IT-Security 1

Chapter 2: Symmetric Encryption

Prof. Dr.-Ing. Ulrike Meyer

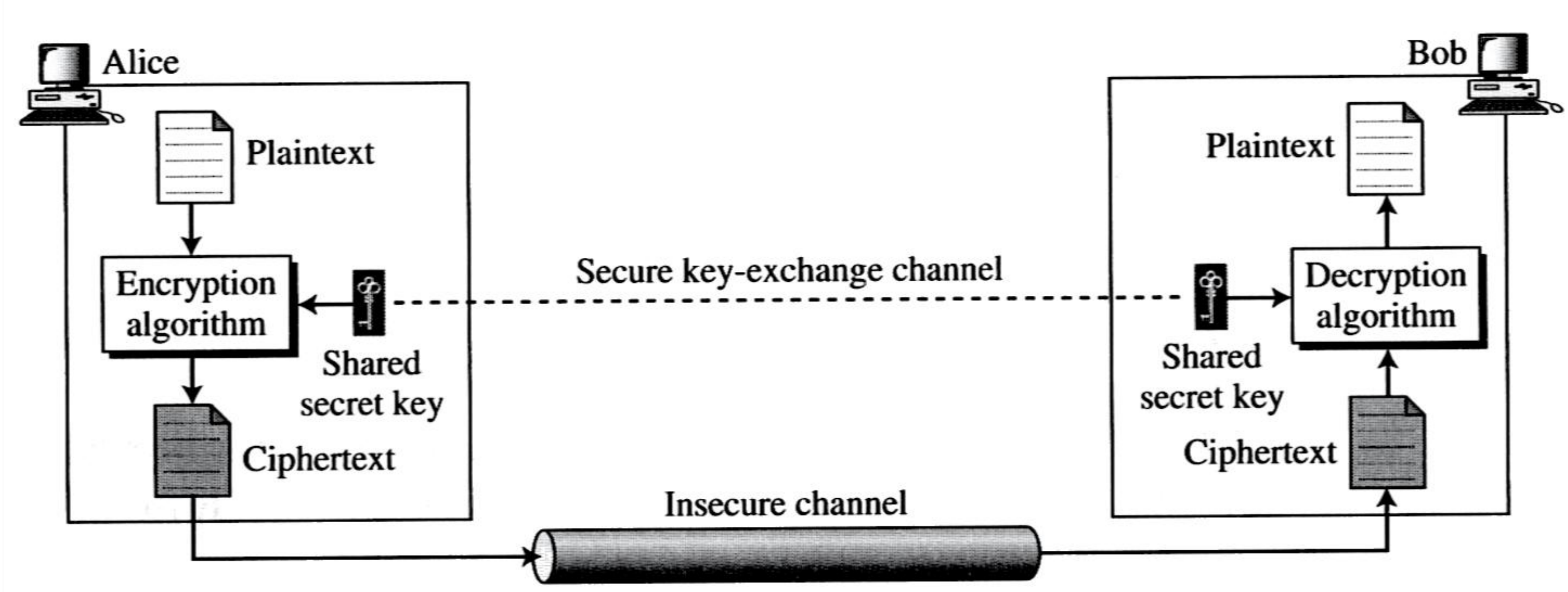
WS 15/16

Chapter Overview

- General Idea of Symmetric Encryption
- Block ciphers
- Modes to use block ciphers
- Stream ciphers
- Classification of attacks against ciphers

General Idea of Symmetric Encryption

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



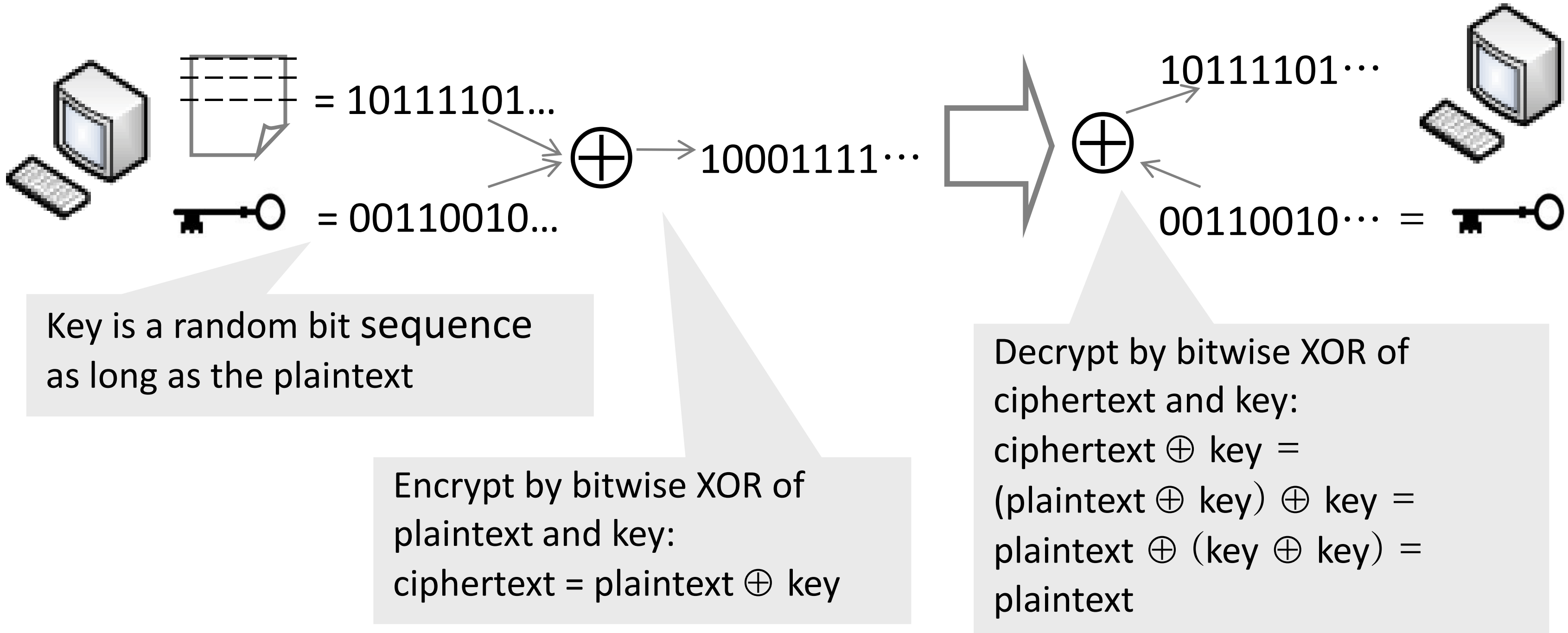
Encryption Scheme

- 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

- 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 as “security through obscurity”

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)

Advantages of One-Time Pad

- 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 selected uniformly at random
    - True randomness is expensive to obtain in large quantities
  - …as long as the key is of the same length as the plaintext
    - But how does the sender communicate the key to the 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 for all applications

When Is a Cipher “Secure”?

- So, if we typically will not get perfect secrecy, when do we call a cipher “secure” anyway?
- 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

How Can a Cipher Be Attacked?

