CS306: Introduction to IT Security

Fall 2018

Lecture 3: Symmetric encryption II

Instructor: **Nikos Triandopoulos**

September 11, 2018

STEVENS
INSTITUTE *of* TECHNOLOGY
THE INNOVATION UNIVERSITY®

Last week

- Symmetric encryption
  - classical symmetric-key ciphers
  - definition of perfect secrecy
  - the one-time pad

Today

- Symmetric encryption
  - introduction to modern cryptography
  - computational security
  - symmetric encryption in practice
  - modes of operation

3.0 Announcements

CS306: Announcements

- TA hours start this week

Monday	Tuesday	Wednesday	Thursday	Friday	
1-2pm	4-6pm	3-4pm	4-5pm	11am-12pm	3-4pm
Janiece	Instructor	Matt	Taylor	Rushabh	Samaneh
NB 327	NB 321	NB 327	NB 327	NB 327	NB 314

- course add/drop deadline has passed
  - contact me if you need a permanent lab/lecture section change
- HW 1 to go out on Friday

CS306: Lectures & labs

CS306 is offered in **2 required sessions**, each offered in **multiple sections**

- lectures
  - *Skip lunch* CS306-A, Tue 12:00pm - 14:30pm, Kidde 360 10/60 (50)
  - *Skip dinner* CS306-B, Tue 18:15pm - 20:45pm, Babbio 104 7/60 (53)

103

- labs
  - *Stay focused* CS306-Lx, sometime & somewhere on Thursdays

x	B	C	D	E	F
time	9:25 - 10:15	10:50 - 11:40	12:15 - 13:05	13:40 - 14:30	15:05 - 15:55
place	Morton 105	Kidde 360	Babbio 220	Babbio 304	Burchard 514
capacity	10/25 (15)	closed (25)	4/25 (21)	5/25 (20)	4/25 (21)

102

PLEASE contact me if you have have not enrolled to any lab section!

CS306: Tentative Syllabus

Week	Date	Topics	Reading	Assignment
1	Aug 28	Introduction	Ch. 1	-
2	Sep 4	Symmetric encryption	Ch. 2 & 12	Lab 1
3	**Sep 11**	**Symmetric encryption II**	**Ch. 2 & 12**	**Lab 2, HW 1**
4	Sep 18	Public-key crypto I		
5	Sep 25	Public-key crypto II		
6	Oct 2	Access control & authentication		
–	Oct 9	**No class** (Monday schedule)		
7	Oct 16	**Midterm** (closed books)	All materials covered	

CS306: Tentative Syllabus (continued)

Week	Date	Topics	Reading	Assignment
8	Oct 23	Software & Web security		
9	Oct 30	Network security		
10	Nov 6	Database security		
11	Nov 13	Cloud security		
12	Nov 20	Privacy		
13	Nov 27	Economics		
14	Dec 4	Legal & ethical issues		
15	Dec 11 (or later)	**Final** (closed books)	All materials covered*	

* w/ focus on what covered after midterm

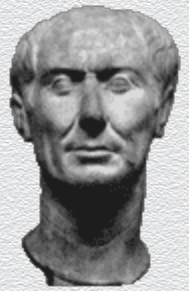
CS306: Course outcomes

- **Terms**
  - describe common security terms and concepts
- **Cryptography**
  - state basics/fundamentals about secret and public key cryptography concepts
- **Attack & Defense**
  - acquire basic understanding for attack techniques and defense mechanisms
- Impact
  - acquire an understanding for the broader impact of security and its integral connection to other fields in computer science (such as software engineering, databases, operating systems) as well as other disciplines including STEM, economics, and law
- Ethics
  - acquire an understanding for ethical issues in cyber-security

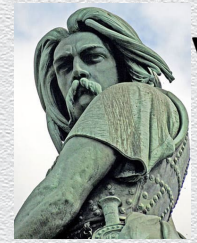
Recall: Perfect Vs. imperfect encryption

Just before a battle more than 2000 years ago...

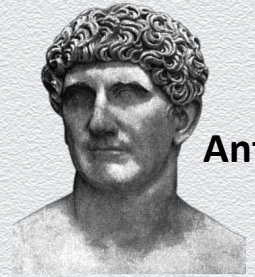
Caesar



$Enc_k(m) \rightarrow c$



Vercingetorix



Antonius

m = `impetus!` or `receptum`

m = { `attack` w/ prob. 0.8 ; `no attack` w/ prob. 0.2

If c = `lpshwxvc`, then history is not as we know it

If c = `rreeccee`, then enemy's view remains unchanged

Alas, one-time pads are impractical (or else insecure!)

Discussion: Lab 1

- rekeying
  - good security hygiene, generally; “vulnerability” in Quiz 1.1
- key reuse in OPT
  - attacks go beyond learning the XOR of two messages
- definition of perfect secrecy
  - independent of message distribution & cardinality of message space
  - independent of uniform key selection
- OTP & brute force attack
- (additive) secret sharing

Questions?

