

Introduction into Cyber Security

Chapter 2: Symmetric Encryption

WiSe 18/19

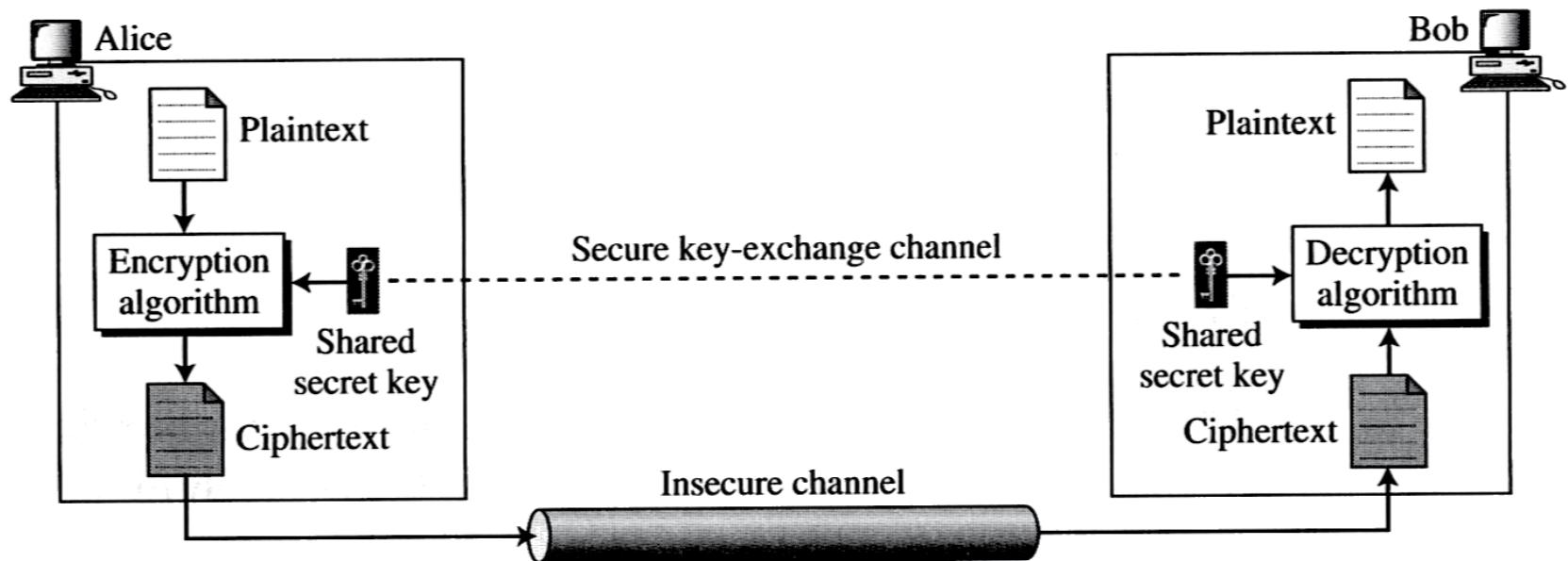
Chair of IT Security

Chapter Overview

- General Idea of Symmetric Encryption
- Block ciphers
- Modes to use block ciphers
- Stream ciphers
- (Pseudo) Random Number Generators

General Idea of Symmetric Encryption

- The two communication endpoints share a secret key
- The secret key is used for both encryption and decryption



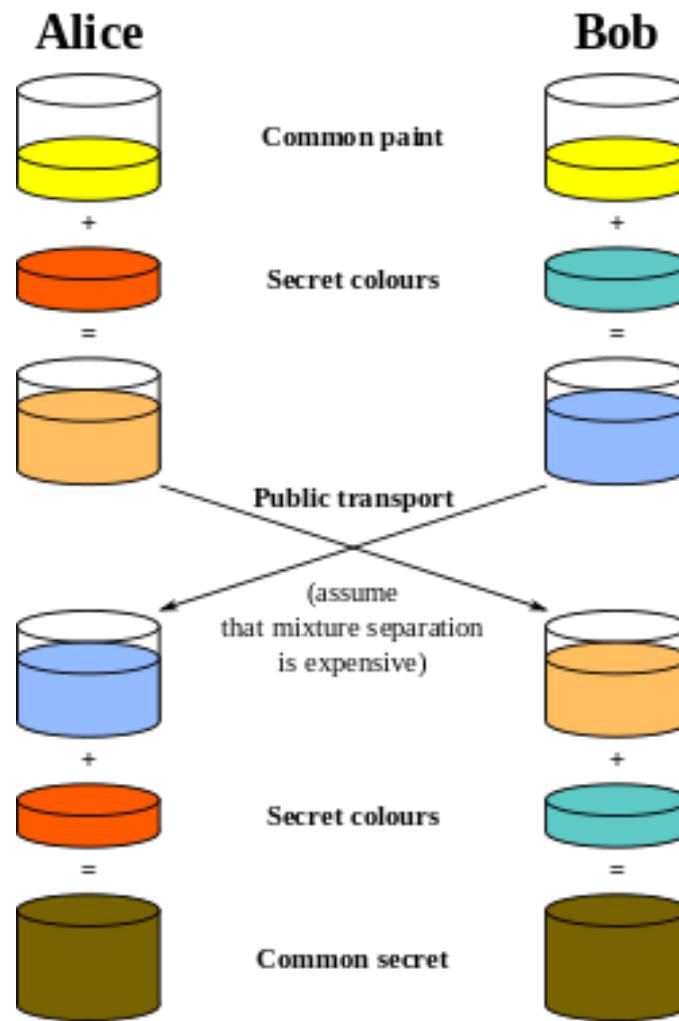
Encryption

- A symmetric encryption scheme consists of
 - A key generation algorithm
 - An encryption algorithm
 - A decryption algorithm
- An **encryption** algorithm E is an algorithm that
 - Takes a **plaintext** message M of arbitrary length $M \in \{0,1\}^*$
 - and a key $K \in \{0,1\}^n$ as input
 - and outputs a **ciphertext** $C = E_K(M) \in \{0,1\}^*$
- A **decryption** algorithm D is an algorithm that
 - Takes a ciphertext C and a key K as input
 - And outputs a plaintext $M = D_K(C)$
- For every K and every M , $D_K(E_K(M)) = M$

Kirckhoff Principle & Avalanche Effect

- A cryptosystem should be secure even if everything about the system, **except the key**, is **public knowledge**
- In contrast, keeping the design of a cryptosystem secret is often referred to “**security through obscurity**”
- **Avalanche effect:** **small change** in either **plaintext** or **the key** should produce a **significant change** in the **ciphertext**

Preview: Diffie-Hellman Key Exchange (src: Wikipedia)



Diffie-Hellman Key Exchange: Idea

- Prime numbers p and primitive root g to p are publicly known

(plural primitive roots) (mathematics, number theory) a number g is a primitive root modulo n if there is an integer 'k' for every 'a' coprime to 'n' such that $gk \equiv a \pmod{n}$;
 g is called a generator of the multiplicative group of integers modulo n .
- Alice picks a secret number a and computes $g^a \bmod p$ (let's call it A) and sends the result to Bob.
- Bob does the same thing, but with its own secret number b . So $g^b \bmod p$ (called B) is sent to Alice
- Now, Alice can compute $B^a \bmod p$.
- Bob can do the same with the input he got from Alice: $A^b \bmod p$.

Caesar Cipher

- Caesar cipher is a **shift cypher**. It shifts letters by a **fixed value**, e.g.,

A B C D E F G H I J K L M ...

F G H I J K L **M N O P Q R...**

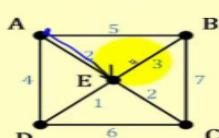
HELLO => MQQT

Encryption $E_k(x) = x + k \bmod 26$

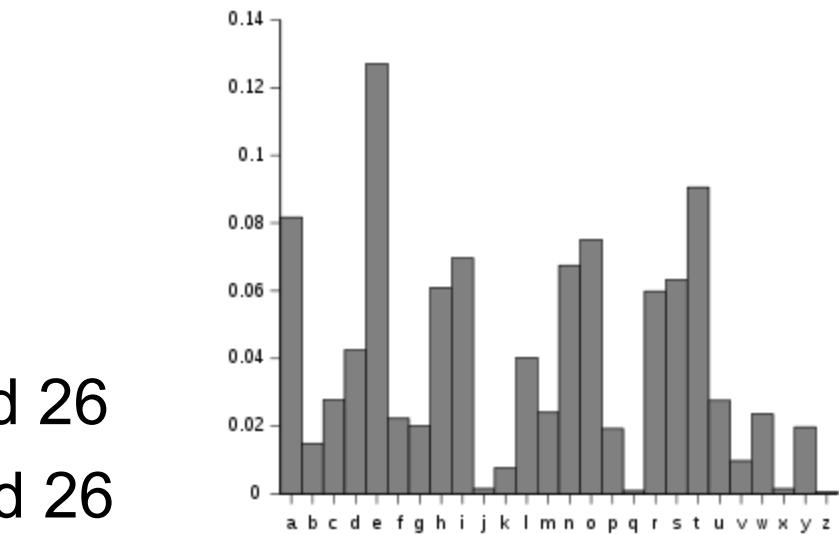
Decryption $D_k(y) = y - k \bmod 26$

- Another example: Column transposition
- Problems:** **brute force** attack and **frequency analysis**

Apply the brute force algorithm to find the lowest cost Hamiltonian circuit.



Circuit	Weight
ABCDEA	
ABCEDA	
ABECDA	
ADCBEA	
ADCEBA	
ADECBA	



Frequency analysis is the study of letters or groups of letters contained in a ciphertext in an attempt to partially reveal the message. The English language (as well as most other languages) have certain letters and groups of letters appear in varying frequencies.

Column Transposition

- Plaintext is written down in a rectangle, row by row, and read column by column. The order of columns is the key.

