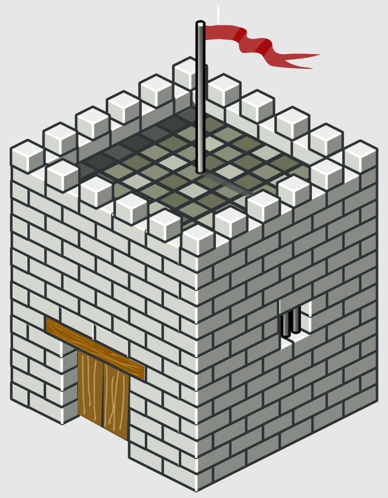
Foundations of Cybersecurity

I - Symmetric Encryption



Paweł Szałachowski

2017

About me

- ISTD's faculty member since Aug 2017
- Research and systems building
  - SCION Internet Architecture: https://www.scion-architecture.net/
  - Public-key Infrastructures, SSL/TLS, Internet security...
  - Blockchain and FinTech
- https://pszal.github.io ; pawel@sutd.edu.sg ; 1.402-35

Cryptography

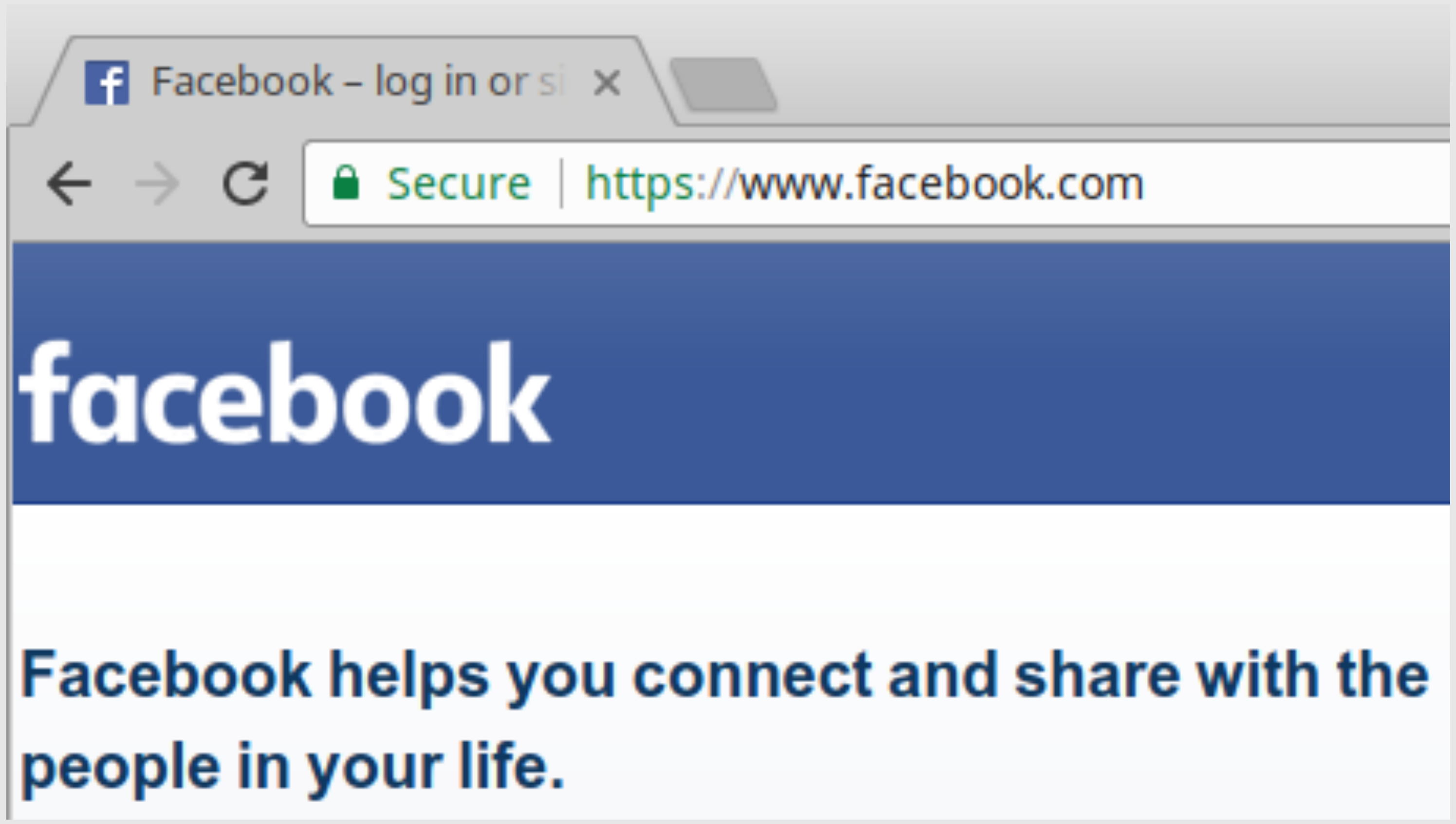
- Cryptography: art and science of encryption (ciphers)
- More than encryption (other primitives)
    - hash functions, MACs, (P)RNG, RSA, DH
- Higher-level constructions
    - secure channel, key server, PKI
- Real-world systems

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 and without well-defined boundaries
- Cryptography is not the solution
  - storing password vs. their management

How it happens?

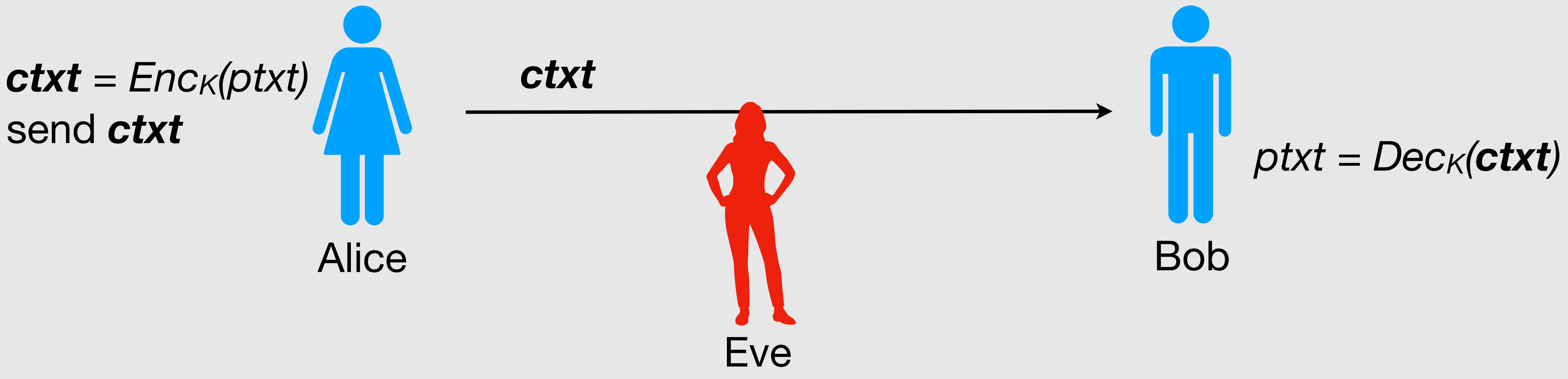
- Secure communication
    - Client-Server via HTTP**S**



Facebook helps you connect and share with the people in your life.