3.1 Introduction to modern cryptography

Cryptography / cryptology

- Etymology
  - two parts: “crypto” + “graphy” / “logy”
  - original meaning: κρυπτός + γράφω / λόγος (**in Greek**)
  - English translation: secret + write / speech, logic
  - meaning: secret writing / the study of secrets
- Historically developed/studied for secrecy in communications
  - i.e., message encryption in the symmetric-key setting
  - main application area: use by military and governments

Classical Vs. modern cryptography

antiquity – ~70s

*"the **art** of **writing** and **solving codes**"*

- approach
  - ad-hoc design
  - trial & error methods
  - empirically evaluated

~80s – today

*"the study of **mathematical techniques** for **securing** information, systems, and distributed computations against **adversarial attacks**"*

- approach
  - systematic design & analysis
  - formal notions of security / adversary
  - rigorous proofs of security (or insecurity)

Classical Vs. modern cryptography: Encryption

antiquity – ~70s

*"the **art** of **writing** and **solving codes**"*

- **ad-hoc study**
  - vulnerabilities/insecurity of
    - Caesar's cipher
    - shift cipher
    - mono-alphabetic substitution cipher

~80s – today

*"the study of **mathematical techniques** for **securing** information, systems, and distributed computations against **adversarial attacks**"*

- **rigorous study**
  - problem statement
    - secret communication over insecure channel
  - abstract solution concept
    - symmetric encryption, Kerckhoff's principle, perfect secrecy
  - concrete solution & analysis
    - OTP cipher, proof of security, Shannon's Theorem

Formal treatment in modern cryptography

Problem is formulated as an abstract crypto primitive

- captures the essence of the problem at hand, provides clarity and focus

Design & evaluation of crypto primitives follows a systematic process

- (A) **formal definitions** (what it means for a crypto primitive to be secure?)
- (B) **precise assumptions** (which forms of attacks are allowed – and which aren't?)
- (C) **provable security** (why a candidate solution is secure – or not)?

(A) Formal definitions

abstract but rigorous description of security problem

- **computing setting** (to be considered)
  - involved parties, communication model, core functionality
- **underlying cryptographic scheme** (to be designed)
  - e.g., symmetric-key encryption scheme
- **desired properties** (to be achieved)
  - security related
  - non-security related
    - e.g., correctness, efficiency, etc.

(A) Why formal definitions are important?

- **successful project management**
  - good design requires clear/specific security goals
    - helps to avoid critical omissions or over engineering
- **provable security**
  - rigorous evaluation requires a security definition
    - helps to separate secure from insecure solutions
- **qualitative analysis/modular design**
  - thorough comparison requires an exact reference
    - helps to secure complex computing systems

(B) Precise assumptions

precise description of all relevant problem components

- **adversary**
  - type of attacks – a.k.a. **threat model**
  - **capabilities** (e.g., a priori knowledge, access to information, party corruptions)
  - **limitations** (e.g., bounded memory, passive Vs. active)
- **computational assumptions** (about hardness of certain tasks)
  - e.g., factoring of large composite numbers is hard
- **computing setting**
  - system set up, initial state, key distribution, randomness…
  - means of communication (e.g., channels, rounds, …)
  - timing assumptions (e.g., synchronicity, epochs, …)

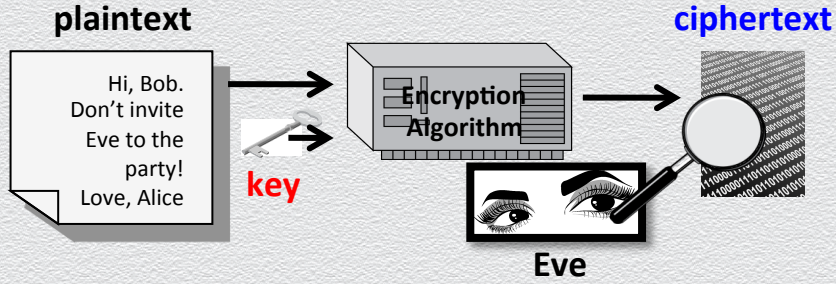
(B) Why precise assumptions are important?

- **basis** for proofs of security
  - security holds under specific assumptions
- **comparison** among possible solutions
  - relations among different assumptions
    - stronger/weaker (i.e., less/more plausible to hold), “A implies B” or “A and B are equivalent”
    - refutable Vs. non-refutable
- **flexibility** (in design & analysis)
  - **validation** – to gain confidence or refute
  - **modularity** – to choose among concrete schemes that satisfy the same assumptions
  - **characterization** – to identify simplest/minimal/necessary assumptions

Possible eavesdropping attacks (I)

An attacker may possess a

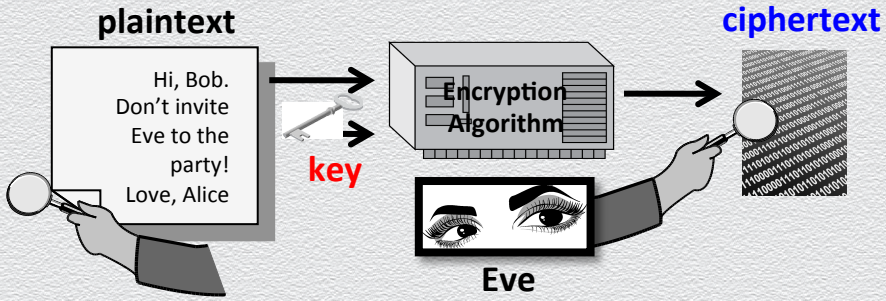
- **(a) collection of ciphertexts**
  - ciphertext only attack (or simply EAV)
    - this will be the **default attack type**



Possible eavesdropping attacks (II)