- Key: 4 3 1 2 5 6 7

- Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z

- Ciphertext: ttnaaptm. tsuoaodwcoixknlypetz

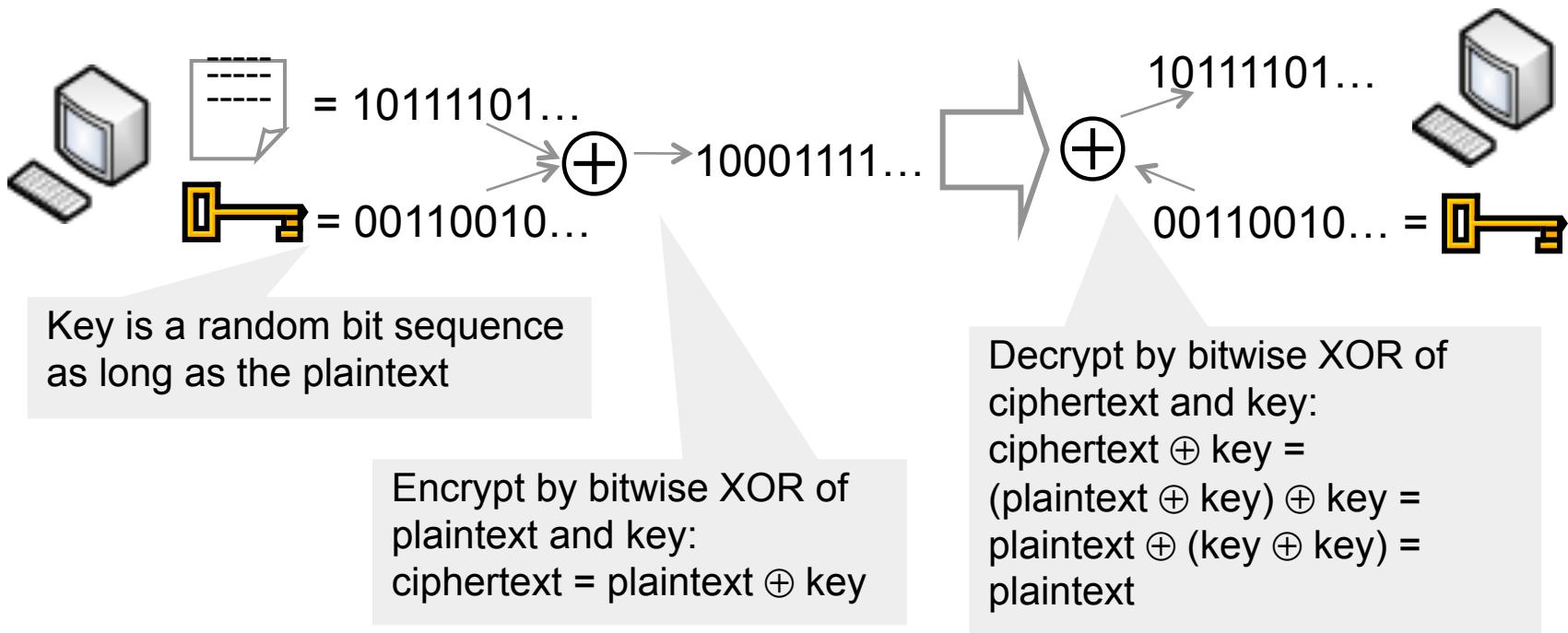
Encryption
Given text = Geeks for Geeks
Keyword = HACK Length of Keyword = 4 (no of rows) Order of Alphabets in HACK = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	-	f	o
r	-	G	e
e	k	s	-

Print Characters of column 1,2,3,4

Encrypted Text = e kefGsGsreko_

One-Time Pad



- A cipher achieves **perfect secrecy** if and only if there are **as many possible keys as possible plaintexts**, and **every key is equally likely** (Claude Shannon)

[Equally likely events are events that have the same theoretical probability (or likelihood) of occurring. Example. Each numeral on a die is equally likely to occur when the die is tossed.]

Advantages of One-Time Pad

Advantage: 1. Easy to compute, 2, Theoretically Secure

As all plaintext are equally likely and key must be same length as plaintext, then how does the sender communicate the each and every key to receiver?

Disadvantage:

1. Key must be as long as plaintext, 2. Does not guarantee integrity but only guarantees confidentiality, 3. Insecure if keys are reused, so that attacker obtain XOR of plaintext

■ Easy to compute

- Encryption and decryption are the same operation
- Bitwise XOR is very cheap to compute

■ As secure as theoretically possible

- Given a ciphertext, all plaintexts are equally likely, regardless of attacker's computational resources
- ...as long as the key sequence is truly random
 - True randomness is expensive to obtain in large quantities
- ...as long as each key is same length as plaintext
 - But how does the sender communicate the key to receiver?

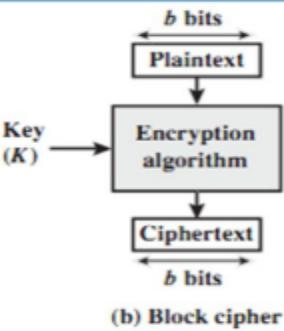
Problems with One-Time Pad

- Key must be as long as plaintext
 - Impractical in most realistic scenarios
 - Still used for diplomatic and intelligence traffic
- Does not guarantee integrity
 - One-time pad only guarantees confidentiality
 - Attacker cannot recover plaintext, but can easily change it to something else
- Insecure if keys are reused
 - Attacker can obtain XOR of plaintexts
- Obviously not practical

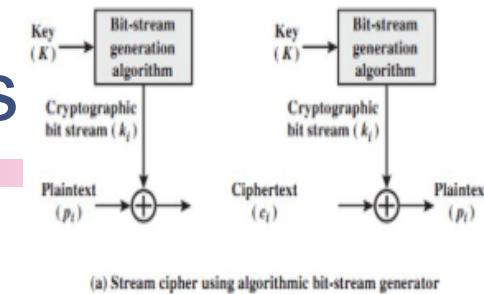
Brute Force Attacks

- Try every possible key
 - Successful on average after trying half of the keys
- Difficulty of brute force attack is proportional to key size

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs	Time required at 10^6 decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31} \mu\text{s} = 35.8 \text{ minutes}$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55} \mu\text{s} = 1142 \text{ years}$	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127} \mu\text{s} = 5.4 \times 10^{24}$ years	$5.4 \times 10^{18} \text{ years}$
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167} \mu\text{s} = 5.9 \times 10^{36}$ years	$5.9 \times 10^{30} \text{ years}$
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26} \mu\text{s} =$ $6.4 \times 10^{12} \text{ years}$	$6.4 \times 10^6 \text{ years}$



Block and Stream Ciphers



- Block ciphers encrypt blocks of plaintext of the same length with the same key
- Stream ciphers produce a pseudo-random stream of key bits
 - Plaintext is XORed bitwise with the key stream to produce ciphertext
- Block ciphers can, however, be turned into stream ciphers as we will see
- Stream ciphers are also block ciphers with a block size of “1”
- I. e. this distinction is somewhat blurred, particularly at the edges

(unable to see or be seen clearly.)

A solution or example that is not trivial. Often, solutions or examples involving the number zero are considered trivial. Nonzero solutions or examples are considered nontrivial.

For example, the equation $x + 5y = 0$ has the trivial solution $(0, 0)$. Nontrivial solutions include $(5, -1)$ and $(-2, 0.4)$.

Block Ciphers

Operate on a single chunk(block) of plaintext and same key is reused for each block. As plaintext bits were randomly shuffled, it look like a random permutation. Only break by brute-force and time & cost of breaking the cipher exceed useful lifetime, so it is non-trivial and only computational guarantee of secrecy.
Block Ciphers : DES, 3DES, AES, Twofish, etc.

- Operate on a single chunk (“block”) of plaintext
 - For example, 64 bits for DES, 128 bits for AES
 - Same key is reused for each block (can use short keys)
- Result should look like a random permutation
 - “As if” plaintext bits were randomly shuffled
- Only computational guarantee of secrecy
 - Not impossible to break, just very expensive
 - If there is no efficient algorithm (unproven assumption!), then can only break by brute-force, try-every-possible-key search
 - Time and cost of breaking the cipher exceed the value and/or useful lifetime of protected information

Commonly known Block Ciphers

- DES
- 3DES
- **AES**
- Twofish
- ...

DES

(Data Encryption Standard)

Data Encryption Standard(DES)

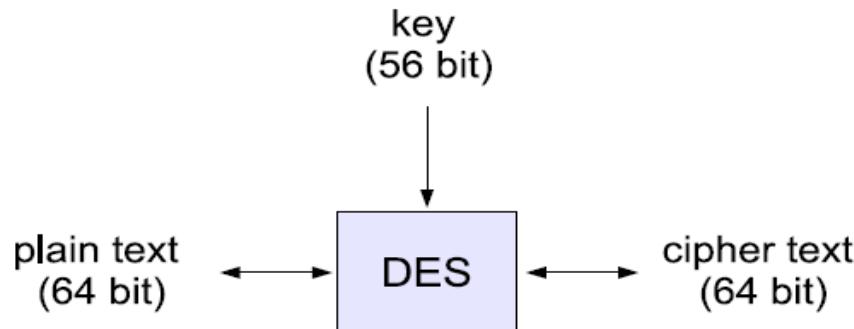
Question : like as,

1. theory,

2. structure,

3. about different different rounds, etc.

