - ✓ Suppose block size is 128 bits.
  - ✓ After how many blocks of encryption, such a collusion may happen?

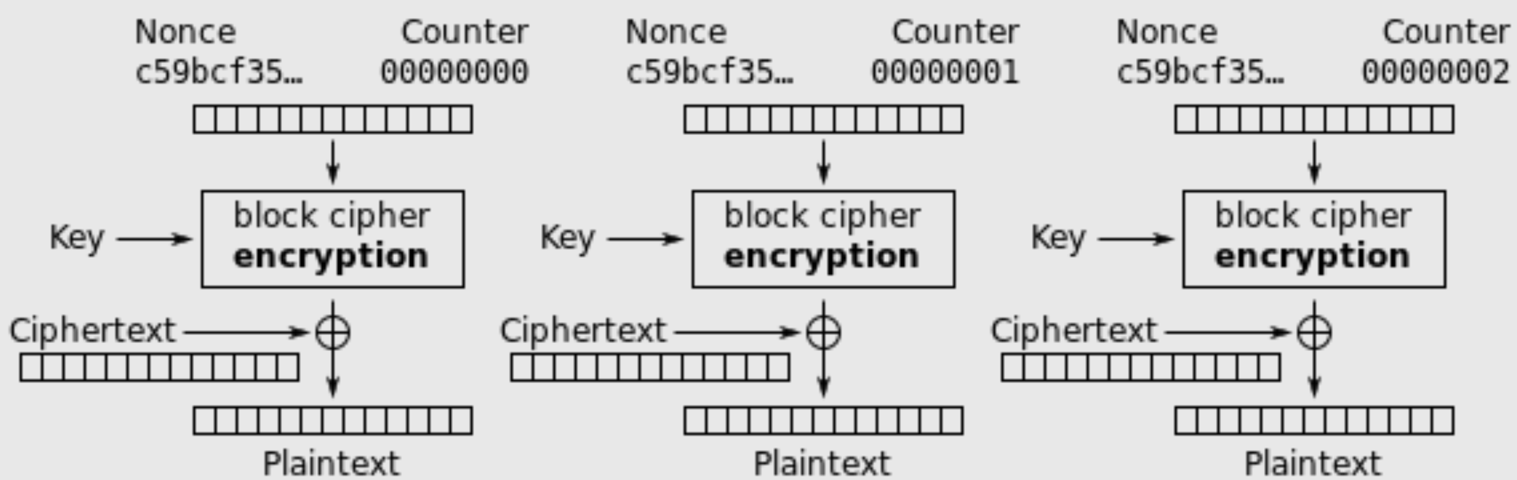
Counter (CTR)

- $C_i = E_K(Nonce \,||\, i) \oplus P_i$



Counter (CTR) mode encryption

- $P_i = E_K(Nonce \,||\, i) \oplus C_i$



Counter (CTR) mode decryption

CTR Properties

- Eliminates the problems of ECB by producing a **key stream.**
  - ✓ Involves Nonce and counter to randomize key stream.
  - ✓ Typical setting for 128-bit block: 64-bit *Nonce* + 64-bit counter *i*.
- Encryption/decryption can be parallelized (as in ECB).
- Only encryption operation is used (as in OFB).
- No padding is needed (as in OFB).
- A receiver needs to know Nonce.
- Reused (Nonce, counter) pair is very dangerous.

Other Issues

- Usually CBC or CTR mode is used.
  - ✓ ECB is not secure.
  - ✓ CTR is better than OFB.
- CBC, CTR, and OFB provide CPA security, is it enough?
  - ✓ It guarantees that Eve will not learn anything about plaintexts (except their lengths).
  - ✓ What else can go wrong?
    - Let's assume that Eve can manipulate communication... → authentication.
  - ✓ All the modes presented are **not** CCA-secure.
- Limit the amount of data to be encrypted by one key.

Key Points

- Kerckhoffs' principle
- Types of attacks to symmetric ciphers
- Security level of a cipher
- Block cipher & AES
- Padding in block cipher
- Block cipher modes (ECB, CBC, OFB, CTR)

Exercises & Reading

- Classwork (Exercise Sheet 5): due on Fri Oct 12, 10:00 PM
- Homework (Exercise Sheet 5): due on Fri Oct 26, 6:59 PM
- Reading: FSK [Ch2, Ch3, Ch4]
- Mid-term exam (Week 6): Fri 19 Oct, 7:30 PM (covering Part I Foundations: Week 1 – Week 4)

End of Slides for Week 5