Symmetric Encryption

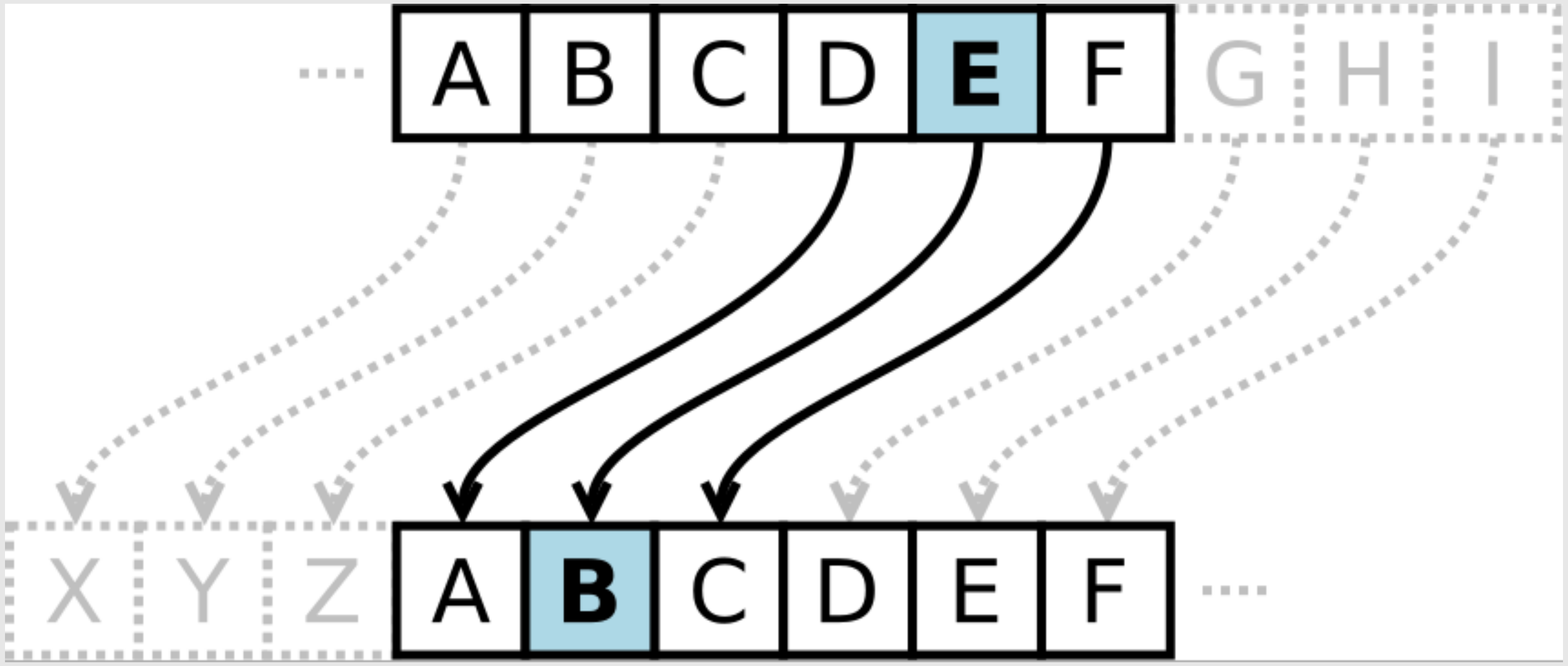
- Encryption scheme
  - encryption and decryption algorithms *Enc* and *Dec*
- Alice, Bob, and Eve
  - Alice and Bob share a secret (symmetric) key
  - Eve sees all (encrypted) communication



- Kerckhoffs' Principle
  - *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