- Published in 1977 by the National Bureau of Standards*
 - Designed by IBM and the NSA
- Uses a 64-bit key and a block length of 64 bit
- Main operations: substitutions and permutations
- 8 bits of the key are used as parity bits
 - Effective key size is 56 bits



* called the National Institute of Standards and Technology (NIST) since 1988

Principles of DES

- First, each input block is subjected to a fixed input permutation
- Over the two resulting 32-bit blocks L and R , 16 similar encryption steps are executed, each depending on a 48-bit sub-key of the external (56-bit) key k .
 - Sub-keys are generated by a key selection procedure
- Finally, execution of an output permutation inverse to the input permutation
- Decryption analogous to encryption
 - 16 sub-keys are required in reverse order

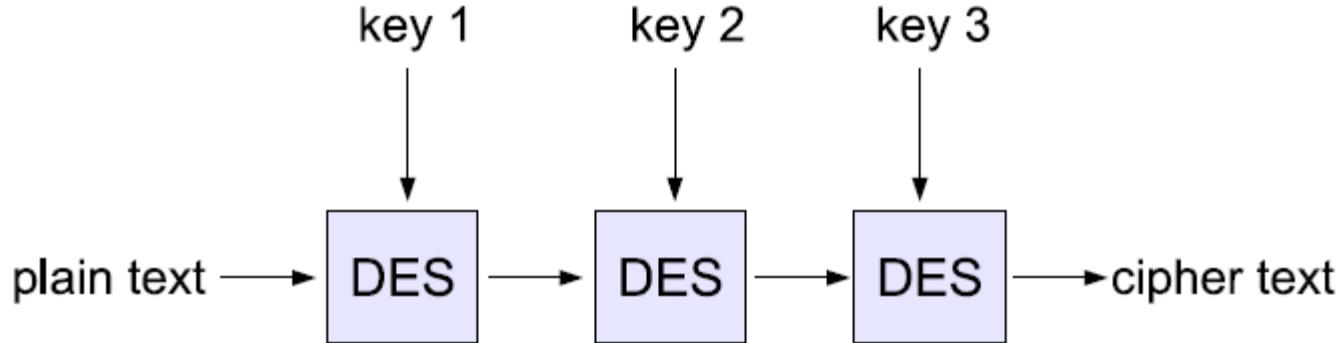
Security of DES

- January 13th, 1999: DES key broken within 22 hours and 15 minutes
 - In a contest sponsored by RSA Labs using
 - EFF's Deep Crack custom DES cracker ...
 - ... and the idle CPU time of around 100,000 computers
- It is no longer advisable to use DES
 - Especially not for new applications
- Biggest weakness still is the key length of 56 bits only!

Problems with 2DES

- First idea to increase the key size of DES
 - Use DES twice in a row with two independent keys k_1, k_2
- Problem: this does not double the effective key size
- “Meet-in-the-middle-attack”
 - Assume attacker has a plaintext/ciphertext pair (M, C) with $\text{DES}(k_2, \text{DES}(k_1, M)) = C$ but no knowledge of the keys k_1, k_2
 - Attacker can compute a list of intermediate ciphertexts Z by encrypting M with each possible key k_1 : 2^{56} DES operations
 - Attacker can decrypt C with all possible k_2 until he finds one that matches one of the Z 's: again at most 2^{56} DES operations
 - Overall: at most $2 * 2^{56}$ DES operations to find the keys k_1, k_2
 - This is a known-plaintext attack against 2DES with a complexity of 2^{57}

3DES = “Triple DES”

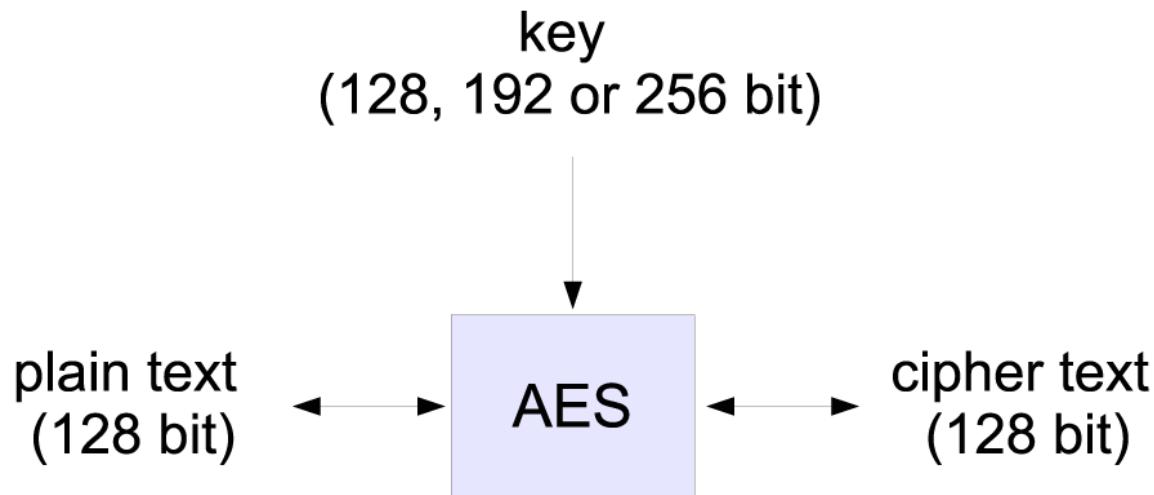


- Use DES three times in a row
 - Two variants in use: 3-key 3DES and 2-key 3DES
 - Both variants first use encryption with key1, decryption with key2, encryption with key3
 - 3-key 3DES: k1, k2, k3 pairwise different
 - 2-key 3DES: k1 = k3

AES

■ Goals

- More secure than 3DES
- More efficient than 3DES
- Support different key lengths



AES Selection

- January 1997: National Institute of Standardization
 - “[...] the AES would specify an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide.“
- August 1998: presentation of 15 candidates
 - Cast-256, Crypton, DEAL, DFC, E2, Frog, HPC, Loki97, Magenta, MARS, RC6, Rijndael, SAFER+, Serpent, Twofish
 - Broken under public scrutiny: DEAL, Frog, HPC, Loki97, Magenta
- August 1999: selection of 5 candidates for the next round
- October 2000: Rijndael is selected as AES
- November 2001: AES is standardized in FIPS 197

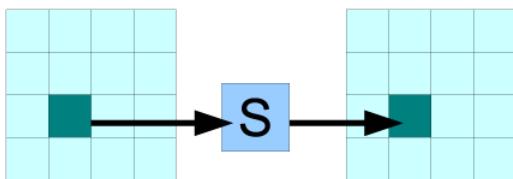
Structure of AES

- AES is round based
- AES uses a State Matrix with byte entries to represent the input and output of each round

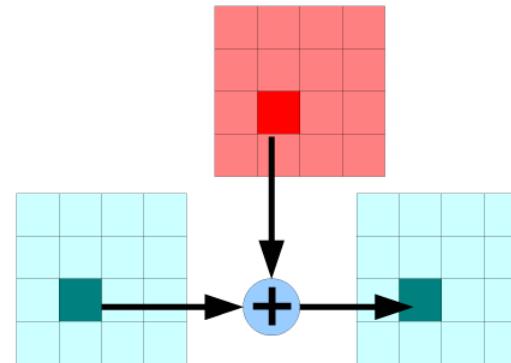


Operations used in each round

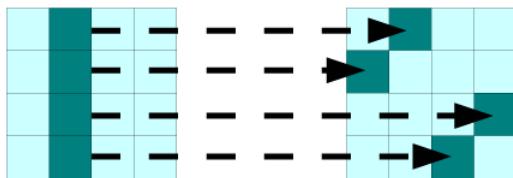
Byte Substitution (SB)



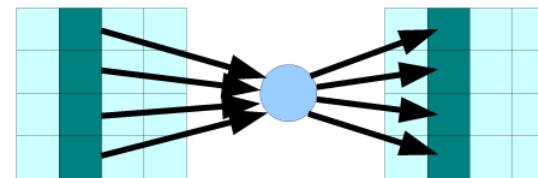
Key Addition (KA)



Mix Column (MC)



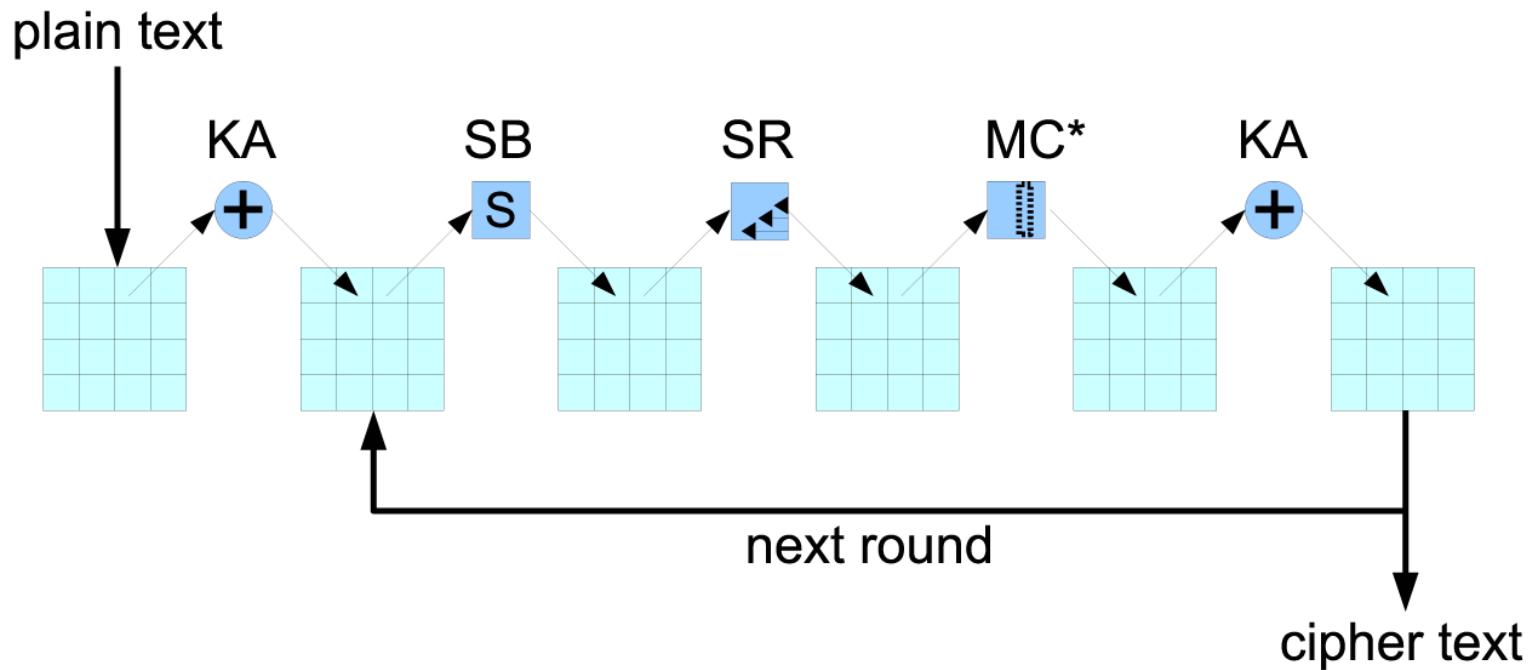
Shift Row (SR)



4 operation in each round:
1. Key Addition (KA)
2. Byte Substitution(SB)
3. Shift Row (SR)
4. Mix Column (MC)
5. then Again Key Addition (KA) for the next round.