An attacker may possess a

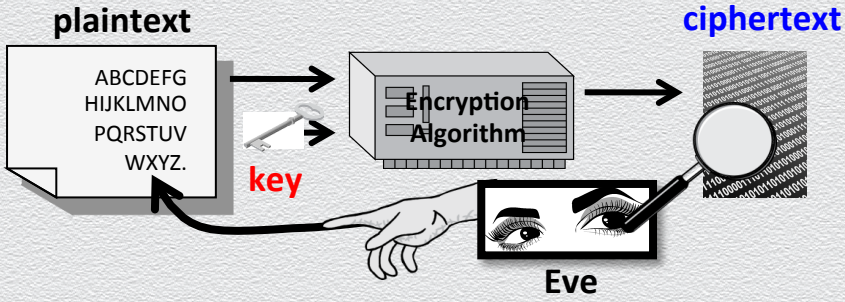
- (a) collection of ciphertexts
  - ciphertext only attack
- **(b) collection of plaintext/ciphertext pairs**
  - known plaintext attack



Possible eavesdropping attacks (III)

An attacker may possess a

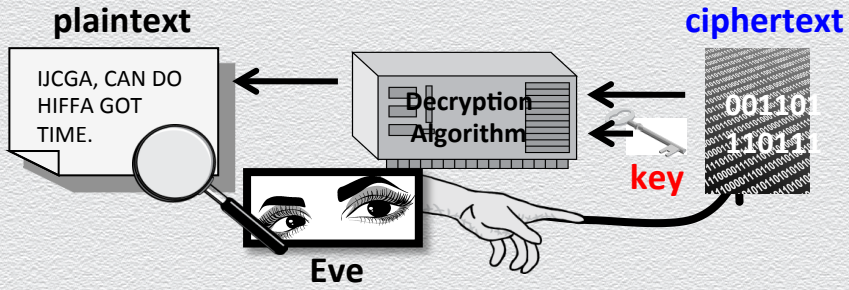
- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- **(c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker**
  - chosen plaintext attack (CPA)



Possible eavesdropping attacks (IV)

An attacker may possess a

- (a) collection of ciphertexts
  - ciphertext only attack
- (b) collection of plaintext/ciphertext pairs
  - known plaintext attack
- (c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker
  - chosen plaintext attack (CPA)
- **(d) collection of plaintext/ciphertext pairs for (plaintexts and) ciphertexts selected by the attacker**
  - chosen ciphertext attack (CCA)



(C) Why provable security is important?

Typical performance

- in some areas of computer science **formal proofs may not be essential**
  - e.g., the behavior of a hard-to-analyze algorithm can still be simulated by experimentally studying its performance on "typical" inputs
- in practice, **typical inputs** are expected to occur

Worst case performance

- in cryptography and secure protocol design **formal proofs are essential**
  - "experimental" security analysis is not possible
  - the notion of a "typical" adversary makes little sense and is unrealistic
  - in practice, **worst case attacks will occur**
    - an adversary will use any means in its power to break a scheme

Discussion: A more careful comparison of ciphers

Caesar's/shift/mono-alphabetic cipher

- substitution ciphers
  - Caesar's cipher
    - **shift is always 3**
  - shift cipher
    - **shift is unknown and the same for all characters**
  - mono-alphabetic substitution cipher
    - **shift is unknown and the same for all character occurrences**

The one-time pad

- also, a substitution cipher
  - **shift is unknown and independent for each character occurrence**

Classical ciphers – examples (IV)

The Vigenère cipher (or poly-alphabetic shift cipher)

- **generalization** of mono-alphabetic substitution cipher
- key space defines **fixed shift pattern** that is applied on **blocks of characters**
  - a key k is a sequence of t shift values applied on blocks of size t
    - if "period" t is 4, and k = (2, 1, 3, 11), then each block of 4 characters in the plaintext are shifted respectively by 2, 1, 3 and 11 positions
    - the plaintext-to-ciphertext mapping is **many-to-many**
      - depending on block-location (in plaintext) and character-location (in block)

Cryptanalysis of Vigenère cipher – case I

If period t is **known**

- problem is **reduced to attacking the shift cipher**
  - consider **t-streams**, subsequences of ciphertext characters that are t-positions apart
    - i.e., subsequences of the form $c_j, c_{j+t}, c_{j+2t}, \ldots$, where j is in [1:t]
  - no text will "make sense" + brute-force attacks may be infeasible (of the order of 26^t)
  - yet, attacks based on statistics (e.g., frequencies or "squares" test) can be used

Cryptanalysis of Vigenère cipher – case II

If period t is **unknown**

- repeat above attack for guessed values of t (if an upper bound T is known); or find t
- Kasiski's method
  - identify repeated patterns of length 2 or 3 in the ciphertext
  - they're likely to correspond to common bigrams/trigrams in the plaintext (e.g., "the")
  - period t can be deduced by locations of these patterns in the text; or
- index of coincidence method (using character frequencies over t-streams)
  - when character i is shifted by j, we expect $p_i \approx q_{i+j}$ and, thus, $\sum_i p_i^2 \approx \sum_i p_i^2 \approx 0.065$
  - compute $S_\tau = \sum_i q_i^2$ and stop when $S_\tau \approx 0.065$; then τ is (a multiple of) t

3.2 Computational security

The big picture

- we formally defined and constructed a perfectly secure cipher
- this encryption has some drawbacks
  - e.g., it employs a very large key
- by Shannon's Theorem, such limitations are unavoidable

Now, what?