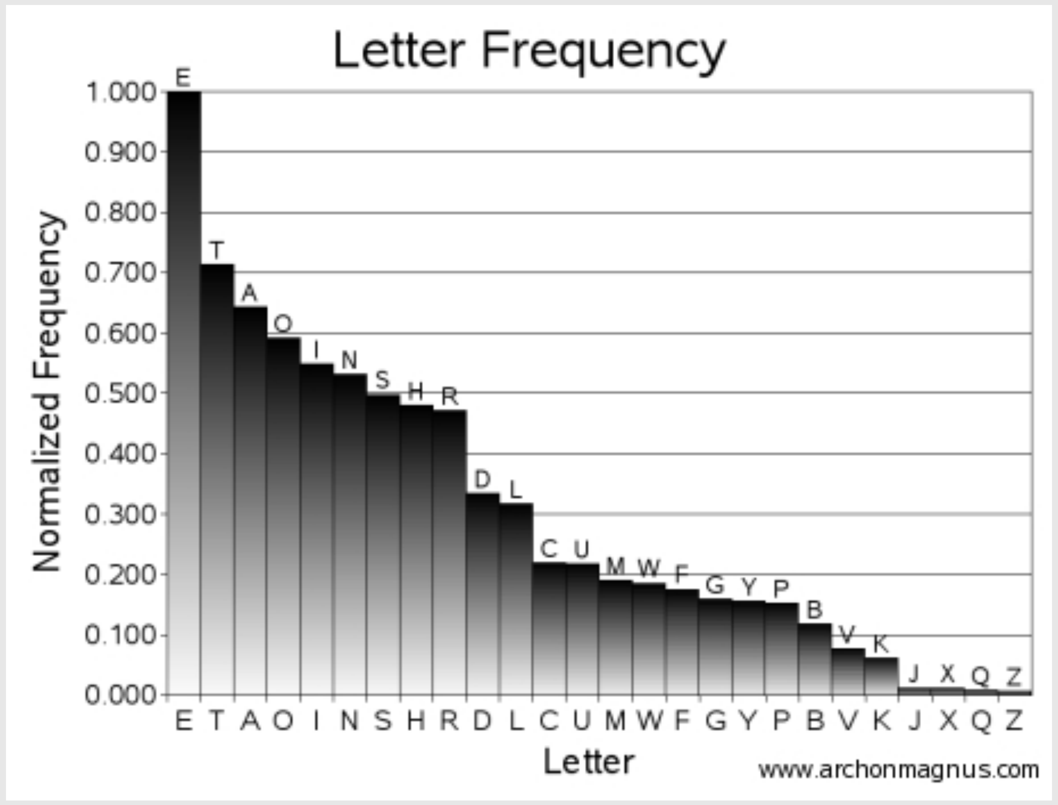
Other Substitution Ciphers

- Monoalphabetic

  Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ

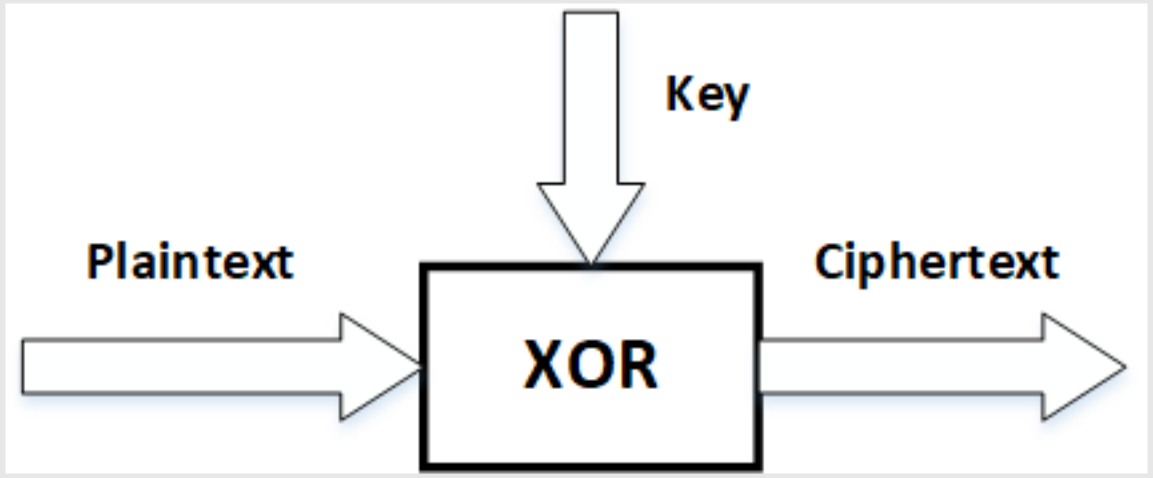
  Key: PDKIFMRBHSONCGXUTJWEYLQAZV

- Trivial to break with letter frequency



One-Time Pad

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



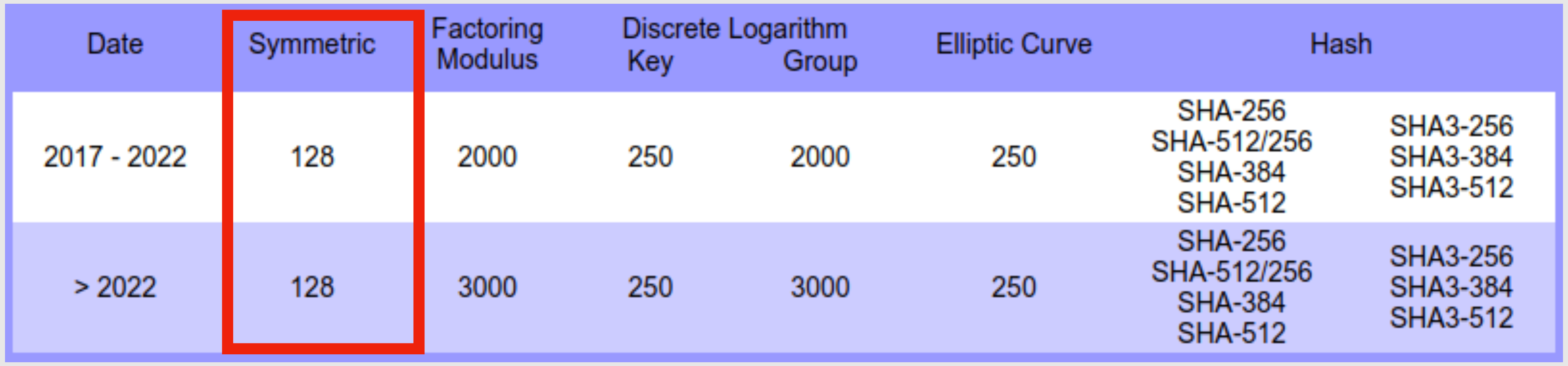
- This scheme **cannot** be *broken*
    - No matter how strong an adversary is, she cannot learn anything about plaintext
- Disadvantages?

Attacks

- The Ciphertext-Only Attack (COA)
  - Eve knows only ciphertexts (without the corresponding plaintexts)
- The Known-Plaintext Attack (KPA)
  - Eve knows some (plaintext, ciphertext) pairs
- The **Chosen-Plaintext Attack** (CPA)
  - Eve can select plaintexts and obtain the corresponding ciphertexts
- The Chosen-Ciphertext Attack (CCA)
  - Eve can select plaintexts and/or ciphertexts and obtain the corresponding ciphertexts and/or plaintexts
- Eve should not be able to *distinguish* between the ideal encryption scheme and the actual one.

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		Elliptic Curve	Hash	
			Key	Group			
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

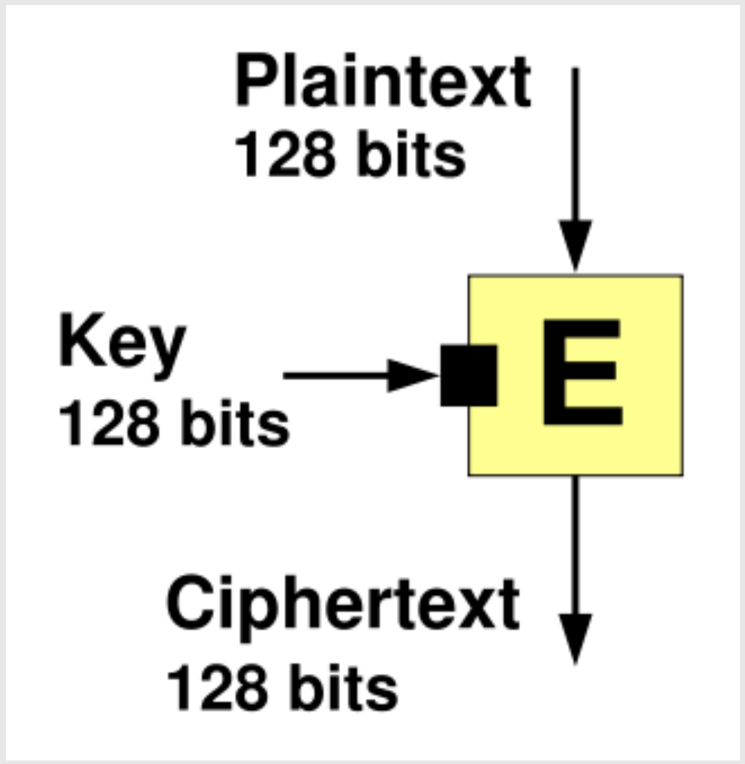
- The level of security is usually a function of the access of the adversary (how many encrypted messages she sees, can she conduct a CCA attacks, etc...)

Modern Ciphers

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

Block Cipher

- An encryption 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.



Security

- For each key the *ideal block cipher* is a random permutation, and the different permutations for the different key values should be chosen independently.

- An attack on a block cipher is a non-generic method of distinguishing the block cipher from an ideal block cipher.

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

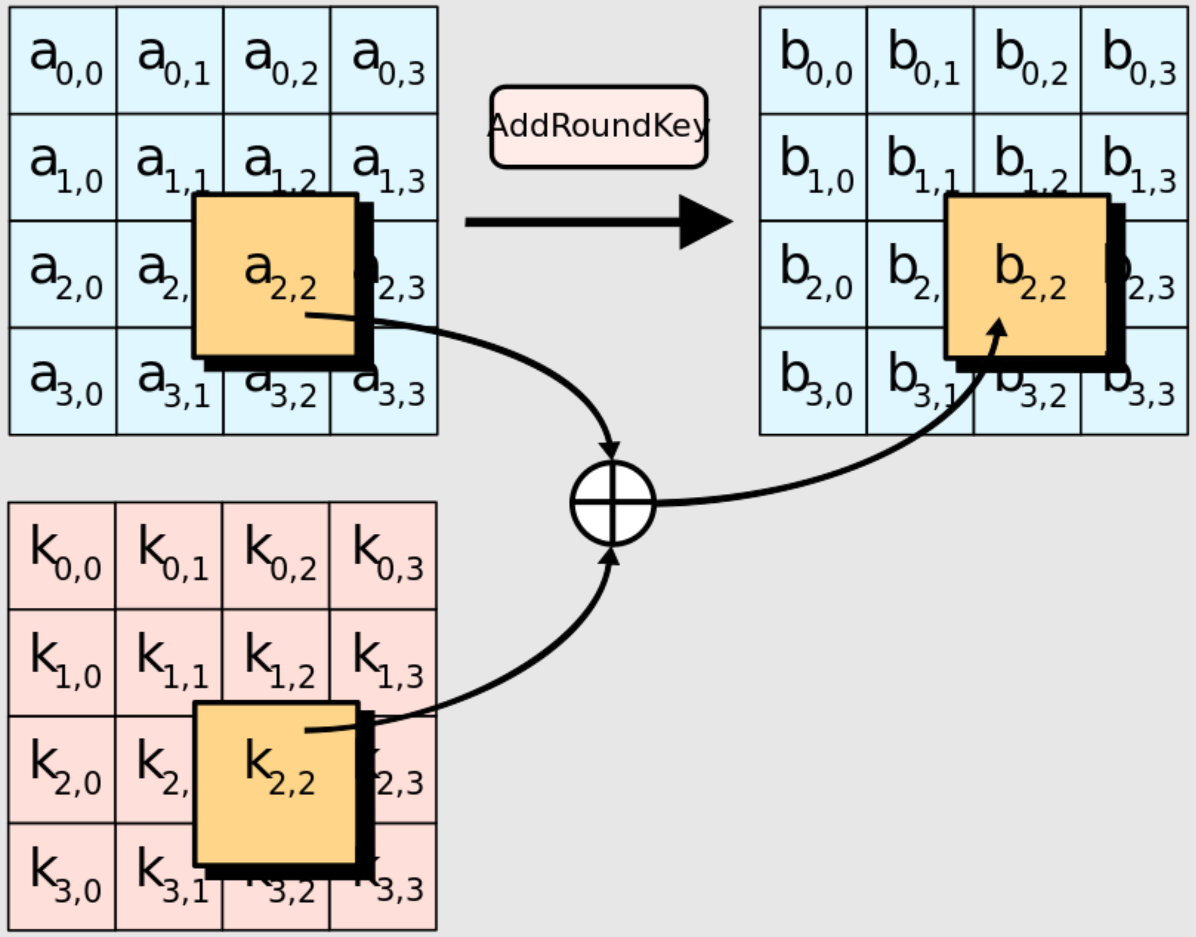
AES Internals

```
AddRoundKey(0)

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

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

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

AES Internals

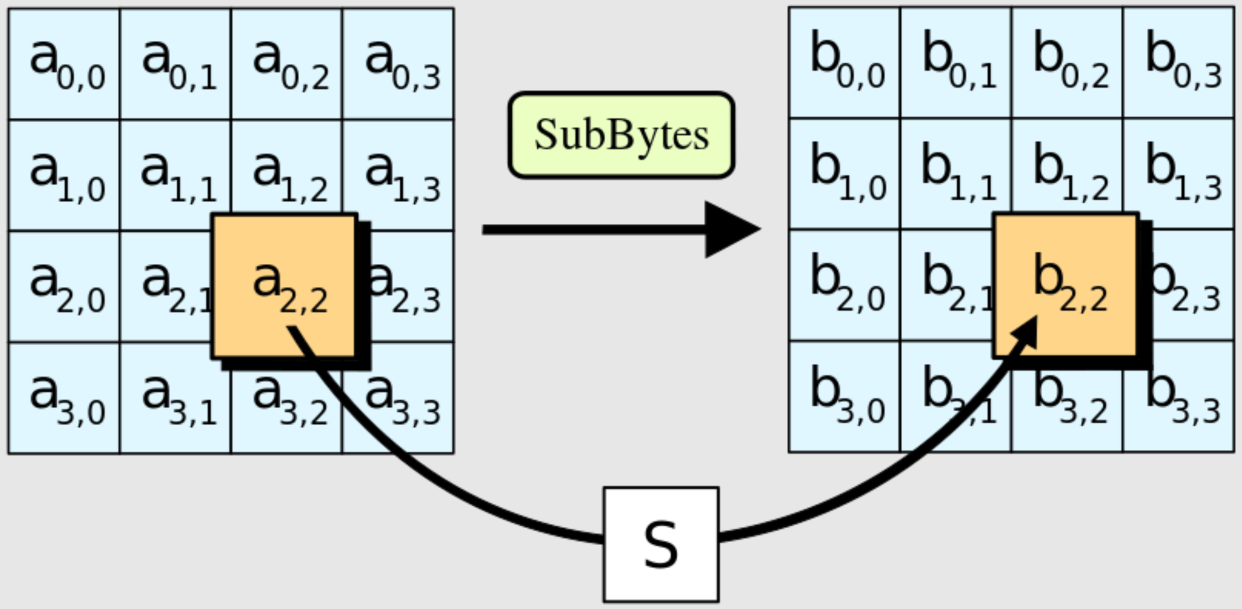
```
**AddRoundKey**(0)