- Assumption: Attacker 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

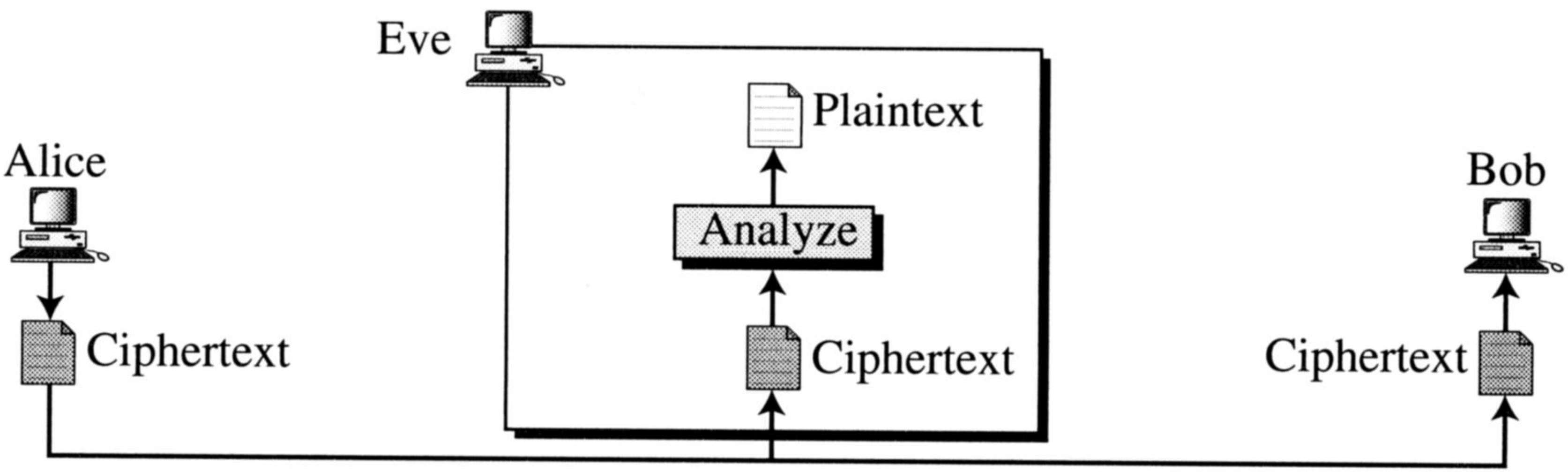
Brute Force Attacks

- 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} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} µs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} µs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} µs = 6.4×10^{12} years	6.4×10^6 years