Our approach: Relax “perfectness”

Initial model

- the **perfect secrecy** (or security) requires that
  - the ciphertext **leaks absolutely no extra information** about the plaintext
  - to adversaries of **unlimited computational power**

Refined model

- a relaxed notion of security, called **computational security**, requires that
  - the ciphertext leaks **a tiny amount of extra information** about the plaintext
  - to adversaries with **bounded computational power**

Computational security

- to be contrasted against information-theoretic security
  - *de facto* way to model security in most settings
  - an integral part of modern cryptography w/ rigorous mathematical proofs
- entails two relaxations
  - security is guaranteed against **efficient** adversaries
    - if an attacker invests in **sufficiently large resources**, it may break security
    - goal: make required resources larger than those available to any realistic attacker!
  - security is guaranteed in a **probabilistic** manner
    - with some **small probability**, an attacker may break security
    - goal: make attack probability sufficiently small so that it can be practically ignored!

Towards a rigorous definition of computational security

Concrete approach

- bounds the maximum success probability of any (randomized) adversary running for some specified amount of time or investing a specified amount of resources
- *"A scheme is (t,ε)-secure if any adversary $\mathcal{A}$, running for time at most t, succeeds in breaking the scheme with probability at most ε"*
- need to
  - define what it means for an adversary to "break" a scheme
  - specify precisely the resources
    - e.g., time in seconds using a particular computer,
      or CPU cycles in a particular available supercomputer architecture

Examples

- almost optimal security guarantees
  - key length n, key space size $|\mathcal{K}| = 2^n$
  - $\mathcal{A}$ running for time t (e.g., CPU cycles) succeeds w/ prob. at most $ct/2^n$
  - this corresponds to a brute-force type of attack (w/out preprocessing)
- parameter c models advanced computing methods
  - c is typically larger than 1 (e.g., parallelism can be used, etc.)
- if c = 1, n = 60, security is enough for attackers running a desktop computer
  - 4 GHz (4×10^9 cycles/sec), 2^{60} CPU cycles require about 9 years
  - however, "fastest" available computers run w/ 2×10^{16} cycles/sec, i.e., in ~1min!
    - choosing n = 80 is better: the supercomputer would still need ~2 years

Today's recommendations

- recommended security parameter is n = 128
- large difference between 2^{80} and 2^{128}
  - #seconds since Big Bang is ~2^{58}
- if probability of success (within 1 year of computation) is $1/2^{60}$
  - it is more likely that Alice and Bob are hit by lighting (and thus don't care much about the breached confidentiality)
  - an event happening once in 100 years corresponds to probability 2^{-30} of happening at a particular second
- limitations of the concrete approach
  - harder to achieve, careful interpretation is needed, what if $\mathcal{A}$ runs for 2t or t/2?

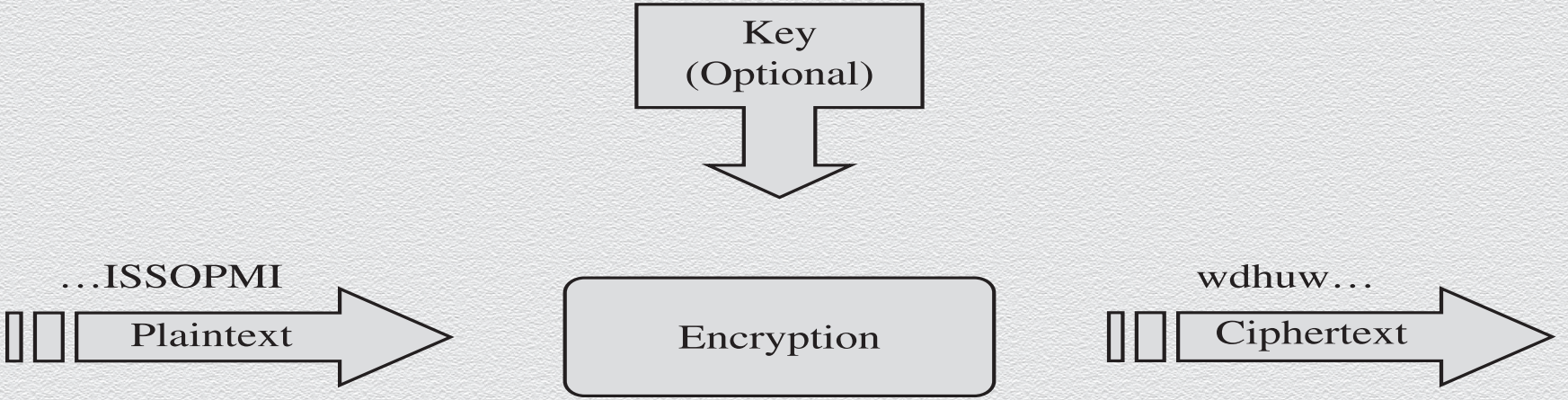
An alternative (less quantitative) approach