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

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



Each byte of the state is combined (XORed) 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)
```



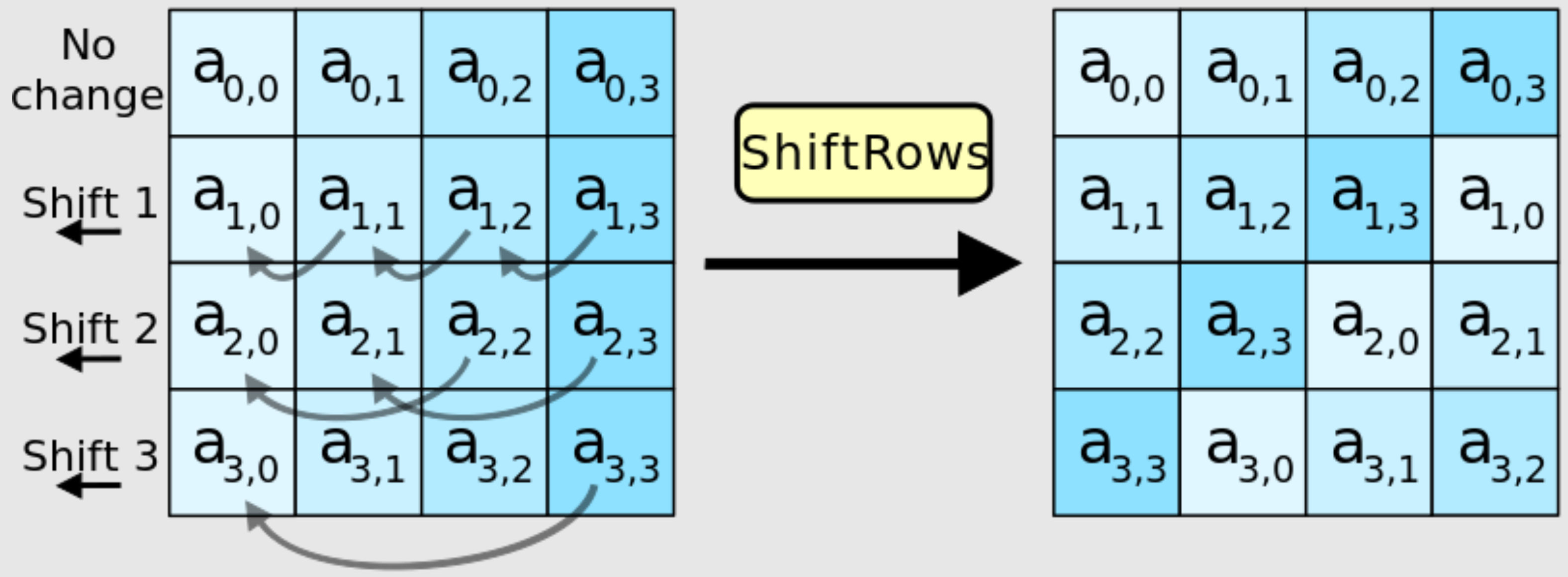
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
SubBytes
S
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

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)
```

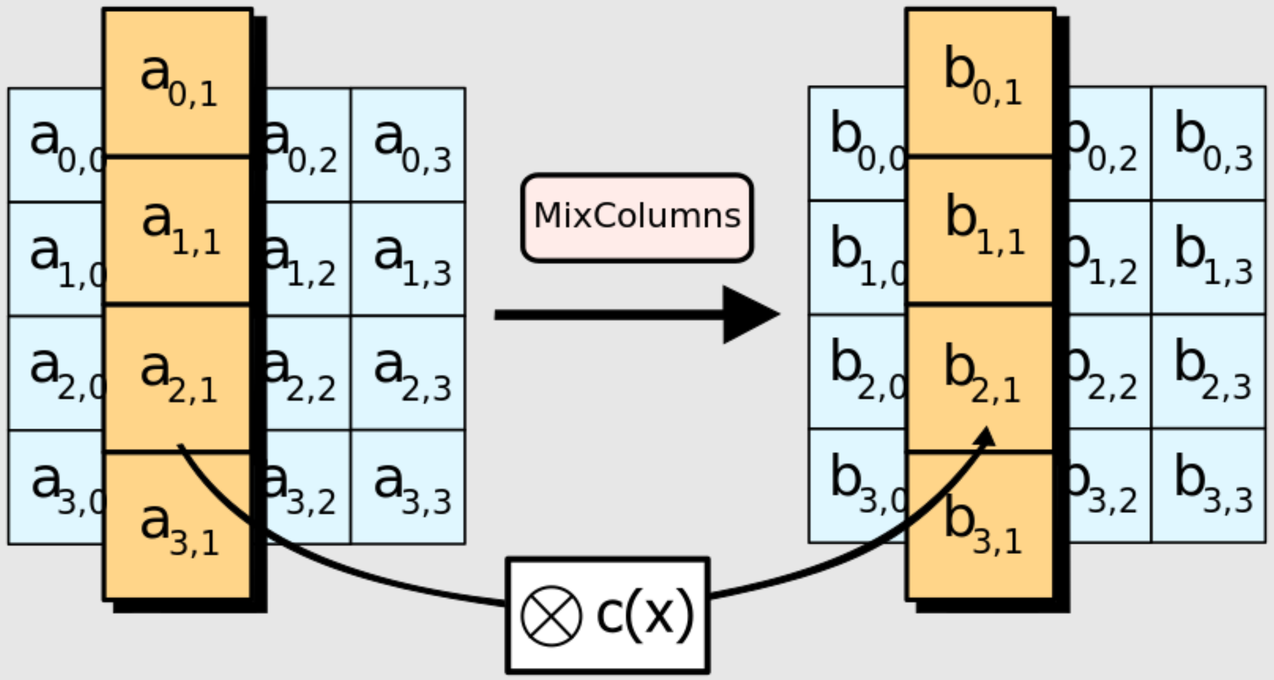
Bytes in each row of the state are shifted cyclically to the left.
The goal is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.