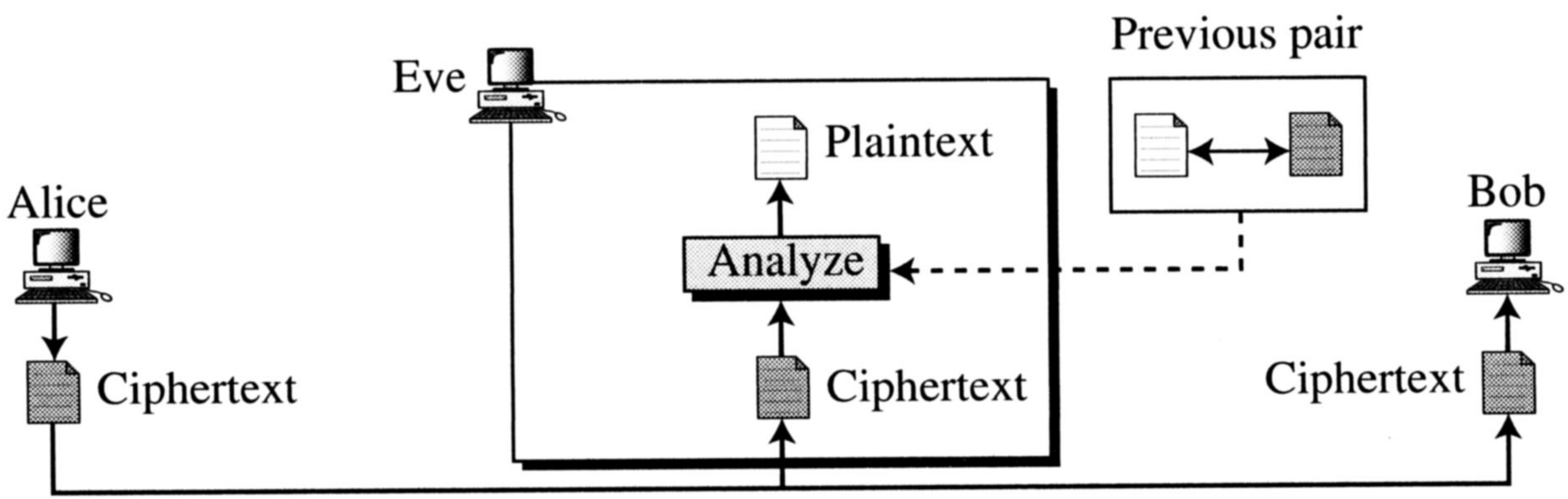
Ciphertext-only Attack

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



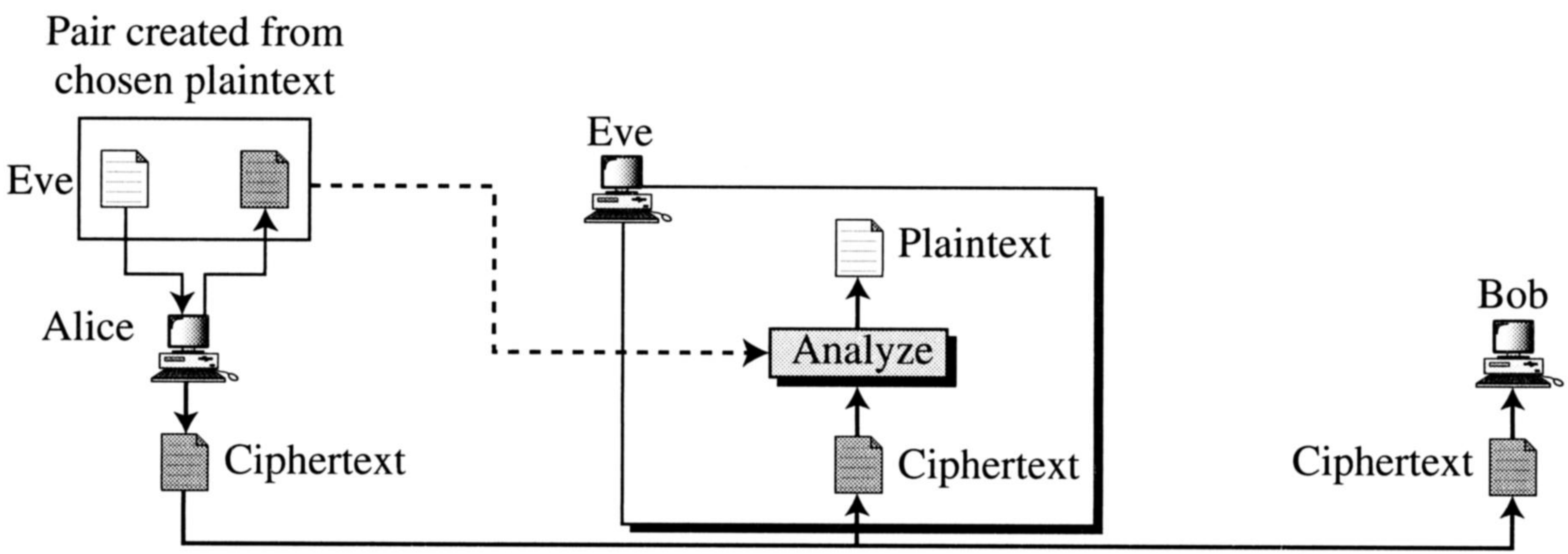
Known-plaintext Attack

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



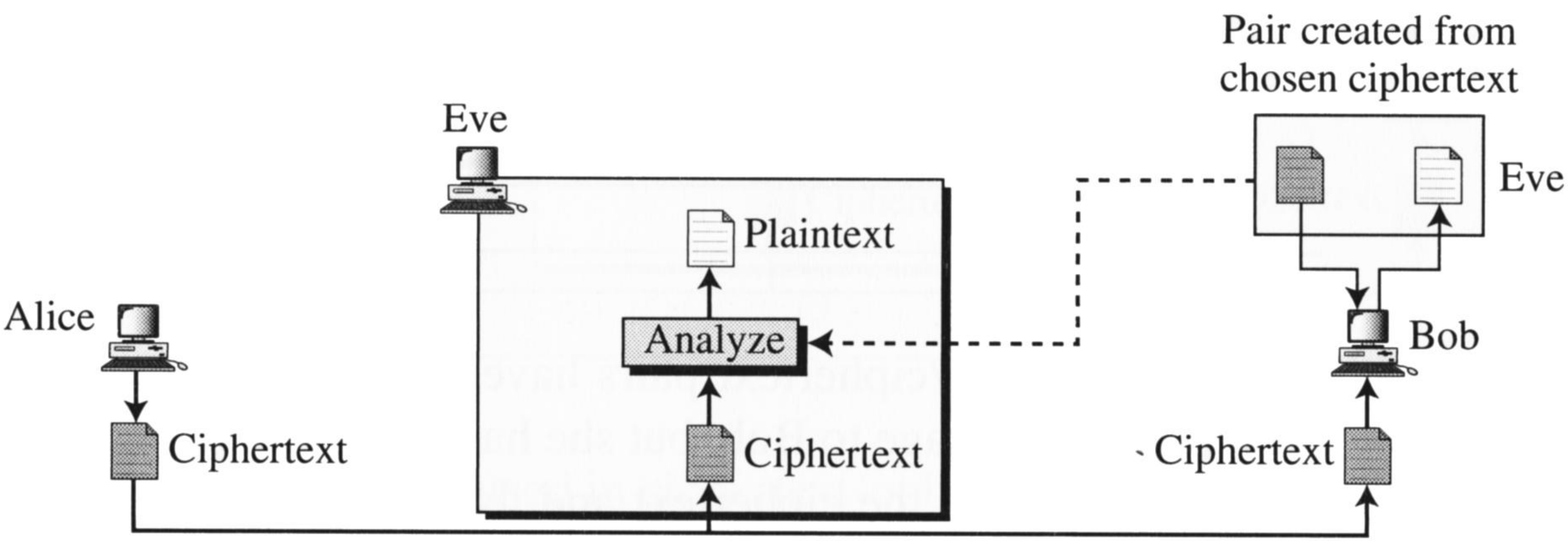
Chosen-plaintext Attack

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



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



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

Block Ciphers

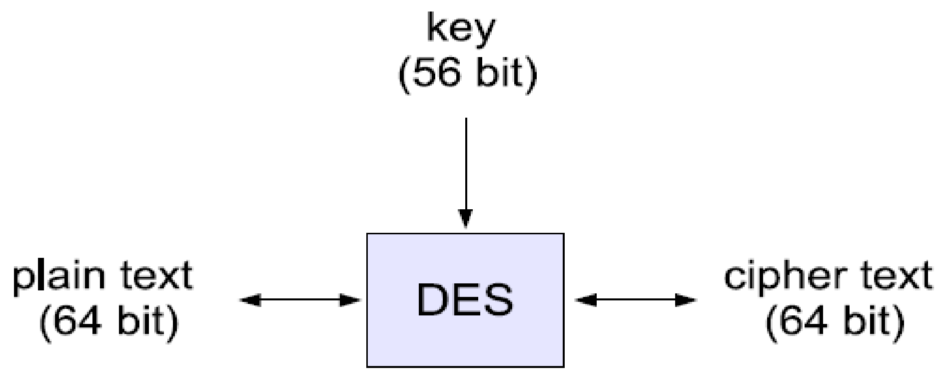
- Operate on a single chunk (“block”) of plaintext
  - For example, 64 bits for DES, 128 bits for AES-128
  - Same key is reused for each block (can use short keys)
- Result should look like a random permutation
  - “As if” plaintext bits were randomly shuffled
- Only computational guarantee of secrecy
  - Not impossible to break, just very expensive
    - If there is no efficient algorithm (unproven assumption!), then can only break by brute-force

Commonly used Block Ciphers

- DES
- 3DES
- AES
- Twofish
- …

DES

- 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
- 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

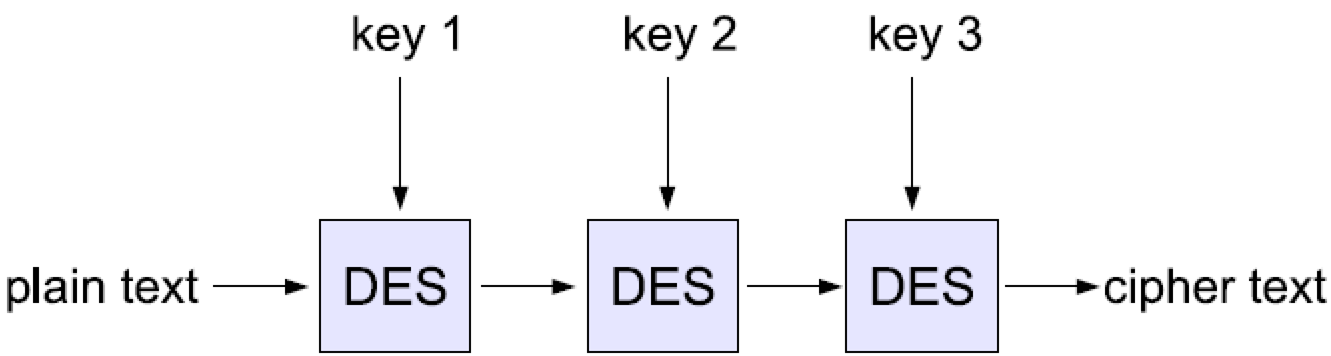
Security of DES

- 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 k1, k2
- Problem: this does not double the effective key size
- “Meet-in-the-middle-attack”
  - Assume attacker has a plaintext/ciphertext pair (M,C) with DES(k2,DES(k1,M)) = C but no knowledge of the keys k1, k2
  - Attacker can compute a list of intermediate ciphertexts Z by encrypting M with each possible key k1: 2^{56} DES operations
  - Attacker can decrypt C with all possible k2 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 k1, k2
  - This is a known-plaintext attack against 2DES with a complexity of 2^{57}