Asymptotic approach

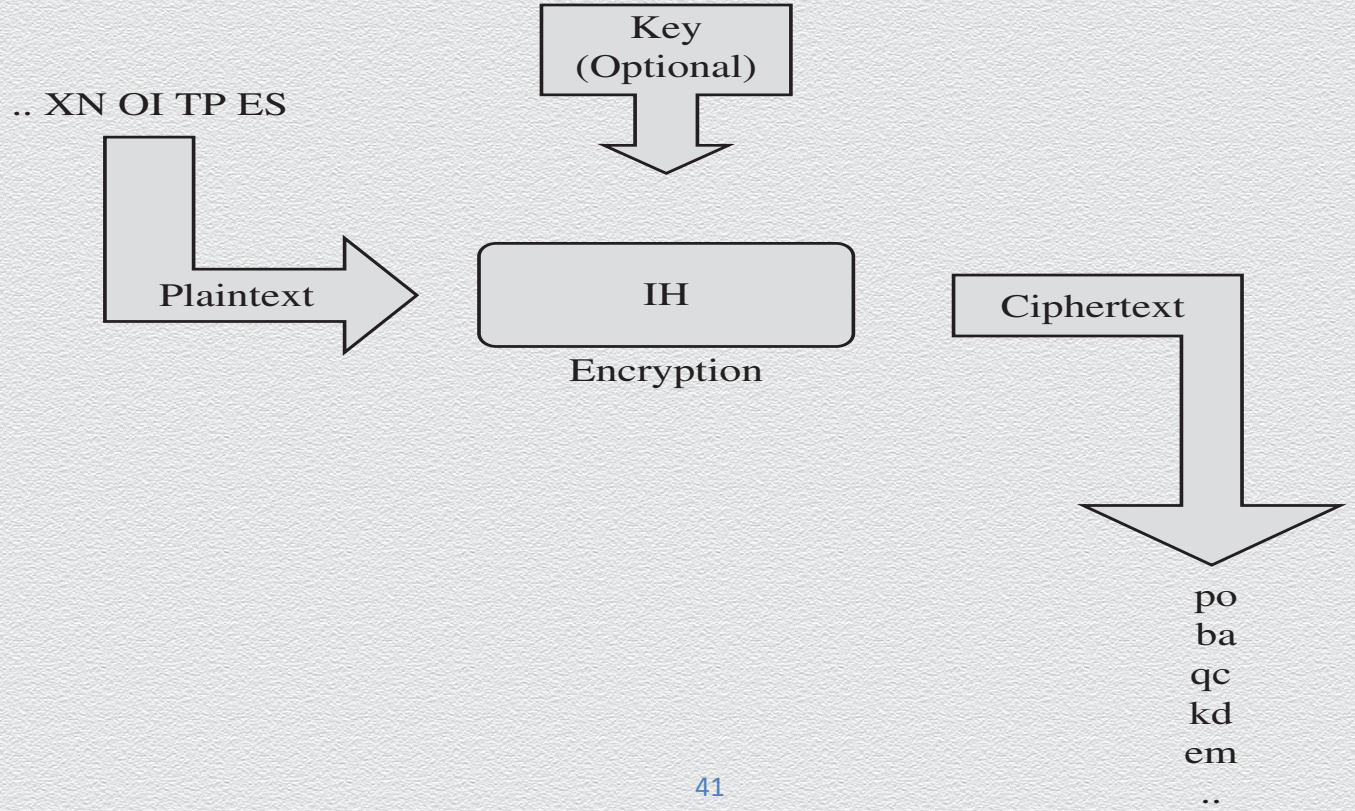
- a security parameter n is used; typically the key length
- **efficient** (or realistic or feasible) adversaries are equated with **probabilistic poly-time (PPT)** algorithms running for time that is a polynomial of n
- **small probability** of success is equated with **negligible** success probabilities, i.e., asymptotically smaller than any inverse polynomial in n
- *"A scheme is secure if any PPT adversary $\mathcal{A}$ succeeds in breaking the scheme with at most negligible probability"*

3.3 Symmetric encryption in practice

Stream ciphers



Block ciphers



A perfect encryption of a block

Goal: encrypt a block of n bits using the same key all the time

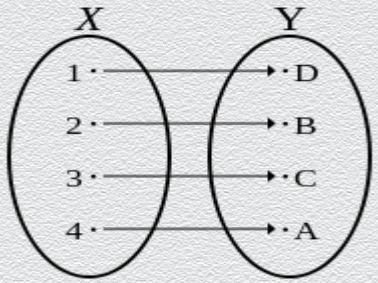
- but not have the problem of one-time pad

Approach: encryption via a bijective random mapping T from $\{0,1\}^n$ to $\{0,1\}^n$

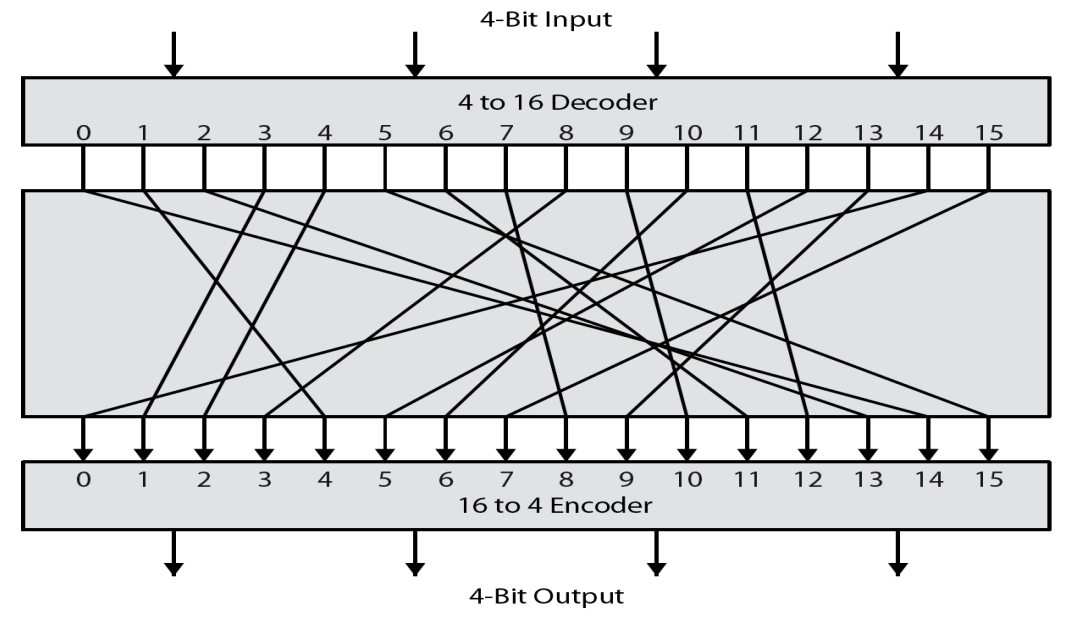
- mapped pairs are computed uniformly at random
- to encrypt x, just output T[x]
- to decrypt y, just output $T^{-1}[y]$
- the secret key is T