No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

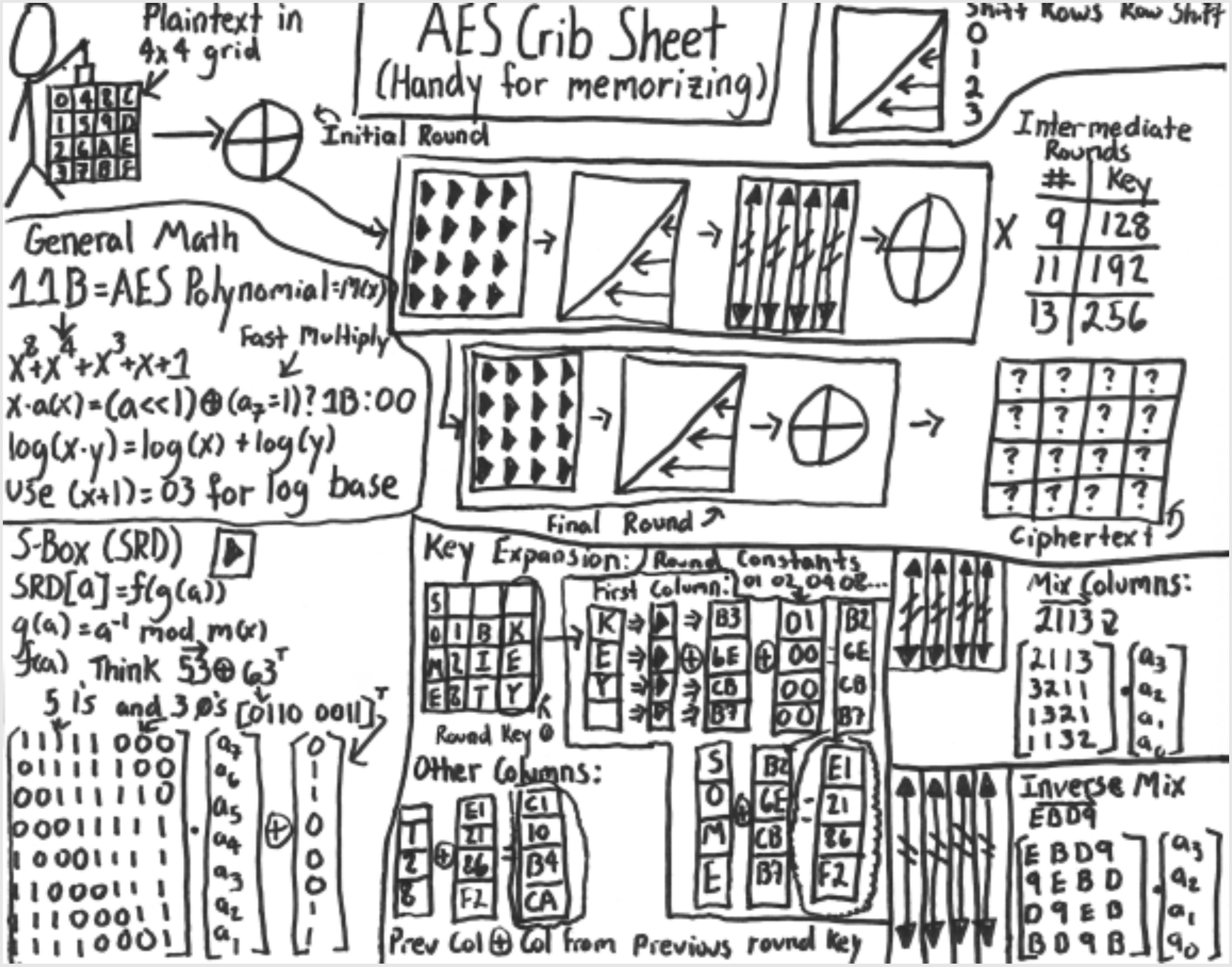
AES Internals

```
AddRoundKey(0)

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

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



MixColumns
$\otimes c(x)$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

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



Intermediate Rounds

#	Key
9	128
11	192
13	256

General Math

11B = AES Polynomial = m(x)

$x^8 + x^4 + x^3 + x + 1$

Fast Multiply

$x \cdot a(x) = (a \ll 1) \oplus (a_7 = 1)\ ?\ 1B : 00$

$\log(x \cdot y) = \log(x) + \log(y)$

Use $(x+1) = 03$ for log base

S-Box (SRD)

$SRD[a] = f(g(a))$

$g(a) = a^{-1} \bmod m(x)$

$f(a)$ Think $\overrightarrow{5}3 \oplus 63^T$

5 1's and 3 0's $[0110\ 0011]^T$

$$\begin{bmatrix} 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \end{bmatrix} \cdot \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Key Expansion:

S			
O	1	B	K
M	2	I	E
E	8	T	Y

$$\begin{bmatrix} K \\ E \\ Y \\ \ \end{bmatrix} \Rightarrow \Rightarrow \begin{bmatrix} B3 \\ 6E \\ CB \\ B7 \end{bmatrix} \oplus \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix} = \begin{bmatrix} B2 \\ 6E \\ CB \\ B7 \end{bmatrix}$$