Note: No Mix Column(MC) takes place in the last round.

Rounds



- The round key is different for each round and generated from the secret key
- * No Mix Column takes place in the last round

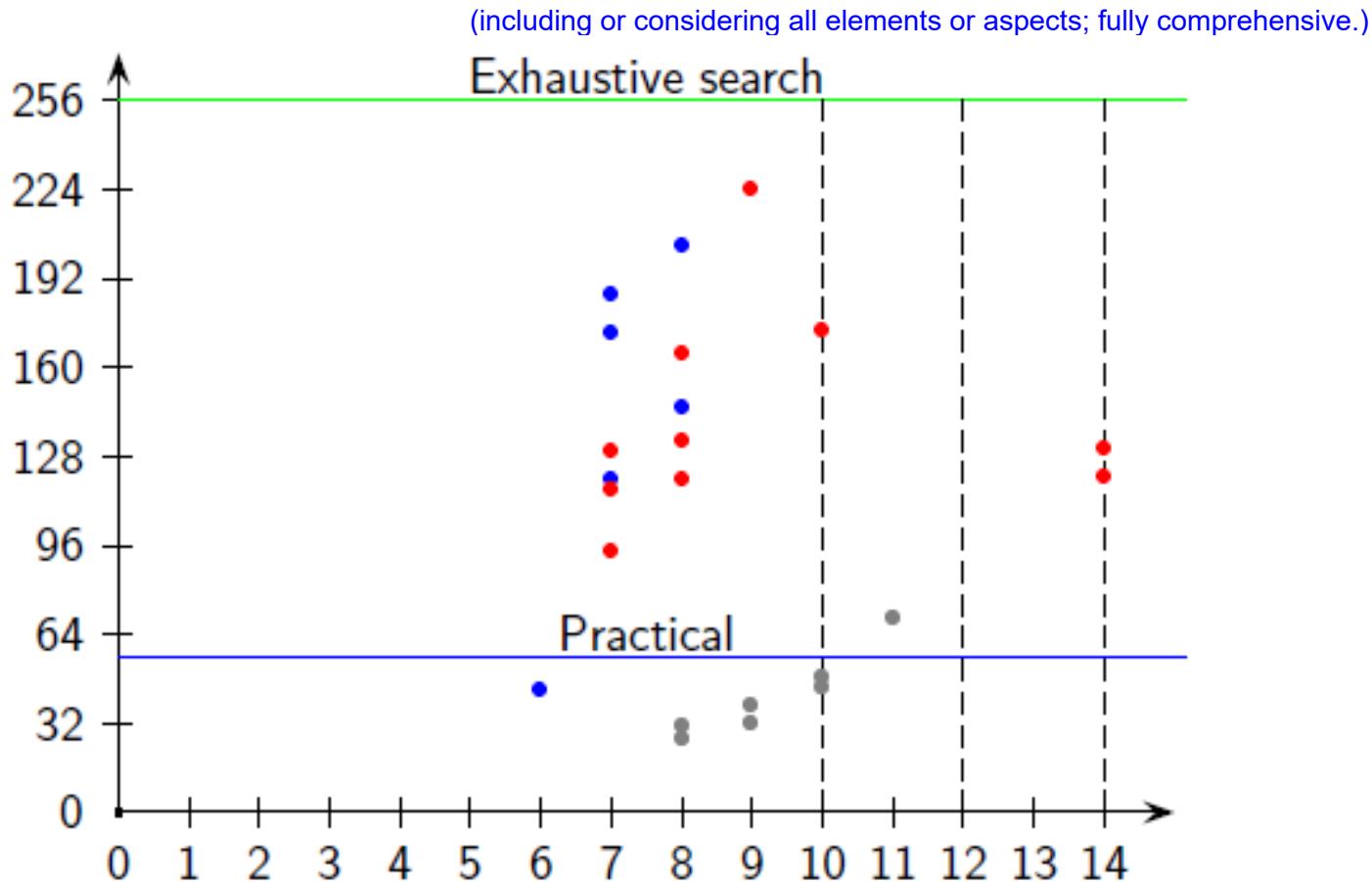
Number of Rounds

- Depends on the key length
 - 128 bit key – 10 rounds
 - 192 bit key – 12 rounds
 - 256 bit key – 14 rounds

Recent Attacks Against AES

- May and August 2009, Biryukov et al. University of Luxembourg
 - Related-key attacks on AES-256 and AES-192
 - Currently best attack against AES-256: key recovery attack with time complexity of 2^{119} (1.05×10^{22} years time complexity attacks.)
 - Attack against AES-192: key recovery within 2^{176}
 - Related-key attacks
 - Requires access to plaintexts encrypted with multiple keys that are related in a specific way
- No reason to worry yet
 - No attacks against full round AES-128 known that are better than brute force
 - No practical attacks against full round AES-256, AES-192

Overview on time-complexity of Attacks Against AES-256



Alex Biryukov, Orr Dunkelman, Nathan Keller,
Dmitry Khovratovich, Adi Shamir

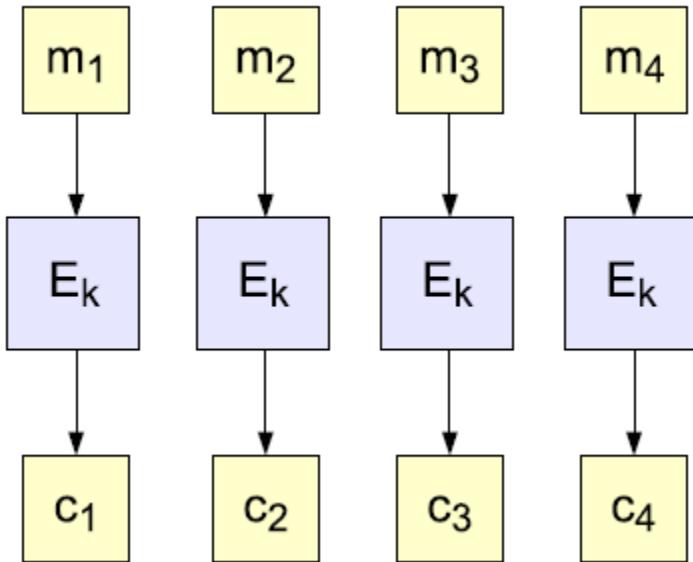
Encrypting a Large Message

To Encrypt a Large Message with a good Block cipher:

1. Electronic Code Book (ECB) mode - encrypt each one block separately using block cipher
2. Cipher Block Chaining (CBC) mode - XOR each block with the result of encrypting previous blocks

- So, we've got a good block cipher, but our plaintext is larger than 128-bit block size
- **Electronic Code Book (ECB) mode**
 - Split plaintext into blocks, encrypt each one separately using the block cipher
- **Cipher Block Chaining (CBC) mode**
 - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Also various counter modes, feedback modes, etc.

ECB Mode



- Encryption: $c_i = E_K(m_i)$
- Decryption: $m_i = D_K(c_i)$

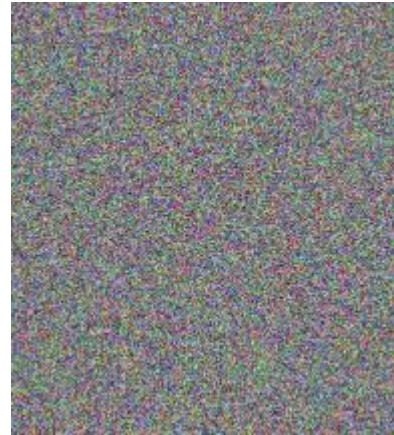
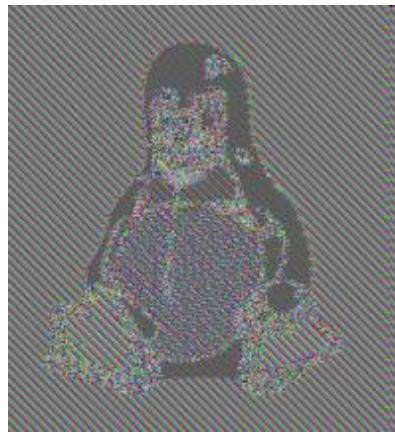
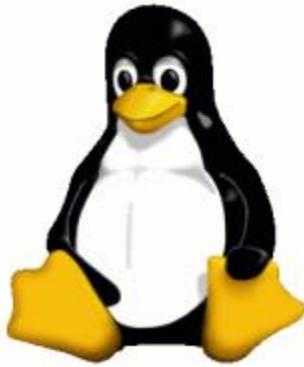
Disadvantage:

1. Same output block for same plaintext
2. Patterns for plaintext be recognized in ciphertext
3. Re-ordering or deletion of ciphertext not detected.

■ Disadvantages

- Same plaintext block always leads to the same output block
- Patterns in the plaintext block still show in the ciphertext
- Re-ordering or deletion of ciphertexts cannot be detected

Why ECB is Not Enough

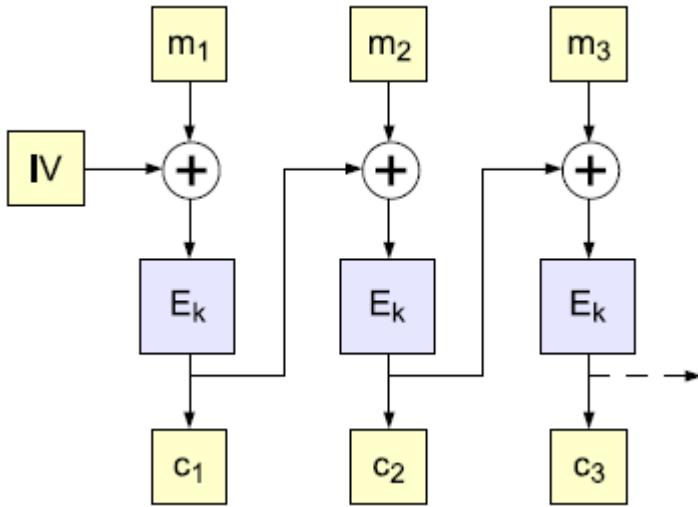


Problem: the original plaintext reveals without an individual block does not reveal anything.