Problem with this approach: T has size ~ $n\,2^n$

a mapping T



Ideal block cipher



Techniques used in practice for symmetric encryption

- Substitution
  - exchanging one set of bits for another set
- Transposition
  - rearranging the order of the ciphertext bits
    - to break any regularities in the underlying plaintext
- Confusion
  - enforcing complex functional relationship between the plaintext/key pair & the ciphertext
    - e.g., flipping a bit in plaintext causes unpredictable changes to resulting ciphertext
- Diffusion
  - distributes information from single plaintext characters over entire ciphertext output
    - e.g., even small changes to plaintext result in broad changes to ciphertext

Substitution boxes

- substitution can also be done on binary numbers
- such substitutions are usually described by substitution boxes, or S-boxes

	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a)

	0	1	2	3
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

(b)

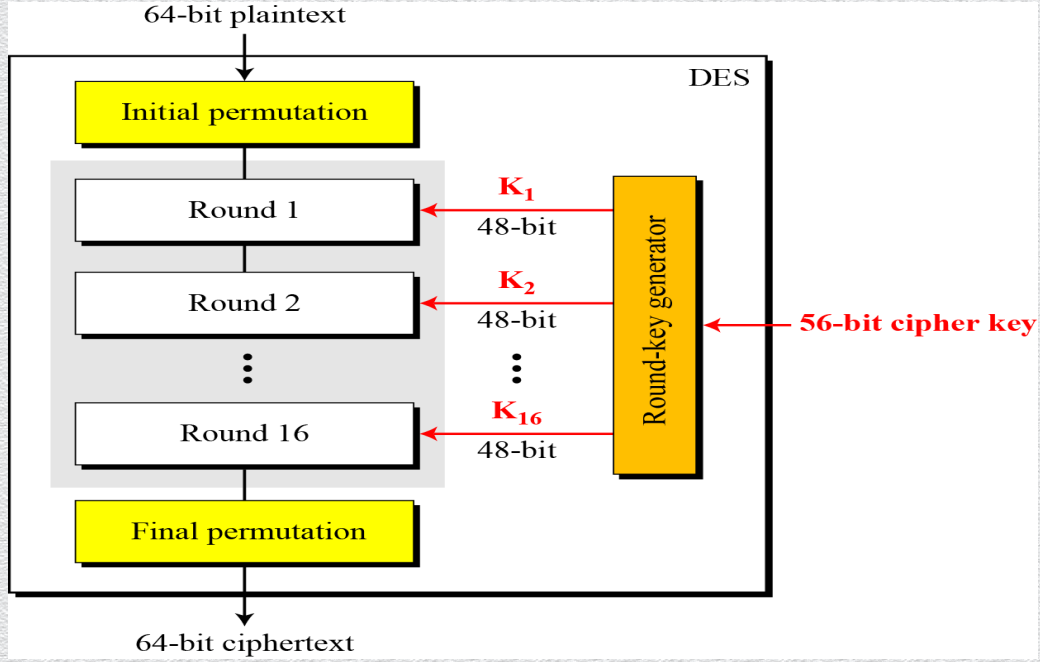
Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal.

Stream Vs. Block ciphers

	Stream	Block
Advantages	• Speed of transformation • Low error propagation	• High diffusion • Immunity to insertion of symbol
Disadvantages	• Low diffusion • Susceptibility to malicious insertions and modifications	• Slowness of encryption • Padding • Error propagation

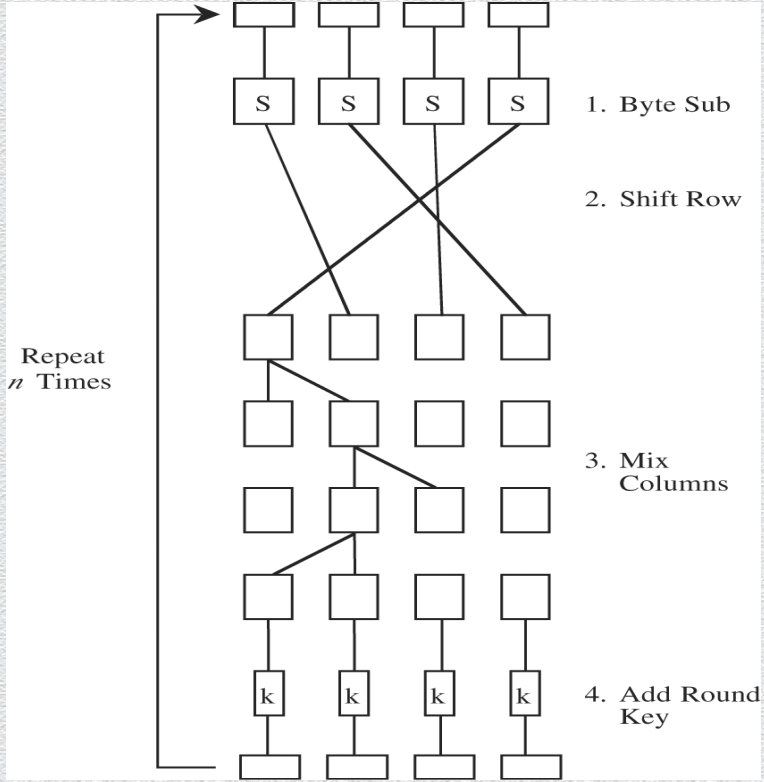
DES: The Data Encryption Standard

- symmetric block cipher
- developed in 1976 by IBM for the US National Institute of Standards & Technology (NIST)
- not commonly used
  - shouldn't be used today as it is considered insecure