$$\begin{bmatrix} S \\ O \\ M \\ E \end{bmatrix} \oplus \begin{bmatrix} B2 \\ 6E \\ CB \\ B7 \end{bmatrix} = \begin{bmatrix} E1 \\ 21 \\ 86 \\ F2 \end{bmatrix}$$

$$\begin{bmatrix} T \\ 2 \\ 8 \end{bmatrix} \oplus \begin{bmatrix} E1 \\ 21 \\ 86 \\ F2 \end{bmatrix} = \begin{bmatrix} C1 \\ 10 \\ B4 \\ CA \end{bmatrix}$$

Mix Columns:

$$\begin{bmatrix} 2&1&1&3 \\ 3&2&1&1 \\ 1&3&2&1 \\ 1&1&3&2 \end{bmatrix} \cdot \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

Inverse Mix

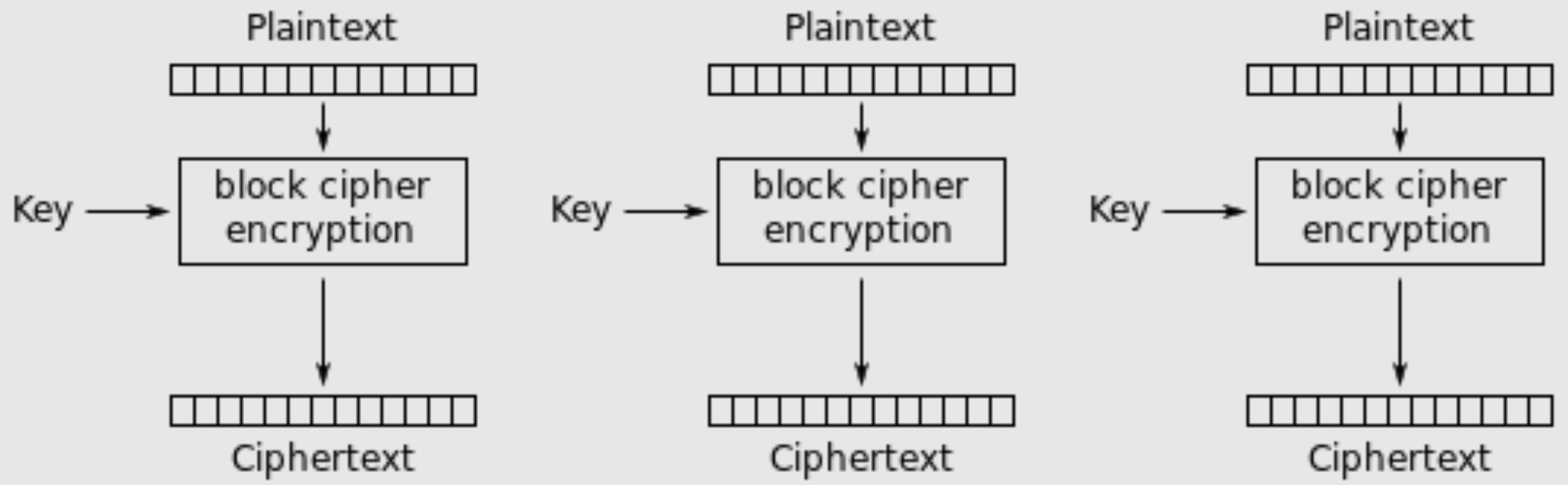
$$\begin{bmatrix} E&B&D&9 \\ 9&E&B&D \\ D&9&E&B \\ B&D&9&E \end{bmatrix} \cdot \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix}$$

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 reversible, 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*

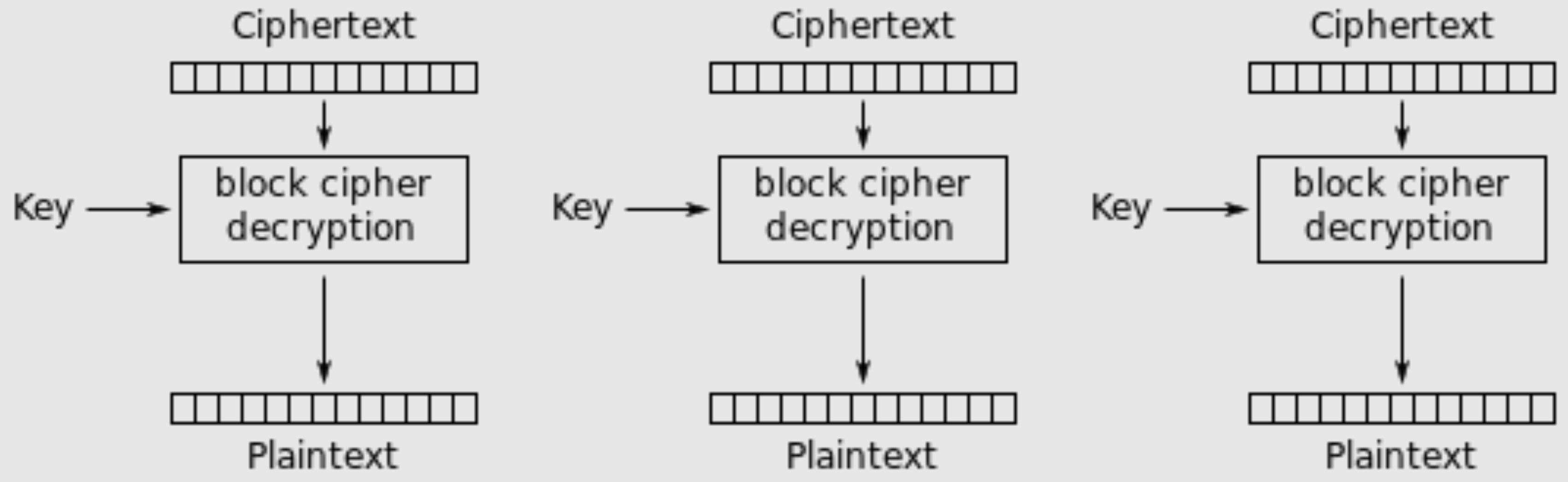
Electronic Codebook (ECB)

$C_i = E_K(P_i)$



Electronic Codebook (ECB) mode encryption

$P_i = D_K(C_i)$

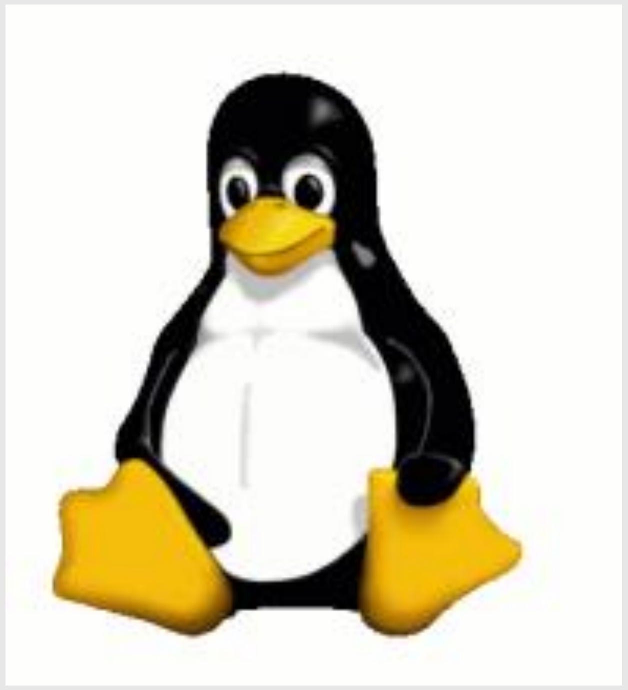


Electronic Codebook (ECB) mode decryption

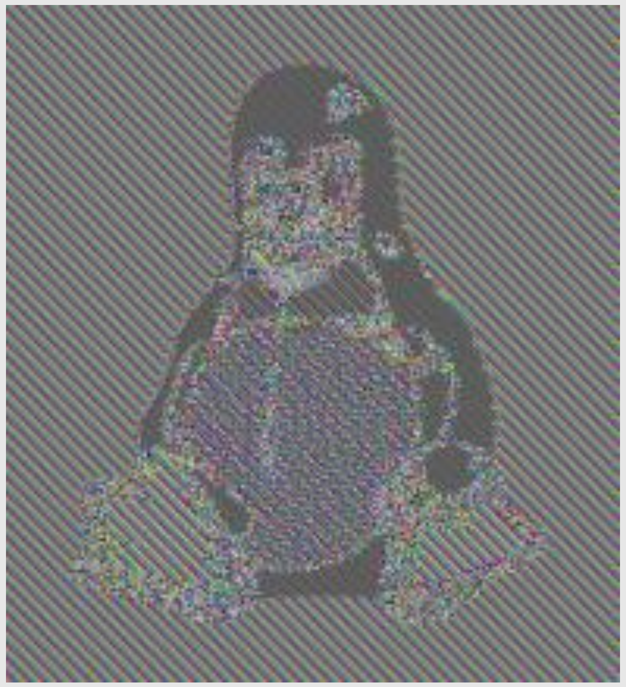
ECB properties