[make (previously unknown or secret information) known to others.]

- Ciphertext as a whole in ECB Mode reveals information about the original plaintext as a whole
 - Even if an individual block does not reveal anything

Cipher Block Chaining Mode (CBC) - Mode

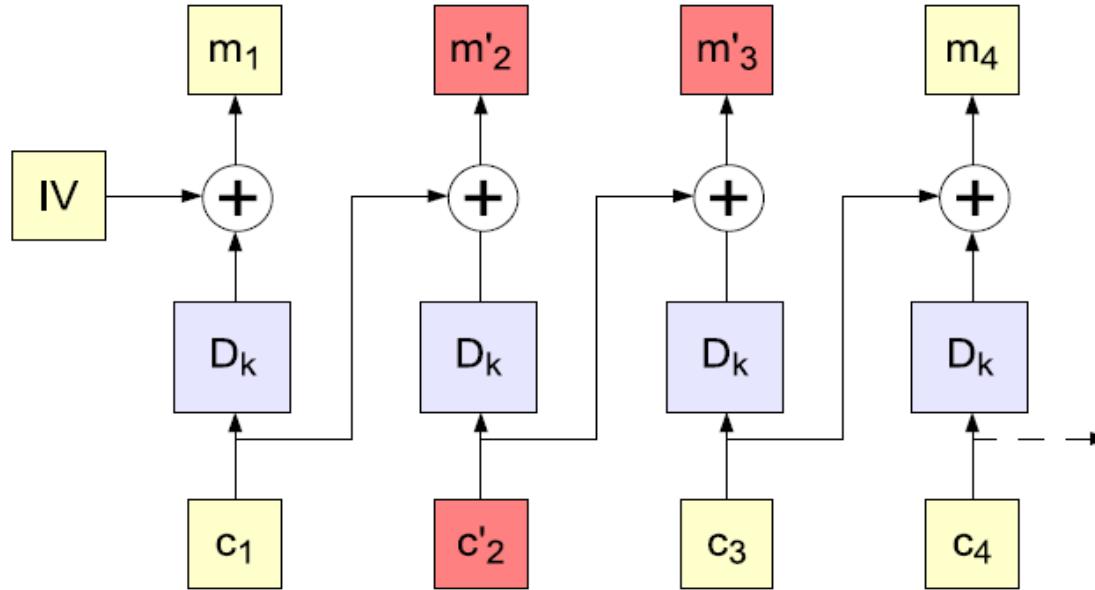


- $\text{IV} := c_0$
- Encryption: $c_i = E_k(m_i \text{ xor } c_{i-1})$
- Decryption: $m_i = D_k(c_i) \text{ xor } c_{i-1}$

- Uses the xor of plaintext block and the ciphertext block corresponding to the previous plaintext as input to the block cipher
- Advantages
 - Deletion of a ciphertext block can be detected
 - Re-ordering of ciphertext blocks can be detected
 - Self-synchronizing on transmission errors

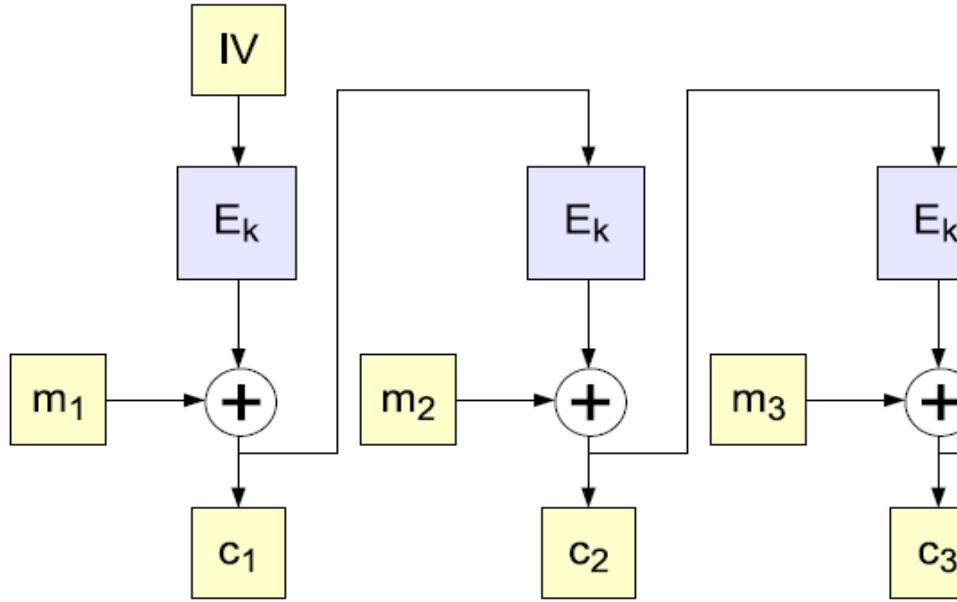
Self-Synchronization

CBC Mode Decryption Technique

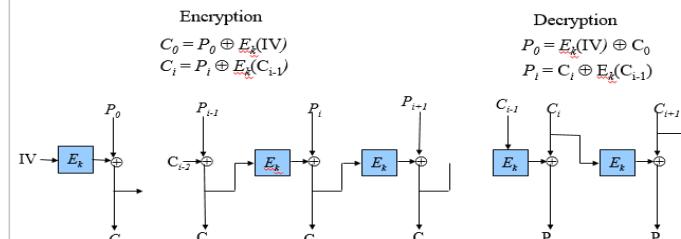


- Transmission error in c_2' will only influence m_2' and m_3'
- Subsequent plaintext will be correctly recovered

Cipher Feedback Mode (CFB) - Simplified

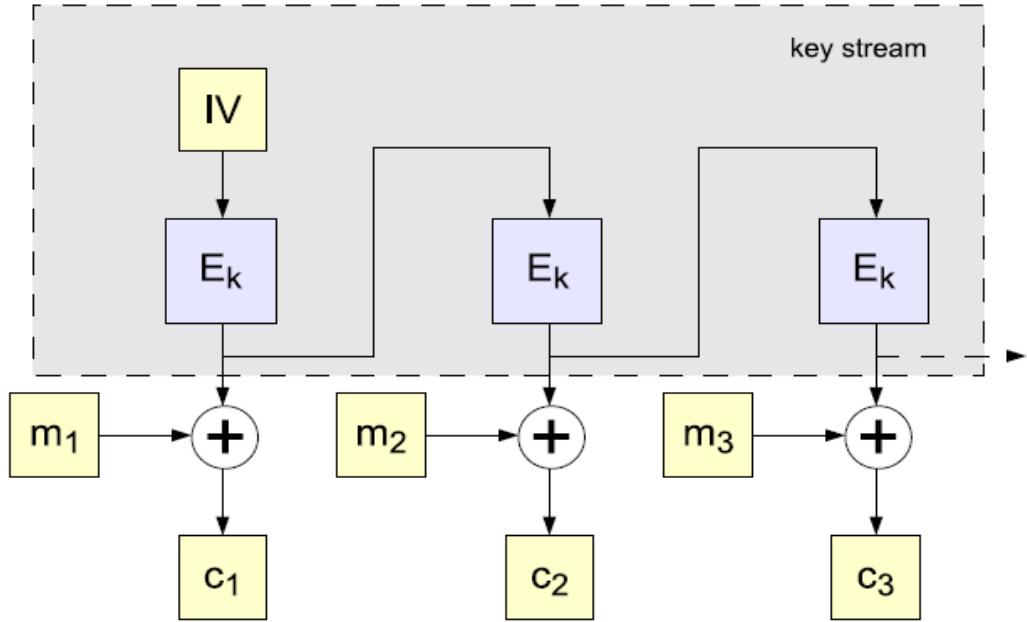


- IV public, IV : = c_0
- Encryption: $c_i = E_k(c_{i-1}) \text{ xor } m_i$
- Decryption: $m_i = c_i \text{ xor } E_k(c_{i-1})$



- Generates a key stream that depends on the ciphertext
- In the non-simplified version
 - block length of the encryption function is longer than plaintext block
 - part of the output of the encryption function is discarded
 - Non-discarded part is used to shuffle the bits of IV to the left

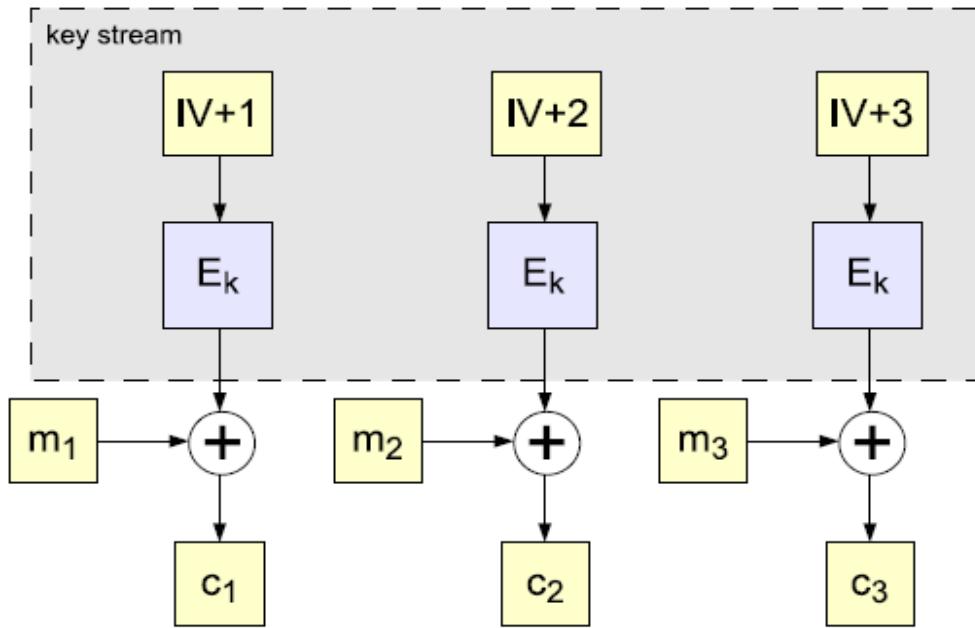
Output Feedback Mode (OFB) - Simplified



- IV public
 - Encryption: $c_i = E_k^i(\text{IV}) \text{ xor } m_i$
 - IV encrypted i-times
 - Decryption: $m_i = c_i \text{ xor } E_k^i(\text{IV})$
- Here we need Encryption