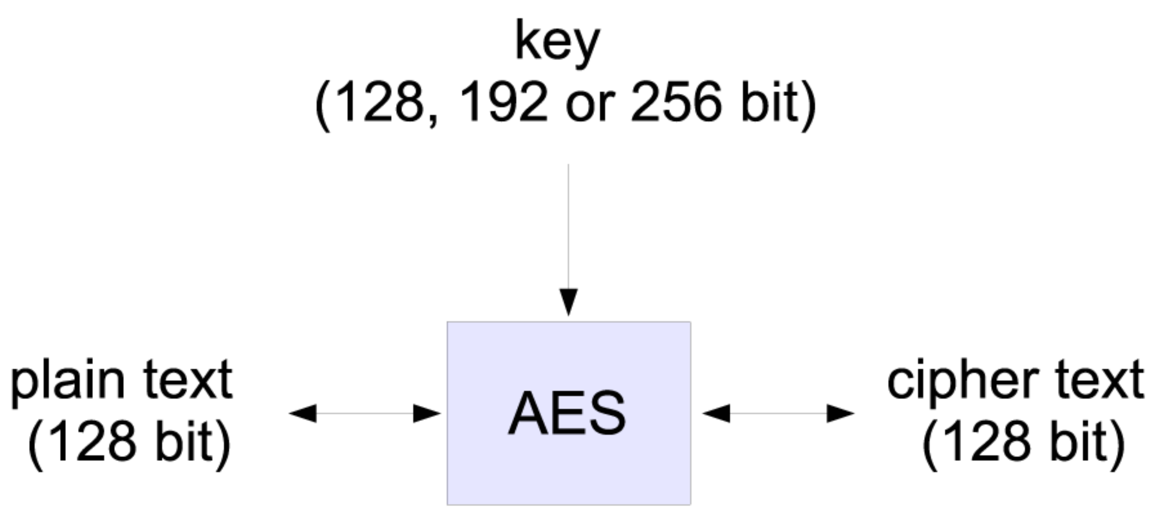
3DES = “Triple DES”



- 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

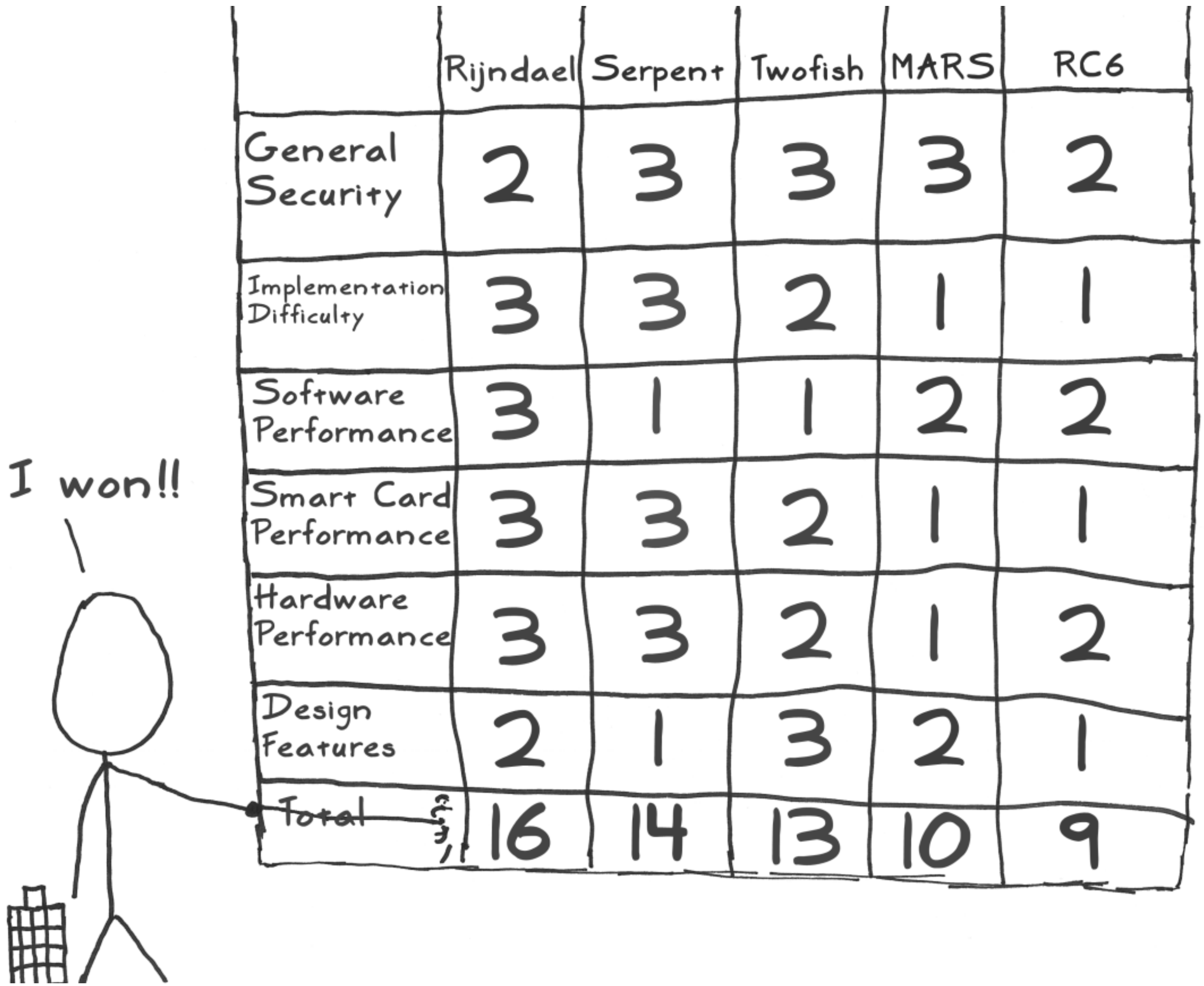
- Goals
  - More secure than 3DES
  - More efficient than 3DES
  - Support different key lengths



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

Why AES and None of the Others?



	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
~~Total~~	16	14	13	10	9

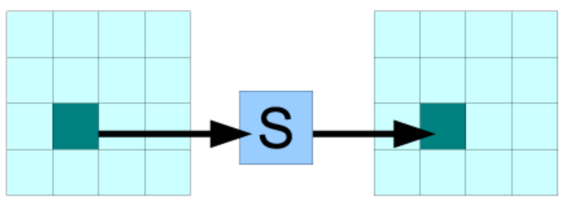
Structure of AES

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

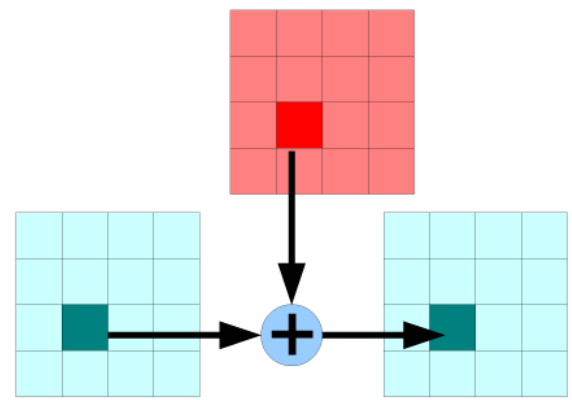
$$
\begin{array}{|c|c|c|c|}\hline i_0 & i_4 & i_8 & i_{12} \\ \hline i_1 & i_5 & i_9 & i_{13} \\ \hline i_2 & i_6 & i_{10} & i_{14} \\ \hline i_3 & i_7 & i_{11} & i_{15} \\ \hline \end{array}
\dashrightarrow
\begin{array}{|c|c|c|c|}\hline s_{00} & s_{01} & s_{02} & s_{03} \\ \hline s_{10} & s_{11} & s_{12} & s_{13} \\ \hline s_{20} & s_{21} & s_{22} & s_{23} \\ \hline s_{30} & s_{31} & s_{32} & s_{33} \\ \hline \end{array}
\dashrightarrow
\begin{array}{|c|c|c|c|}\hline o_0 & o_4 & o_8 & o_{12} \\ \hline o_1 & o_5 & o_9 & o_{13} \\ \hline o_2 & o_6 & o_{10} & o_{14} \\ \hline o_3 & o_7 & o_{11} & o_{15} \\ \hline \end{array}
$$