- 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 $C_i = C_j$

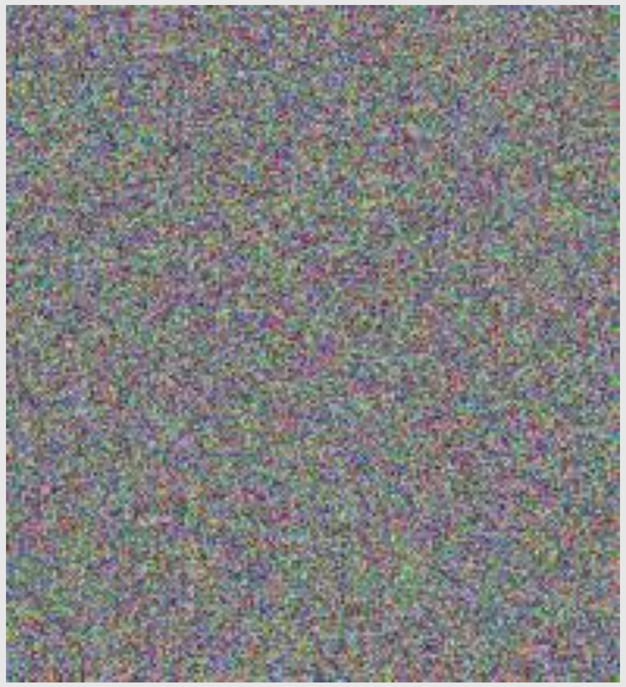
Plaintext



ECB's ciphertext

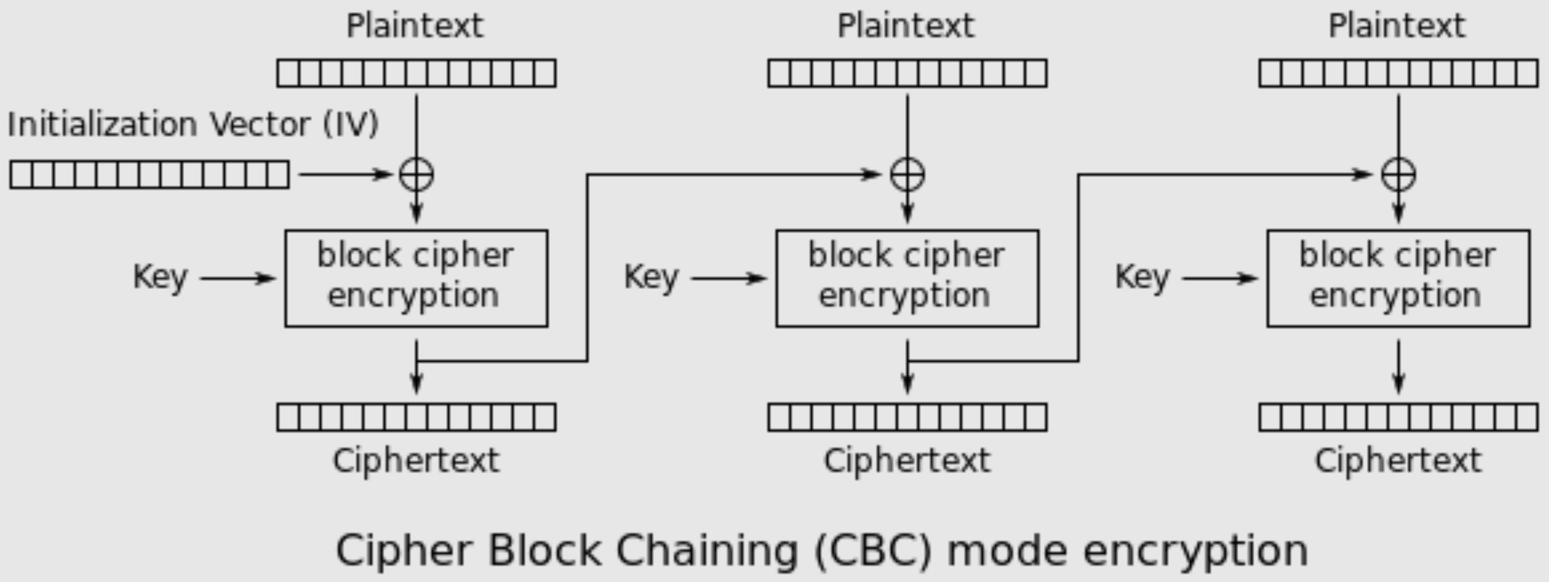


Expected



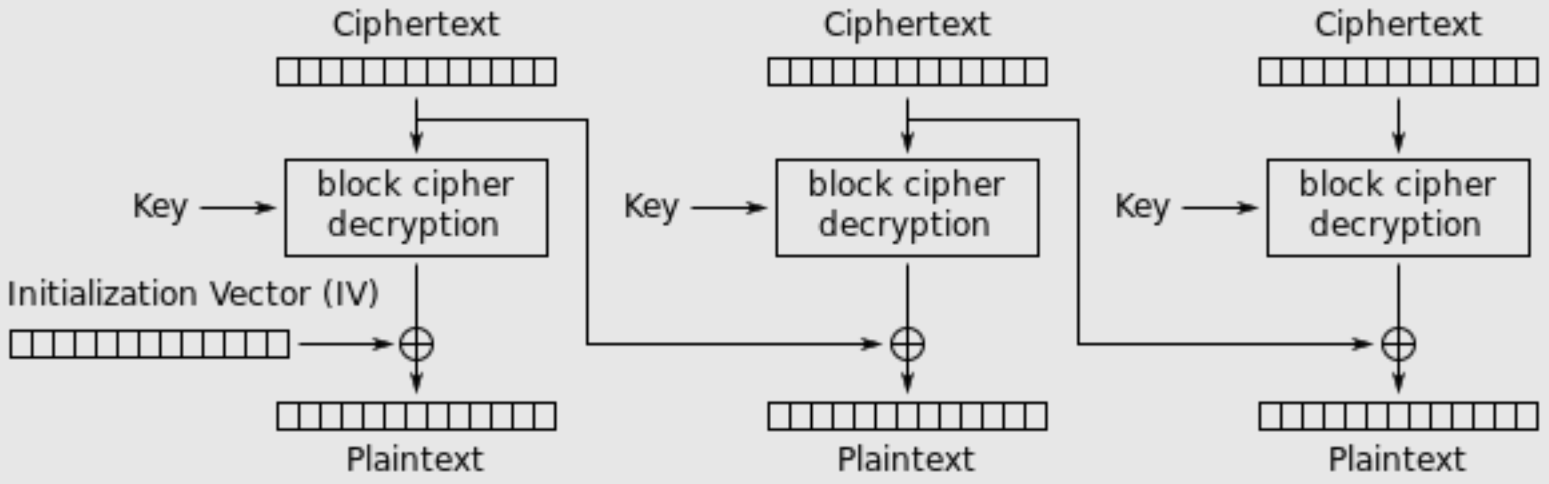
Cipher Block Chaining (CBC)

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



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

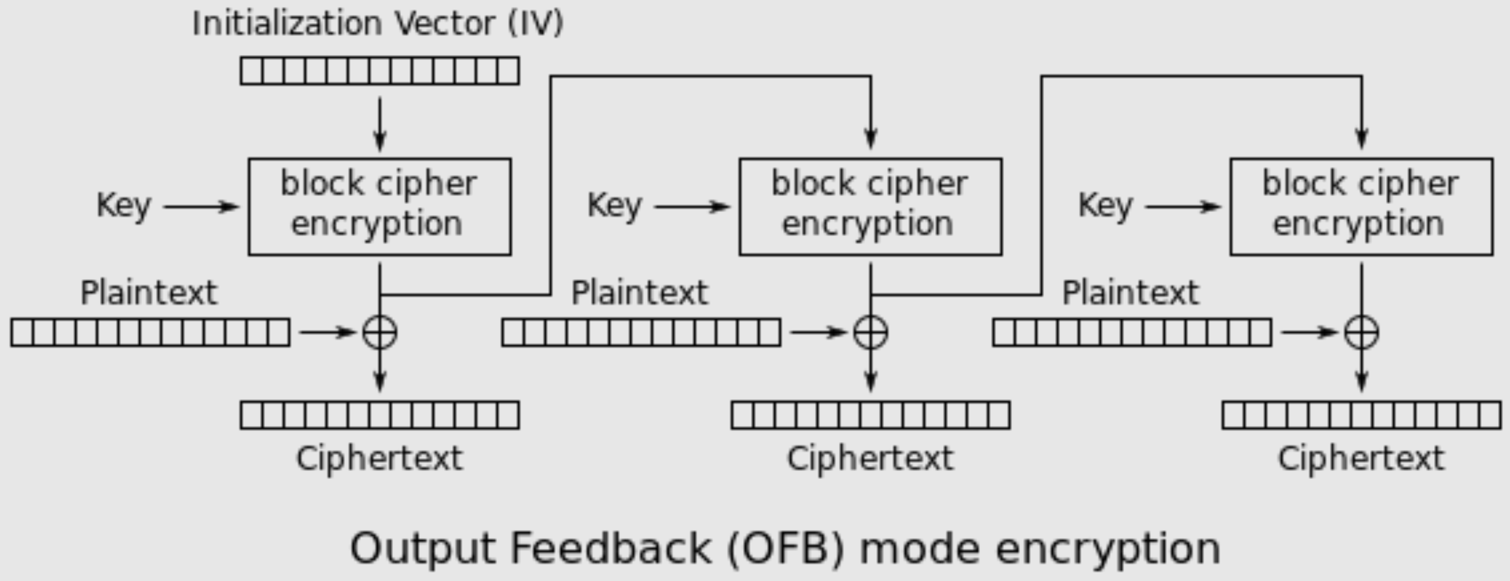
- Eliminates the problems of ECB
    - involves initialization vector (IV) to randomize inputs
        - IV's length is the block size
- Sequential encryption/decryption (cannot be parallelized)
- A receiver needs to know IV
- If $C_i = C_j$ then $P_i \oplus P_j = C_{i-1} \oplus C_{j-1}$ (analogical for $C_i \neq C_j$)

Initialization Vector (IV)

- Fixed IV
  - CBC with a fixed IV has similar properties as ECB
- Counter IV
  - 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 messages and sent with the ciphertext
  - increases communication overhead
- **Nonce-Generated IV**
  - number used **once** (nonce) 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

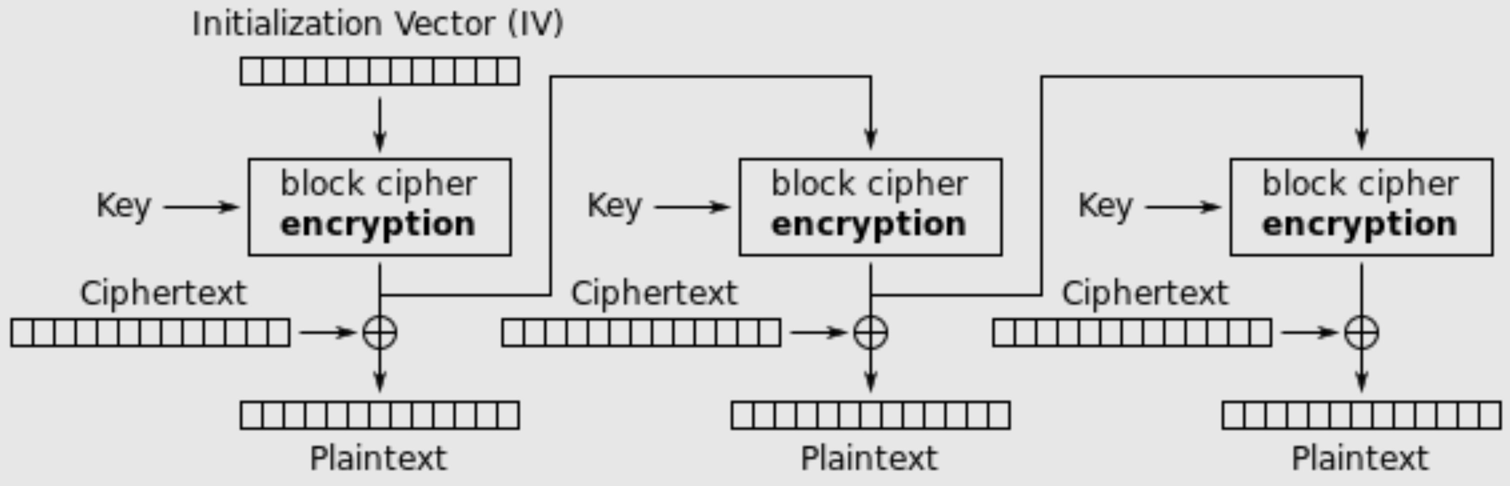
Output feedback (OFB)

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