- Generates a key stream that does not depend on the ciphertext
- Key stream can be pre-computed as soon as IV is known
- Non simplified version as cipher feedback mode

Counter Mode (CTR)



- IV public
- Encryption: $c_i = E_k^i(IV+i) \text{ xor } m_i$
- Decryption: $m_i = c_i \text{ xor } E_k^i(IV+i)$

- Like OFB turns a block cipher into a stream cipher
- Can additionally be parallelized as there is no feedback

Important Properties of the Modes

■ OFB, CFB and CTR

- Not restricted to complete blocks
- Turn a block cipher into a stream cipher (to some extent)
 - Plaintext is xored with key stream bits, key stream depends on IV, Counter, and/or the last ciphertext block

■ ECB, CBC

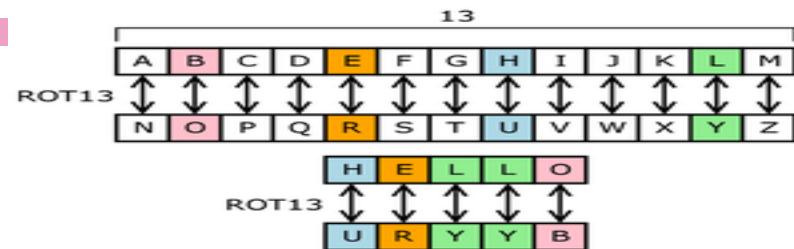
- Require padding to complete blocks
- Padding has to be easy to strip-off

Required padding to complete the blocks,
1. ECB - Electronic Code Book mode
2. CBC - Cipher Block Chaining mode

Turn a block cipher into stream cipher to some extent
1. CFB - Cipher Feed Back mode
2. OFB - Output Feed Back mode
3. CTR - Counter Mode

When Is a Cipher “Secure”?

FIXED MAPPING



- Hard to recover the key?
 - What if attacker can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
 - What if attacker learns some bits or some function of bits?
- Fixed mapping from plaintexts to ciphertexts?
 - What if attacker sees two identical ciphertexts and infers that the corresponding plaintexts are identical?
 - Implication: encryption must be randomized or stateful

[deduce or conclude (something) from evidence and reasoning rather than from explicit statements.]

STATEFUL means the computer or program keeps track of the state of interaction, usually by setting values in a storage field designated for that purpose. Stateless means there is no record of previous interactions and each interaction request has to be handled based entirely on information that comes with it.

Cipher is secure, if attacker can not learn
1. plaintext without learning the key,
2. hard to recover with some bits or some function of bits are known
3. sees two identical ciphertexts and infers the corresponding plaintext as fixed mapping. So encryption must be randomized and stateful.

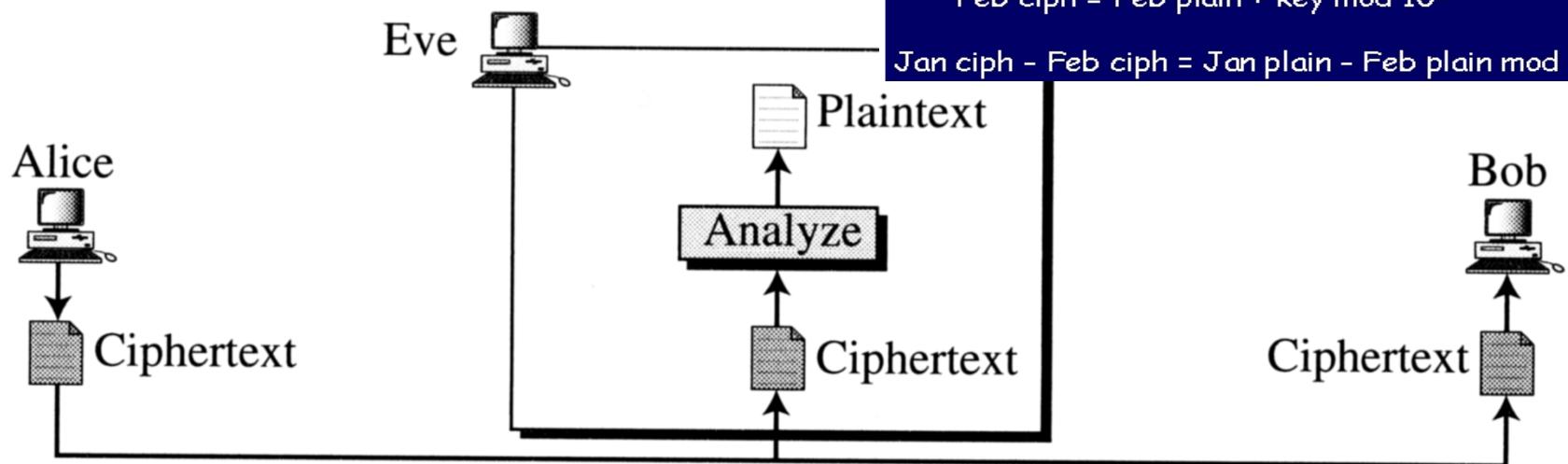
How Can a Cipher Be Attacked?

- Attackers knows ciphertext and encryption algorithm
 - Main question: what else does the attacker know?
 - Depends on the application in which the cipher is used!
- Brute-force attack: try out all possible keys
- Ciphertext-only attack
- Known-plaintext attack (stronger)
 - Knows some plaintext/ciphertext pairs
- Chosen-plaintext attack (even stronger)
 - Can obtain ciphertext for any plaintext of his choice
- Chosen-ciphertext attack (very strong)
 - Can decrypt any ciphertext except the target before target is known
- Adaptive chosen-ciphertext attack
 - Can decrypt any ciphertext chosen adaptively, i.e. depending on the target and the result of the previous ciphertexts

Ciphertext-only Attack

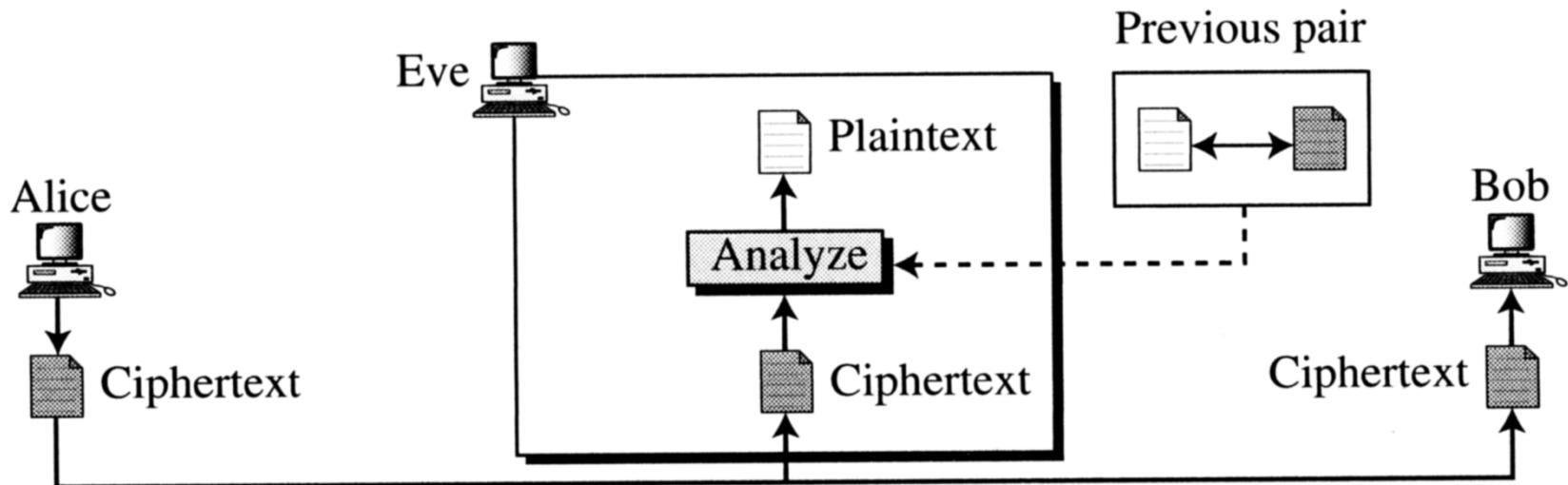
- An attacker tries to recover the **plaintext** but has access only to the **ciphertext**

To get useful information about plaintext.
Each month Alice sends Bob amount to spend,
encrypted with 20-digit addition cipher, same key.
Eve intercepts Jan and Feb ciphertexts.

$$\text{Jan ciph} = \text{Jan plain} + \text{key mod } 10^{20}$$
$$\text{Feb ciph} = \text{Feb plain} + \text{key mod } 10^{20}$$
$$\text{Jan ciph} - \text{Feb ciph} = \text{Jan plain} - \text{Feb plain mod } 10^{20}$$


Known-plaintext Attack

- The attacker tries to recover the plaintext from the ciphertext ...
- ... and has access to some pairs of plaintext and ciphertext