AES: Advanced Encryption System

- symmetric block cipher
- developed in 1999
  by independent Dutch
  cryptographers
- still in common use



DES vs. AES

	DES	AES
Date designed	1976	1999
Block size	64 bits	128 bits
Key length	56 bits (effective length); up to 112 bits with multiple keys	128, 192, 256 (and possibly more) bits
Operations	16 rounds	10, 12, 14 (depending on key length); can be increased
Encryption primitives	Substitution, permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion, diffusion	Confusion, diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret, but open public comments and criticisms invited
Source	IBM, enhanced by NSA	Independent Dutch cryptographers

DES: The Data Encryption Standard

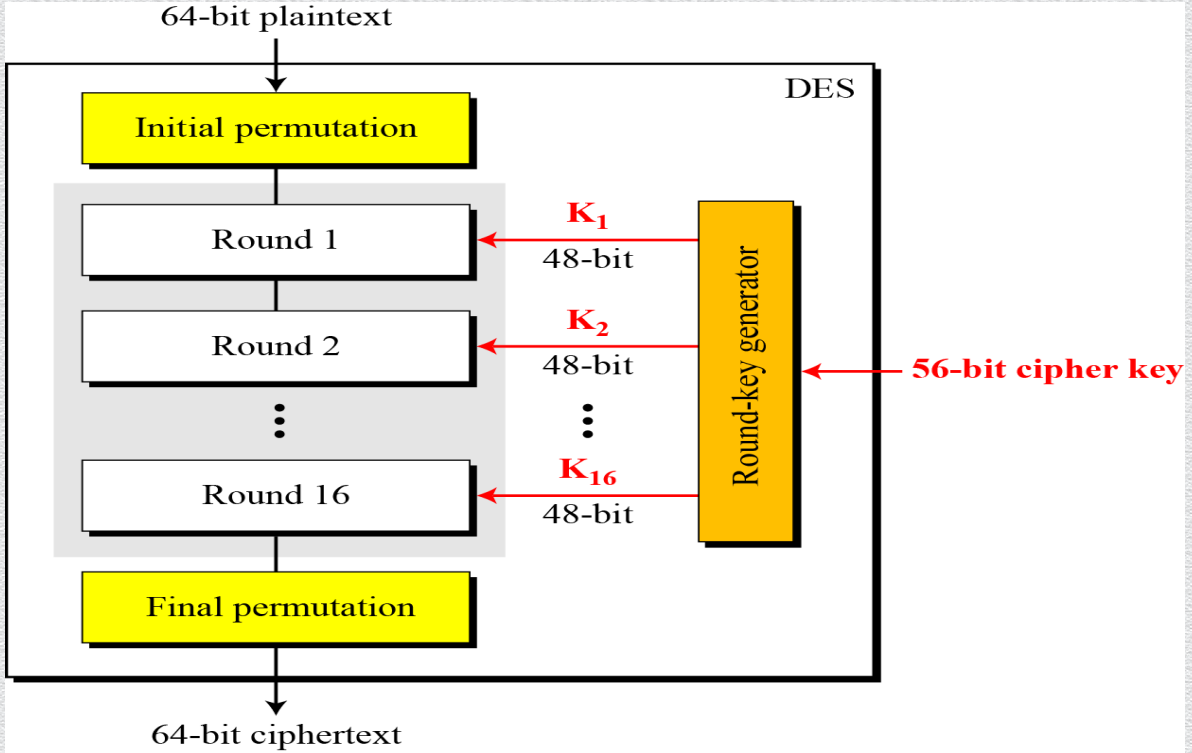
- employs substitution & transposition, on top of each other, for 16 rounds
  - block size = 64 bits, key size = 56 bits
- strengthening (since 56-bit security is not considered adequately strong)
  - double DES: $E(k_2, E(k_1, m))$, not effective!
  - triple DES: $E(k_3, \mathbf{E}(k_2, E(k_1, m)))$, more effective
    - two keys, i.e., $k_1=k_3$, with E-D-E pattern, 80-bit security
    - three keys with E-E-E pattern, 112-bit security

Security strength

Form	Operation	Properties	Strength
DES	Encrypt with one key	56-bit key	Inadequate for high-security applications by today's computing capabilities
Double DES	Encrypt with first key; then encrypt result with second key	Two 56-bit keys	Only doubles strength of 56-bit key version
Two-key triple DES	Encrypt with first key, then encrypt (or decrypt) result with second key, then encrypt result with first key (E-D-E)	Two 56-bit keys	Gives strength equivalent to about 80-bit key (about 16 million times as strong as 56-bit version)
Three-key triple DES	Encrypt with first key, then encrypt or decrypt result with second key, then encrypt result with third key (E-E-E)	Three 56-bit keys	Gives strength equivalent to about 112-bit key about 72 quintillion ($72*10^{15}$) times as strong as 56-bit version