Operations used in each round

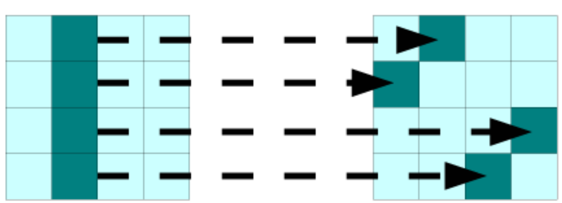
Byte Substitution (SB)



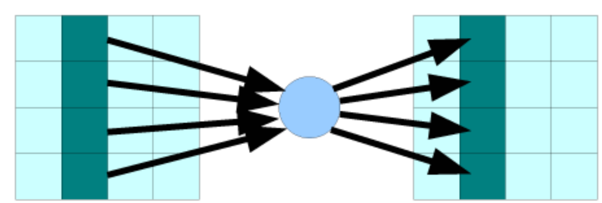
Key Addition (KA)



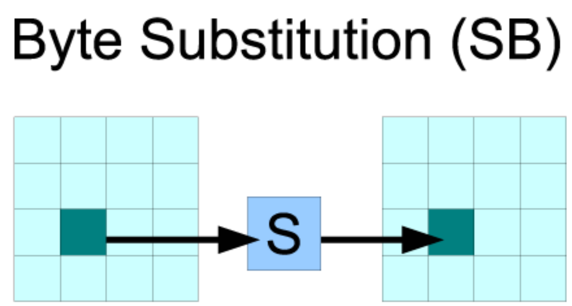
Shift Row (SR)



Mix Column (MC)



Byte Substitution

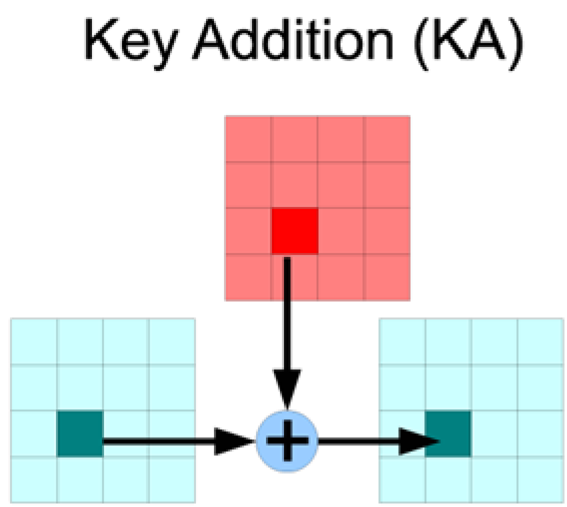


- Each byte in the current state is replaced by an entry from the 16x16 S-Box depicted on the next slide
- First four bit indicate the column, last four bits indicate row to pick

S-Box used for SB

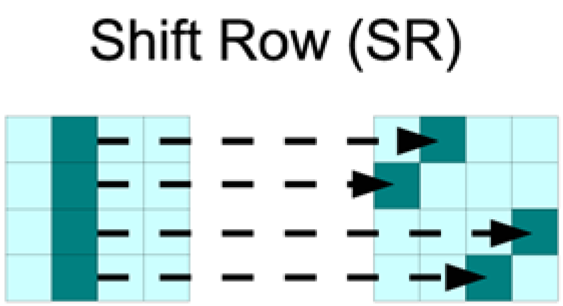
99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Key Addition



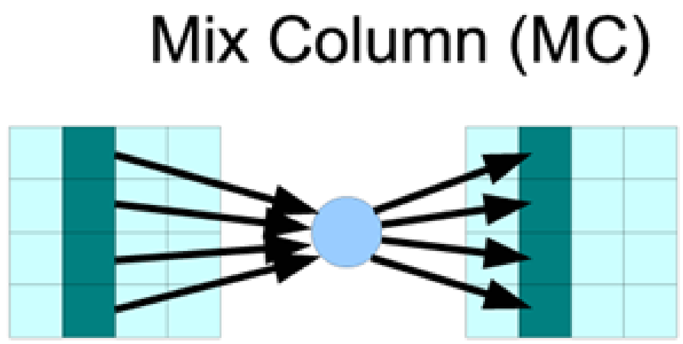
- A 128 bit round key is added to the current state matrix
- i.e. each byte in the state matrix is xored with the corresponding byte of the round key

Shift Row



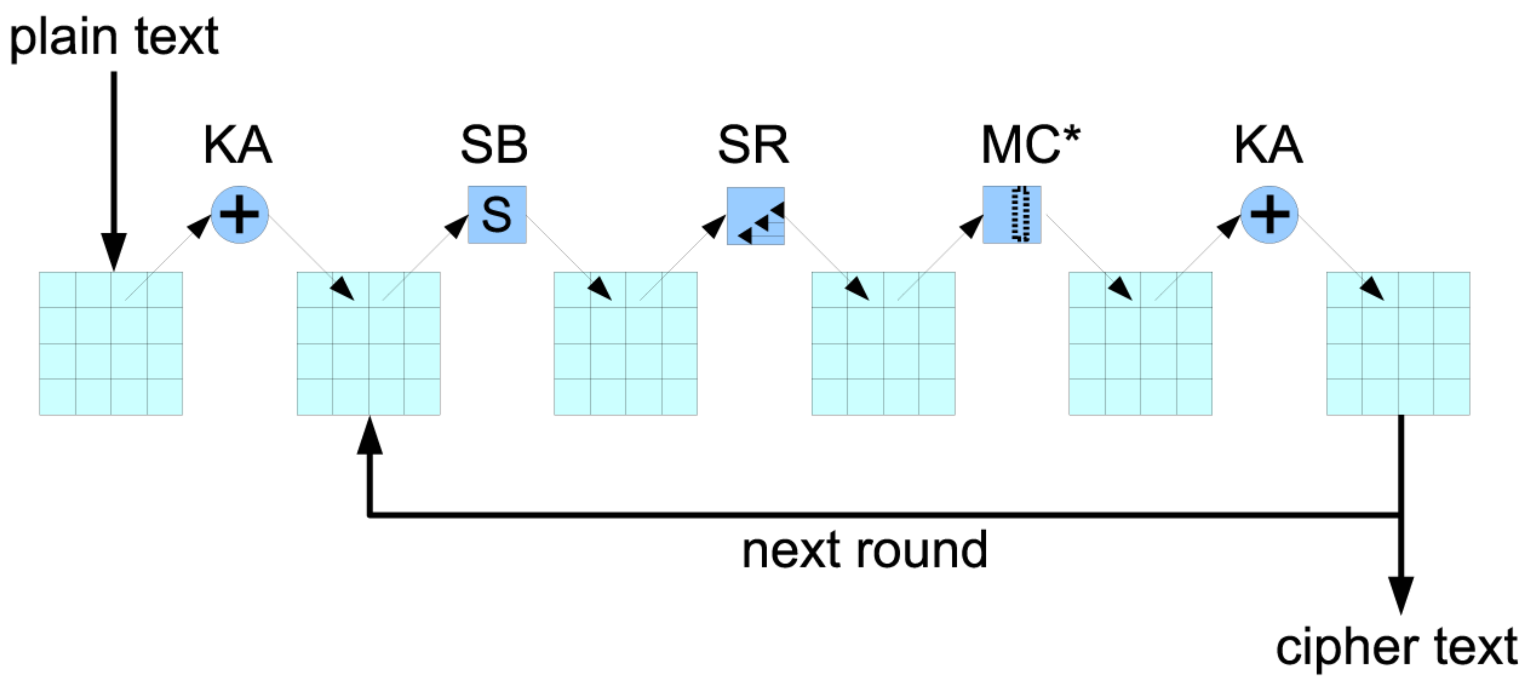
- Each row but the first one is cyclicly shifted to the left
- The second row is shifted by one byte
- The third row is shifted by two bytes
- The fourth row is shifted by three bytes