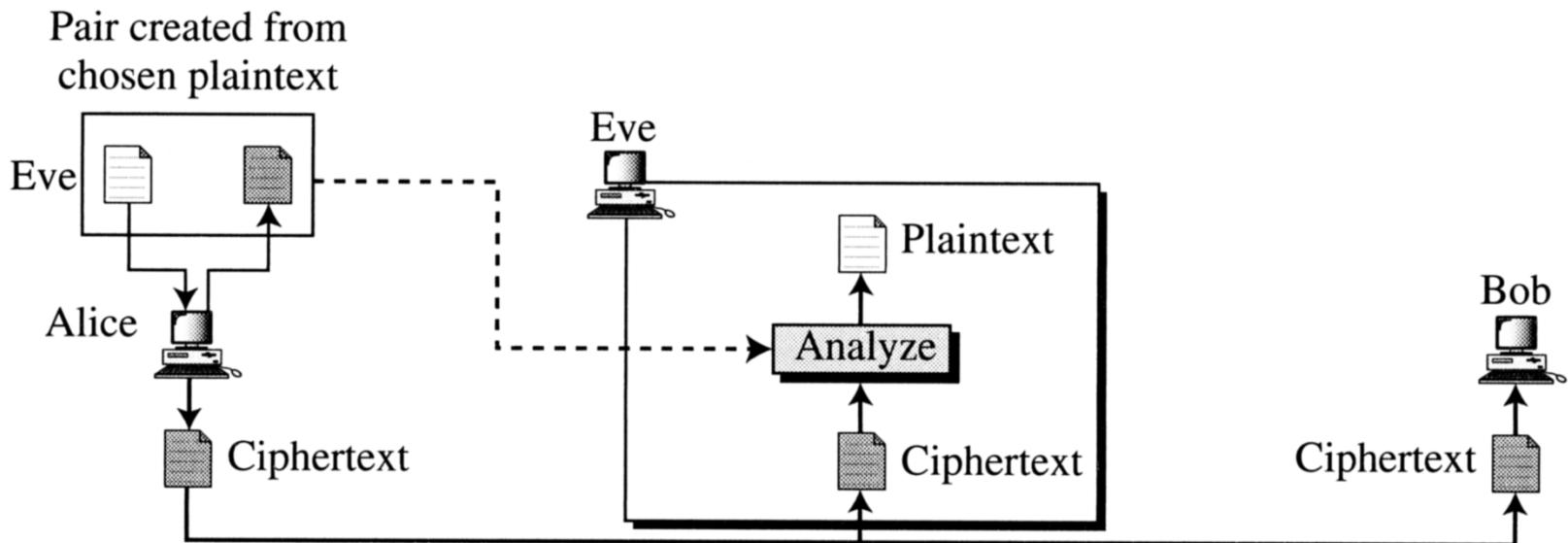
Alice sends Bob message using block size of 20 digits. Eve intercepts, and she knows message starts with "Dear Bob" => knows both plaintext and ciphertext of first block.

$$\text{cyph} = \text{plain} + \text{key mod } 10^{20} \Rightarrow \text{key} = \text{cyph} - \text{plain mod } 10^{20}$$

Also works if Eve knows of some number of ways Alice's message likely to begin. Tries each one => set of possible keys => tries each key on entire message. Using prior knowledge about message -- in English

Chosen-plaintext Attack

- The attacker tries to recover the **plaintext** from the ciphertext ...
- ... and can **obtain ciphertexts for plaintexts** of his choice

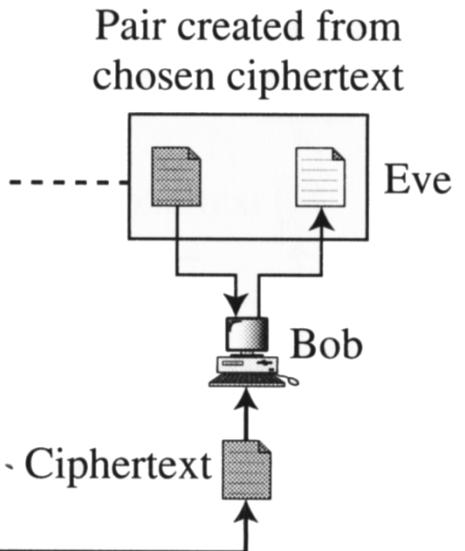
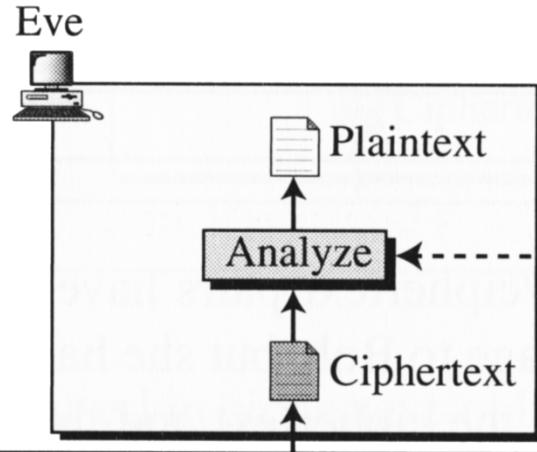
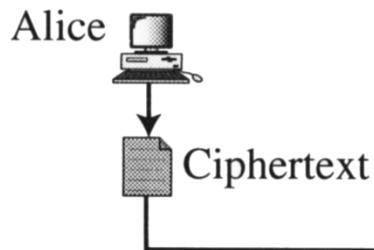


1. Chooses plaintext and receives corresponding ciphertext, 2. Not always viable (E.g. Enigma WWII), 3.Two variations Two types:
 - a. Batch Chosen-Plaintext - Attacker chooses a “batch” of plaintexts before any encrypted ciphertext is received
 - b. Adaptive chosen-plaintext - Attacker makes n-amounts of interactive queries and alters their plaintext based on the previous query

Chosen-ciphertext Attack

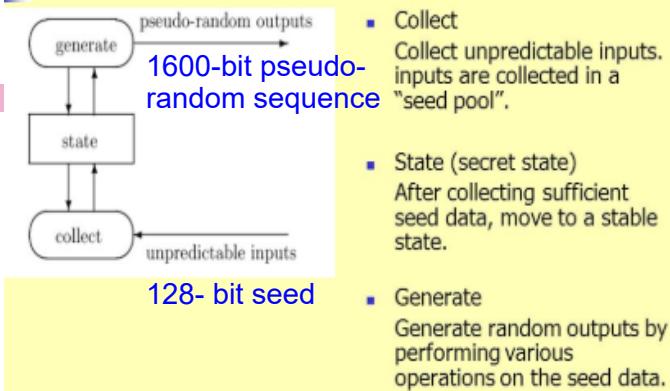
- The attacker tries to recover the **plaintext** from the ciphertext ...
- ... and can select ciphertexts (other than the target) for which **he can obtain plaintexts**

1. Tries to discover the key
2. Uses ciphertext chosen by attacker
3. Relies on being able to obtain decrypted plaintext
4. Two variations



Stream Ciphers

- Remember the one-time pad?
 - $E_K(M) = M \text{ xor Key}$
 - Key must be a random bit sequence as long as message
- Idea: replace “random” with “**pseudo-random**”
 - Encrypt with pseudo-random number generator (PRNG)
 - PRNG takes a short, **truly random secret seed** and expands it into a **long “random-looking”** sequence
 - E.g., 128-bit seed into a 1600-bit pseudo-random sequence
- $E_K(M) = IV, M \text{ xor PRNG}(IV, K)$
 - Message **processed bit by bit**, not in blocks



- Collect
Collect unpredictable inputs. Inputs are collected in a “seed pool”.
- State (secret state)
After collecting sufficient seed data, move to a stable state.
- Generate
Generate random outputs by performing various operations on the seed data.

A random seed (or seed state, or just seed) is a number (or vector) used to initialize a pseudorandom number generator. ... The choice of a good random seed is crucial in the field of computer security. When a secret encryption key is pseudorandomly generated, having the seed will allow one to obtain the key.

Properties of Stream Ciphers

Property of Steam Cipher:
1. Very Fast,
2. Do not provide perfect secrecy as block cipher
3. PRNG is unpredictable so it is hard to predict the next bit as next bit will be '0' or '1' have probability 1/2
Weakness/Problem:
1. No Integrity
2. Known Plain Text Attack for keystream ever repeated, if attacker know M1, then easily recover M2

- Usually **very fast** (faster than block ciphers)
 - Used where speed is important: WiFi, DVD, speech
- Unlike one-time pad, stream ciphers **do not provide perfect secrecy**
 - Only as secure as the underlying PRNG
 - If used **properly**, can be **as secure as block ciphers**
- PRNG is, by definition, **unpredictable**
 - Given the stream of PRNG output (but not the seed!), **it's hard to predict what the next bit will be**
 - **If PRNG(unknown random seed)= $b_1 \dots b_i$, then b_{i+1} is "0" with probability $\frac{1}{2}$, "1" with probability $\frac{1}{2}$**

Weaknesses of Stream Ciphers

- No integrity
 - Associativity & commutativity: $(X \text{ xor } Y) \text{ xor } Z = (X \text{ xor } Z) \text{ xor } Y$
 - $(M1 \text{ xor } \text{PRNG(seed)}) \text{ xor } M2 = (M1 \text{ xor } M2) \text{ xor } \text{PRNG(seed)}$
- Known-plaintext attack is very dangerous if keystream is ever repeated
 - Self-cancellation property of XOR: $X \text{ xor } X = 0$
 - $(M1 \text{ xor } \text{PRNG(seed)}) \text{ xor } (M2 \text{ xor } \text{PRNG(seed)}) = M1 \text{ xor } M2$
 - If attacker knows $M1$, then easily recovers $M2$
 - Most plaintexts contain enough redundancy that knowledge of $M1$ or $M2$ is not even necessary to recover both from $M1 \text{ xor } M2$

Stream Cipher Terminology

- Seed of pseudo-random generator often consists of initialization vector (IV) and key
 - IV is usually sent with the ciphertext
 - The key is a secret known only to the sender and the recipient, not sent with the ciphertext
- The pseudo-random bit stream produced by $\text{PRNG}(\text{IV}, \text{key})$ is referred to as keystream $= \text{PRNG}(\text{IV}, \text{key})$
 - PRNG must be cryptographically secure
- Encrypt message by XORing with keystream
 - ciphertext = message xor keystream