Output Feedback (OFB) mode encryption

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



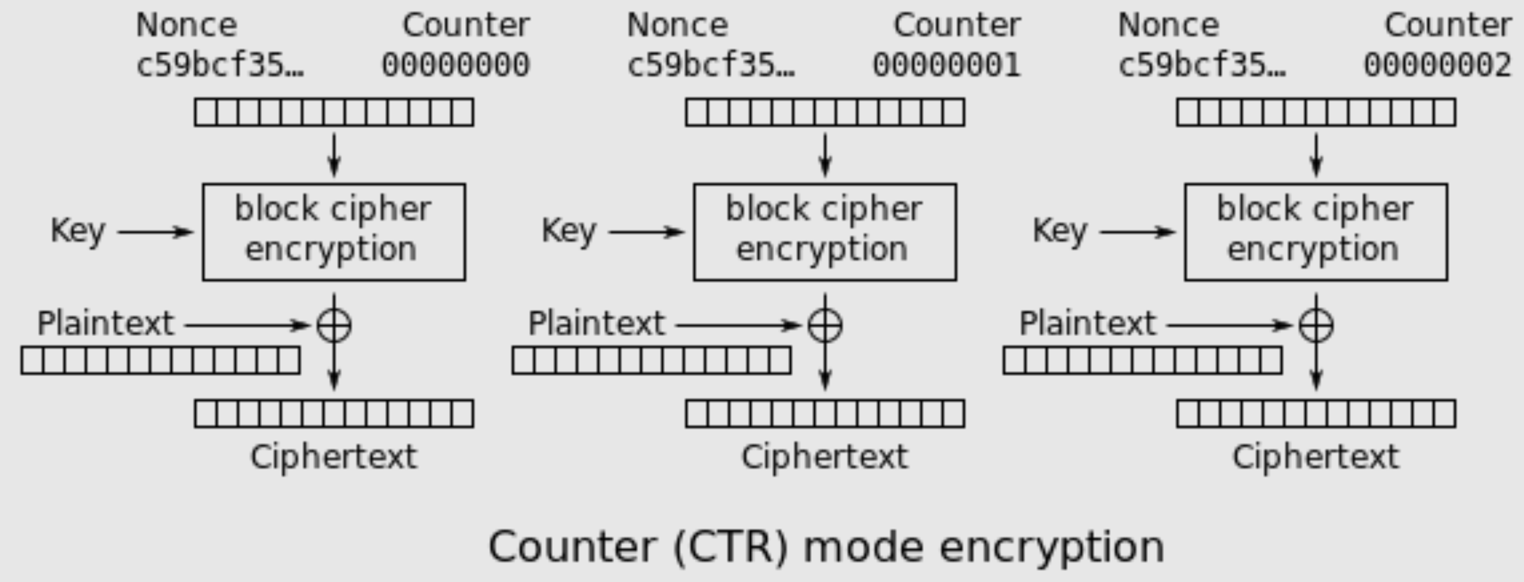
Output Feedback (OFB) mode decryption

Properties of OFB

- 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 padding is needed
- A receiver needs to know IV (as in CBC)
- Reused IV is very dangerous
- Cycles in key streams are possible (although, not very likely)

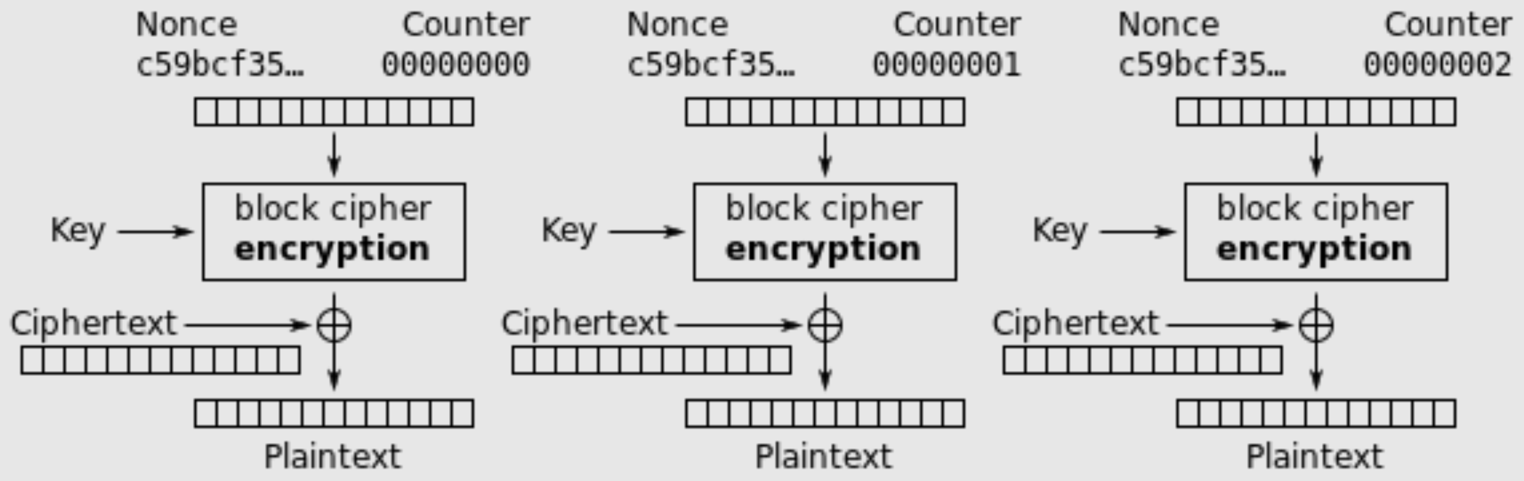
Counter (CTR)

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



Counter (CTR) mode encryption

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



Counter (CTR) mode decryption

Properties of CTR

- Eliminates the problems of ECB by producing a **key stream**
  - involves Nonce and counter to randomize key stream
- Encryption/decryption can be parallelized
- Only encryption operation is used
- No padding is needed
- A receiver needs to know Nonce
- Reused (Nonce, counter) pair is very dangerous

Other Issues

- Usually CBC or CTR mode is used
- 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...
    - All the modes presented are **not** CCA-secure.

Discussion & Classwork