Mix Column



- Multiply matrix from the left with a fixed matrix

Putting it all together



- 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

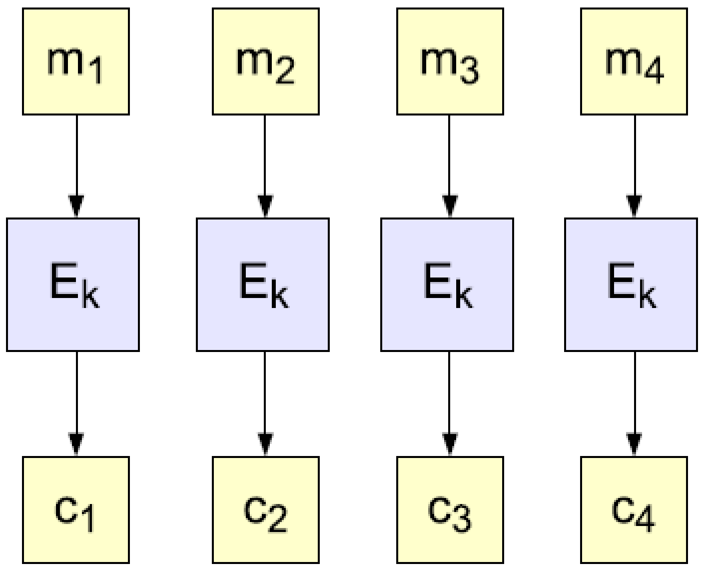
Recent Attacks Against AES

- 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}
    - 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
- August 2011: Bogdanov et al.
  - Known-plaintext attack on AES-128, AES-192, and AES-256 with time complexity of $2^{126.2}$ $2^{189.7}$ $2^{254.4}$
- June 2015: Tao et al
  - Improvement of the prior attacks to AES-128 $2^{126.01}$, AES-192 $2^{189.91}$, AES-256 with time complexity $2^{254.2}$
- No reason to worry yet
  - No practical attacks against full round AES-128, AES-256, AES-192

Encrypting a Large Message

- 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: $c_i = E_k(m_i)$

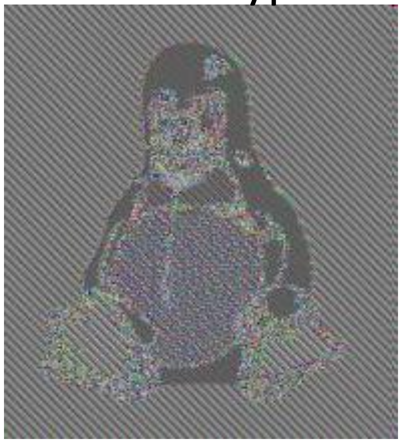
Decryption: $m_i = D_k(c_i)$

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

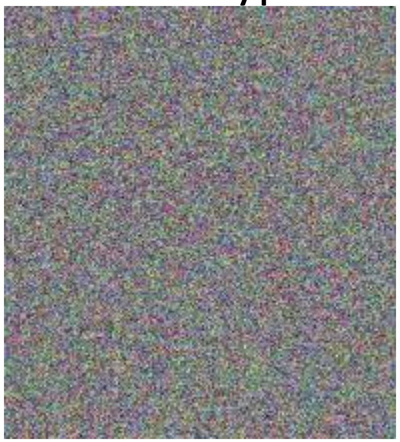
Why ECB is Not Enough

Plaintext

ECB-encrypted

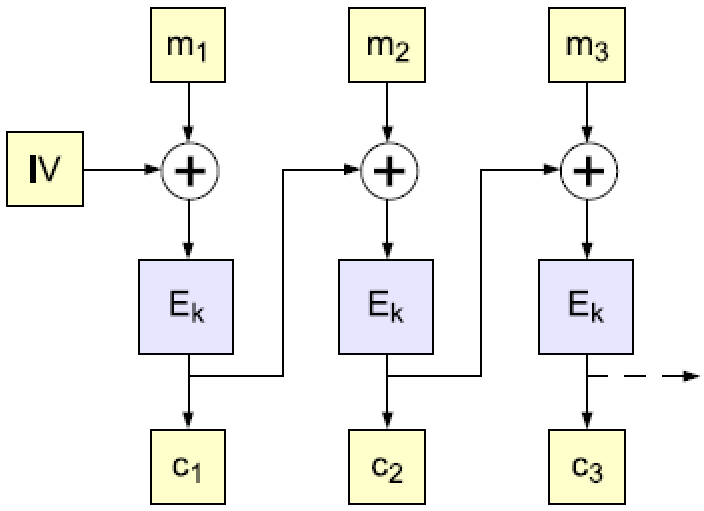


CBC-encrypted



- 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
  - Due to the fact that same plaintext block always leads to the same cipher block