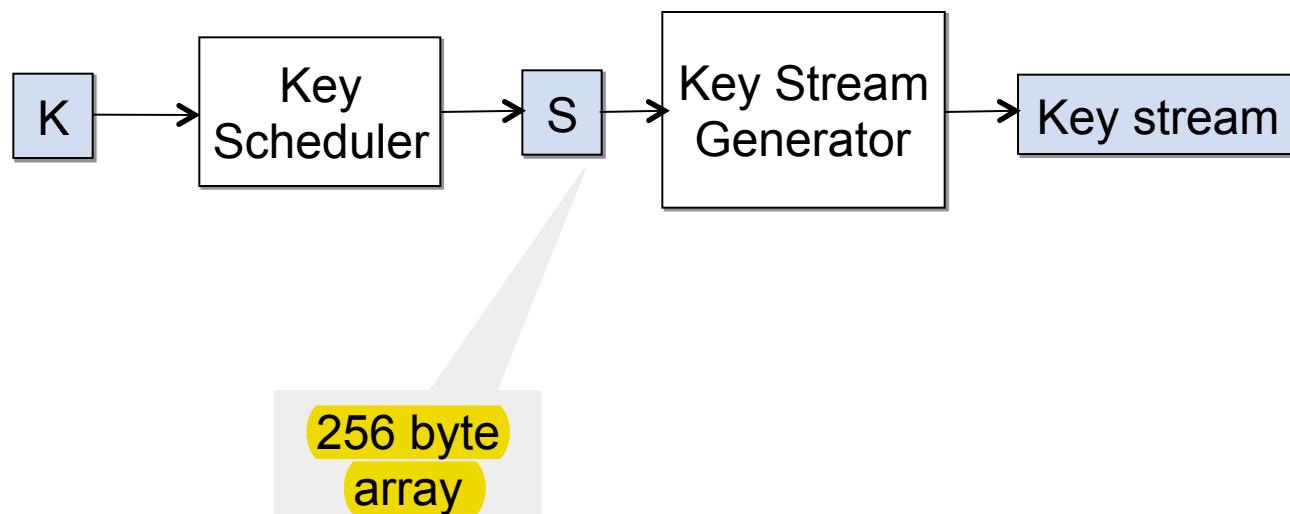
1. Seed of pseudo-random generator consist of initialized vector(IV) and key. IV sent with ciphertext and secret key known by sender and receiver.
2. Pseudo-random bit stream, keystream= $\text{PRNG}(\text{IV}, \text{key})$
3. ciphertext = message XOR keystream.

Examples for Stream Ciphers

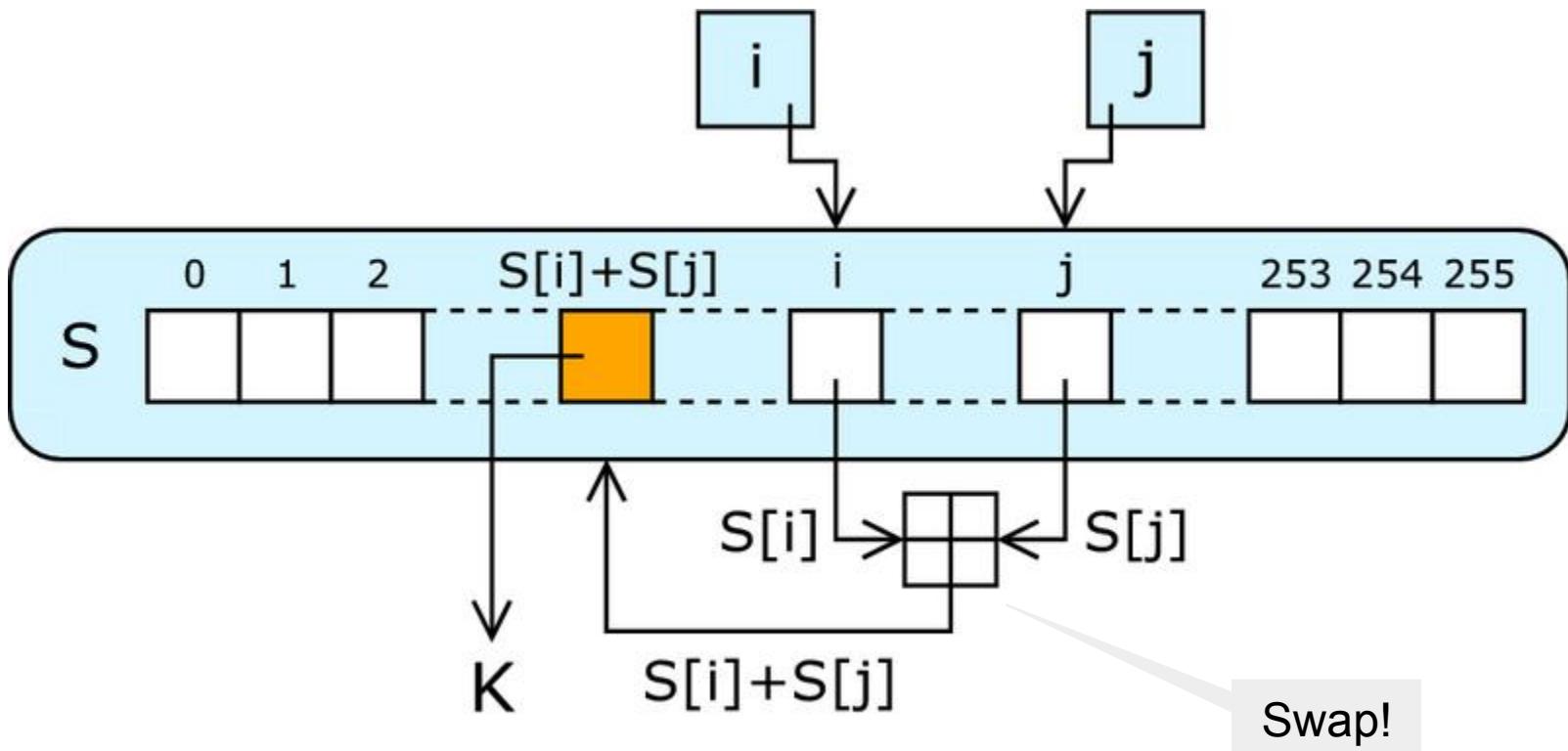
- RC4
 - Used, e.g. in WLAN, TLS, IPsec
- A5/1, A5/2
 - Used in GSM/GPRS
- SEAL
- ...

RC4

- Designed by Ron Rivest for RSA in 1987
 - Simple, fast, widely used
 - SSL/TLS for Web security, WLAN
 - Structure:



RC4 Key Stream Generation



- Key scheduler fills 256 byte array S
- Key stream byte K generated as illustrated above

Key Stream Generator

- In each round of the loop a key stream byte is generate

```
i = j := 0
loop
    i := (i+1) mod 256
    j := (j+S[i]) mod 256
    swap(S[i], S[j])
    output S[(S[i]+S[j]) mod 256]
end loop
```

RC4 Key scheduler – How S is filled

```
Divide key K into L bytes  
for i = 0 to 255 do  
    S[i] := i  
    j := 0  
    for i = 0 to 255 do  
        j := (j+S[i]+K[i mod L]) mod 256  
        swap(S[i],S[j])
```

Key can be any length
up to 2048 bits

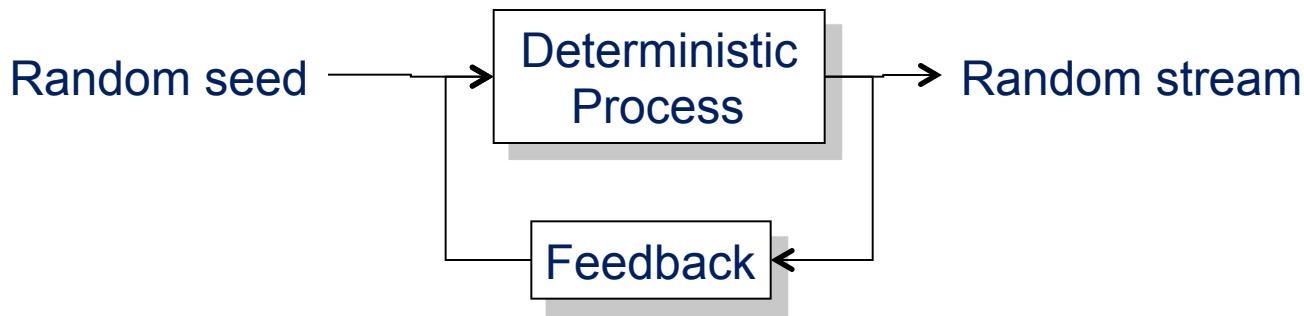
Generate initial
permutation
from key K

- To use RC4, usually pre-pend initialization vector (IV) to the key
 - IV can be random or a counter
- RC4 is not random enough! 1st byte of generated sequence depends only on 3 cells of state array S. This can be used to extract the key.
 - To use RC4 securely, RSA suggests discarding first 256 bytes

Fluhrer-Mantin-Shamir attack

(Pseudo) Random Number Generators

- Random Numbers can be generated by repeating an experiment with a random result
 - E.g. throwing a coin
- Pseudo Random Numbers just “look random” but are generated by a deterministic process with feed back using a (smaller) random “seed” as input



PRNGs

- Pseudo Random Number Generators (**PRNGs**) are used in cryptography for many different purposes
 - Generation of **symmetric keys**
 - Generation of **asymmetric keys or parameters used** in key generation
 - Generation of **random challenges** in authentication mechanisms
- PRNGs are typically based on **PR Bit Gs** that generate one pseudo random output bit
- Some standards also use the term **Pseudo Random Function (PRF)** instead of PRNG

PRBGs

- A PRBG is said to **pass the next bit test** if there is no **polynomial-time** algorithm, which on input of the **first k bits** of the output of **PRBG** can predict the next bit with probability greater than $\frac{1}{2}$
- A PRNG that is based on **a PRBG that passes the next bit test** is called **cryptographically secure**
- **Cryptographically secure** PRBGs can be **constructed from**
 - **(Keyed) Hash functions** (see next chapter)
 - **Block ciphers**
 - **Number theoretic problems**

Cryptographically Secure PRNG

- Next-bit test
 - Given N bits of the pseudo-random sequence, predict $(N+1)$ st bit
 - Probability of correct prediction should be very close to $1/2$ for any efficient adversarial algorithm
- PRNG state compromise
 - Even if attacker learns complete or partial state of the PRNG, he should not be able to reproduce the previously generated sequence
 - ... or future sequence, if there'll be future random input(s)
- Common PRNGs are not cryptographically secure

Reading and Figure Credits

- Basics
 - Stallings: Chapter on Symmetric Encryption
 - Kaufman: Chapters 3 and 4
- Further Reading
 - Random Numbers: RFC 1750
 - Really nice comic on AES
 - <http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html>
- Figure Credits: Forouzan “Introduction to Cryptography and Network Security”