Cipher Block Chaining Mode



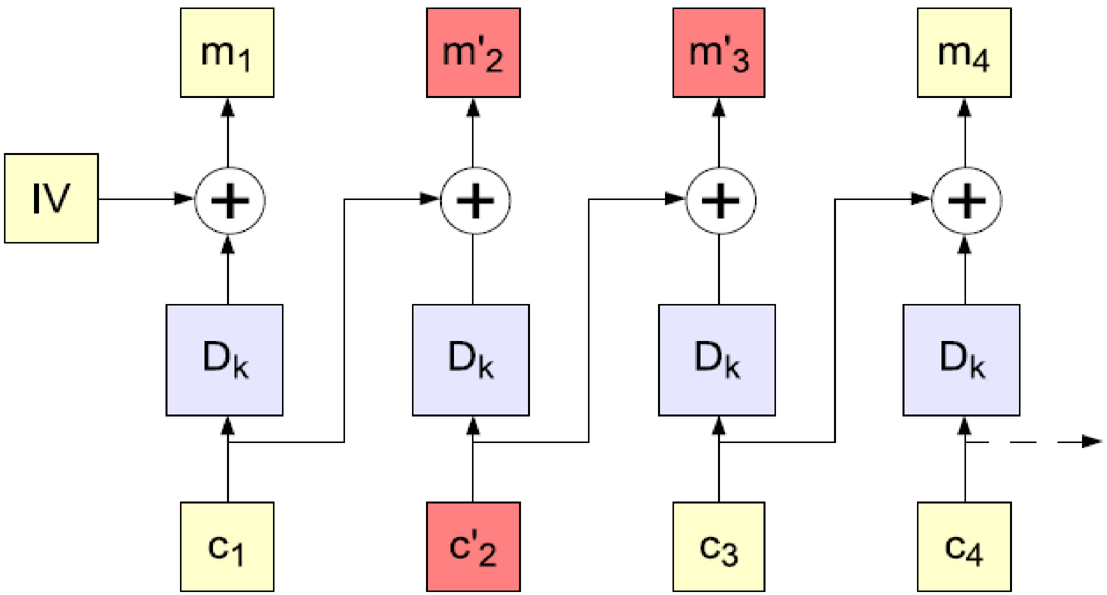
IV : = c_0

Encryption: $c_i = E_k(m_i \oplus c_{i-1})$

Decryption: $m_i = D_k(c_i) \oplus c_{i-1}$

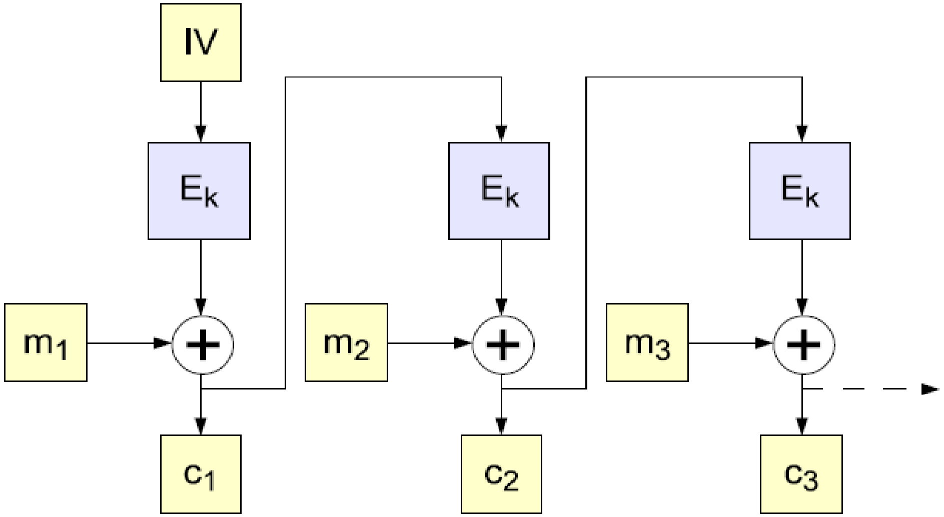
- If a new IV is used with each message to encrypt, messages starting with the same plaintext block do not start with the same ciphertext block
- Advantages
  - Deletion of a ciphertext block can be detected
  - Re-ordering of ciphertext blocks can be detected
  - Self-synchronizing on transmission errors

Self-Synchronization Property of CBC



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

Cipher Feedback Mode (CFB) - Simplified



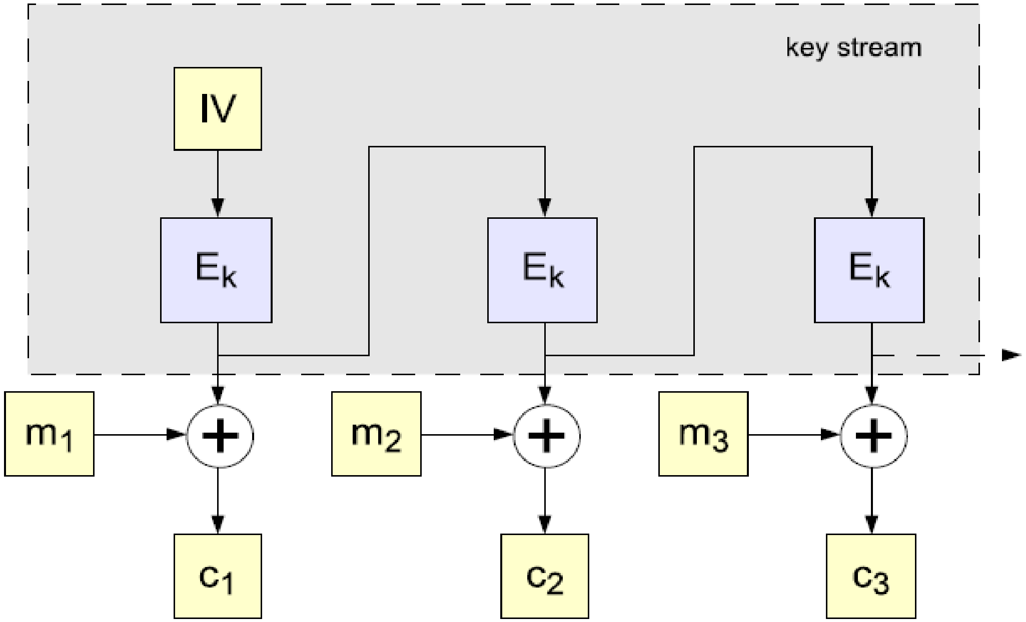
IV public, IV : = c_0

Encryption: $c_i = E_k(c_{i-1}) \oplus m_i$

Decryption: $m_i = c_i \oplus E_k(c_{i-1})$

- Generates a key stream that depends on the ciphertext

Output Feedback Mode (OFB) -Simplified



IV public

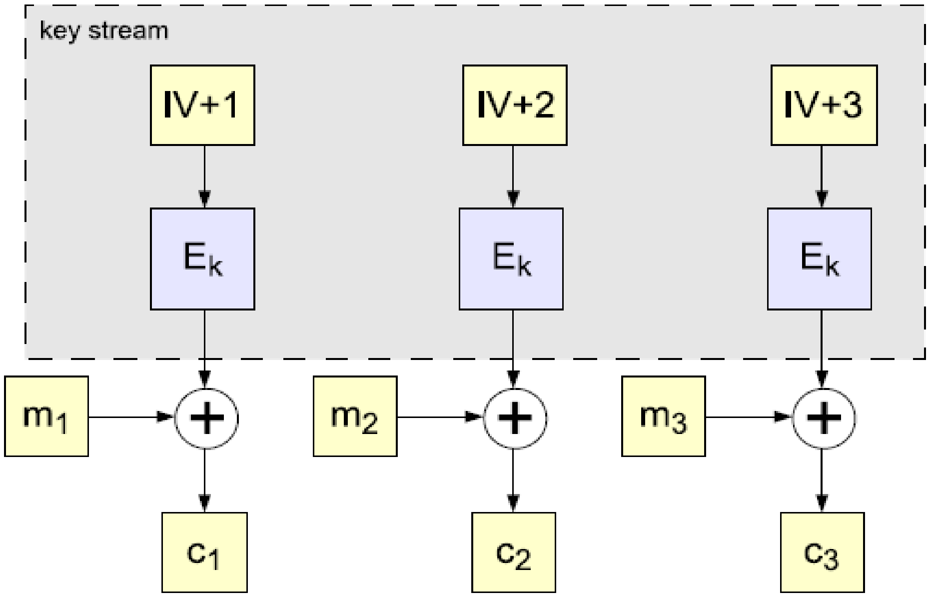
Encryption: $c_i = E_k^i(\ IV) \oplus m_i$

$E_k^i(\ IV)$ = IV encrypted i-times

Decryption: $m_i = c_i \oplus E_k^i(\ IV)$

- Generates a key stream that does not depend on the plaintext
- 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: $c_i = E_k(\text{ IV}+i) \oplus m_i$

Decryption: $m_i = c_i \oplus E_k(\text{ IV}+i)$

- 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, and CTR
  - Not restricted to complete blocks
  - Turn a block cipher into a stream cipher to some extend
    - Plaintext is xored with key stream bits, key stream depends on IV, Counter
- ECB, CBC
  - Require padding to complete blocks
  - Padding has to be easy to strip-off

Stream Ciphers

- Remember the one-time pad?
  - $E_K(M) = M \oplus Key$
  - 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 10^6-bit pseudo-random sequence
- $E_K(M) = IV, M \oplus PRNG(IV,K)$
  - Message processed bit by bit, not in blocks

Examples for Stream Ciphers

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

Properties of Stream Ciphers

- Typically 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, …b_i, then b_{i+1} is "0" with probability ½, "1" with probability ½

Weaknesses of Stream Ciphers

- No integrity
  - Associativity & commutativity: $(X \oplus Y \oplus Z=(X \oplus Z) \oplus Y$
  - $(M1 \oplus PRNG(seed)) \oplus M2 = (M1 \oplus M2) \oplus PRNG(seed)$
- Known-plaintext attack is very dangerous if keystream is ever repeated
  - Self-cancellation property of XOR: $X \oplus X=0$
  - $(M1 \oplus PRNG(seed)) \oplus (M2 \oplus PRNG(seed)) = M1 \oplus M2$
  - 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 $M1 \oplus M2$

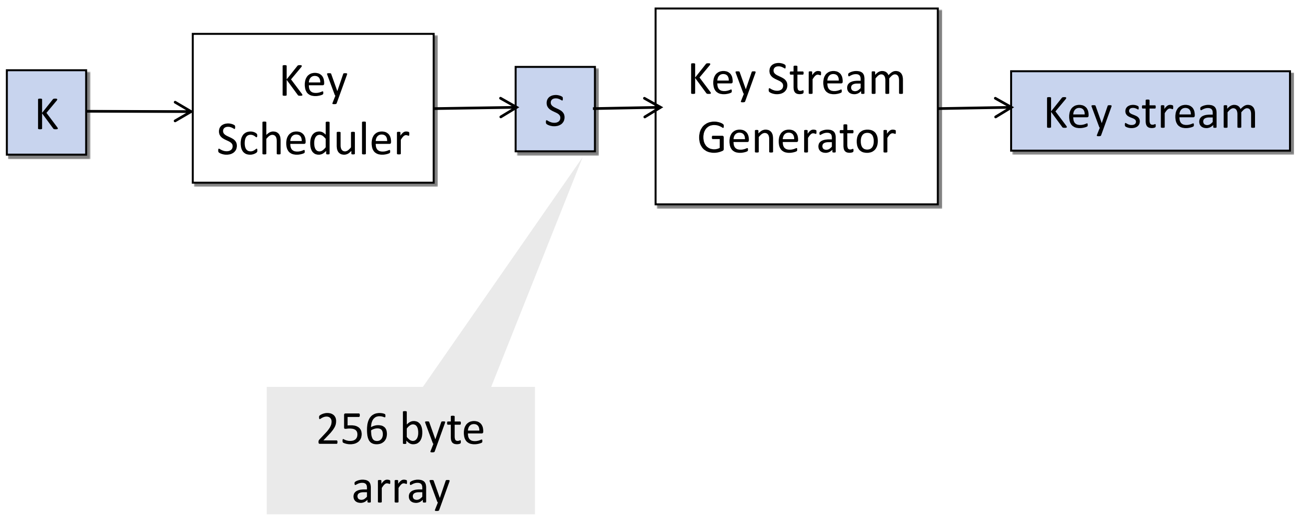
Stream Cipher Terminology

- 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 PRNG(IV,key) is referred to as keystream
  - PRNG must be cryptographically secure
- Encrypt message by XORing with keystream
  - ciphertext = message $\oplus$ keystream

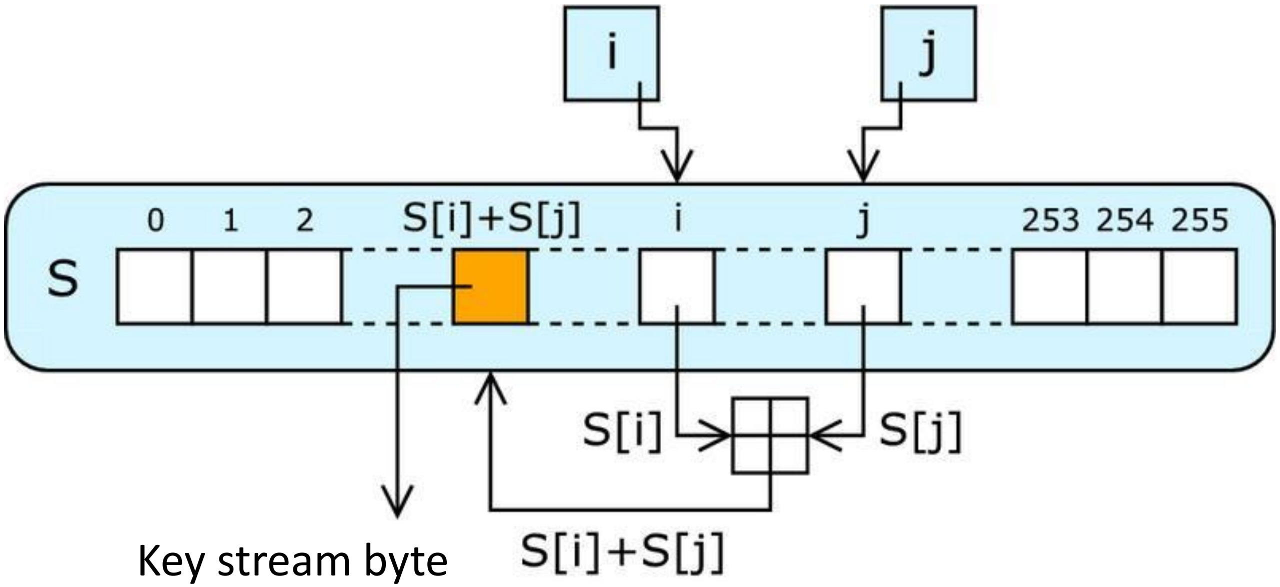
RC4

- 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 is 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

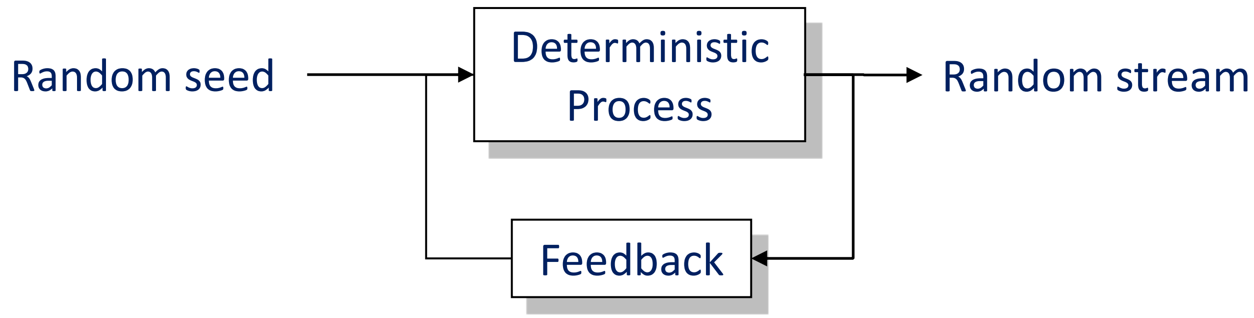
- To use RC4, usually prepend initialization vector (IV) to the key
  - IV can be random or a counter
- RC4 is not random enough! 1st byte of generated sequence depends only on 3 cells of state array S. This can be used to extract the key.
  - To use RC4 securely, RSA suggests discarding the first 256 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 ½
- 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

Reading

- 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