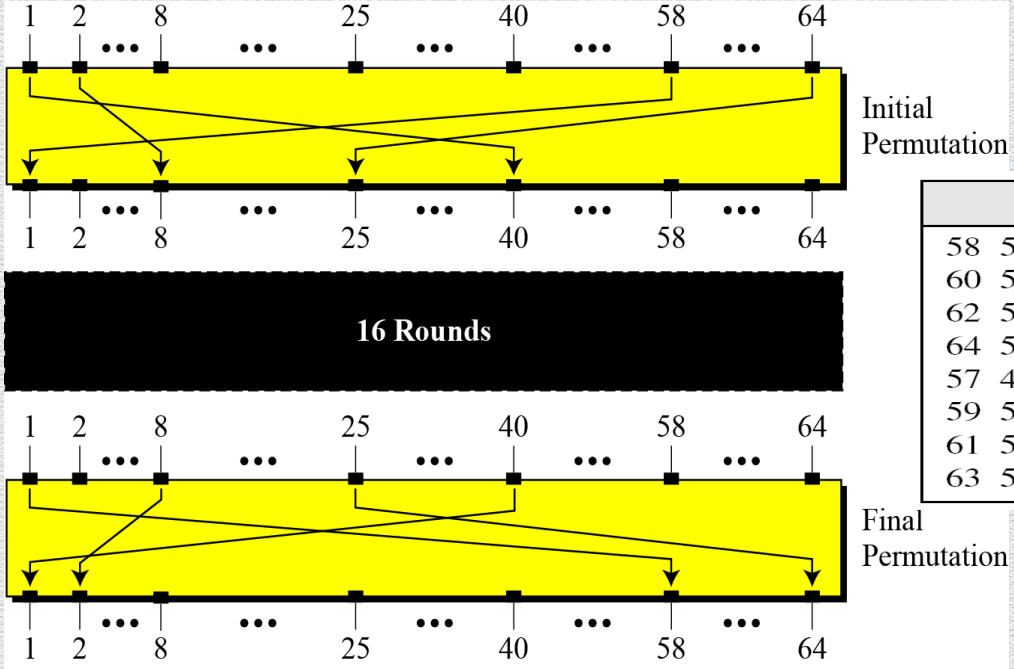
High-level DES view

DES structure



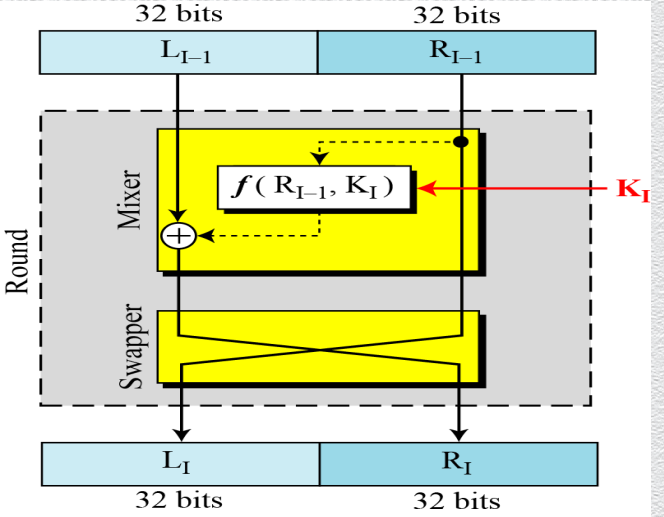
Initial and final permutations

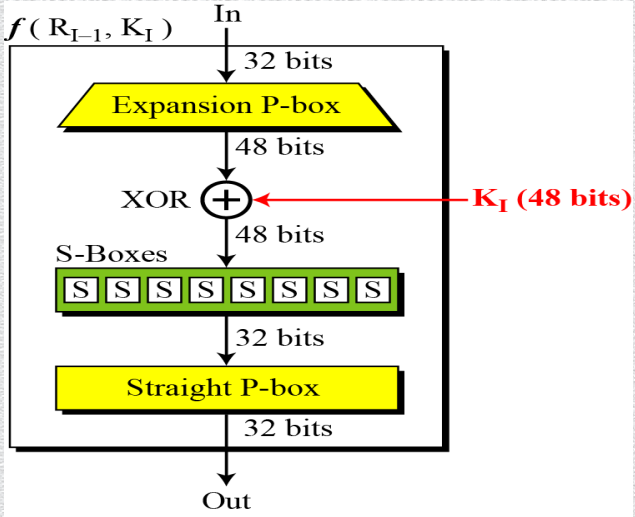
- Straight P-boxes that are inverses of each other w/out crypto significance



Initial Permutation								*Final Permutation*							
58	50	42	34	26	18	10	02	40	08	48	16	56	24	64	32
60	52	44	36	28	20	12	04	39	07	47	15	55	23	63	31
62	54	46	38	30	22	14	06	38	06	46	14	54	22	62	30
64	56	48	40	32	24	16	08	37	05	45	13	53	21	61	29
57	49	41	33	25	17	09	01	36	04	44	12	52	20	60	28
59	51	43	35	27	19	11	03	35	03	43	11	51	19	59	27
61	53	45	37	29	21	13	05	34	02	42	10	50	18	58	26
63	55	47	39	31	23	15	07	33	01	41	09	49	17	57	25

DES round: Feistel network

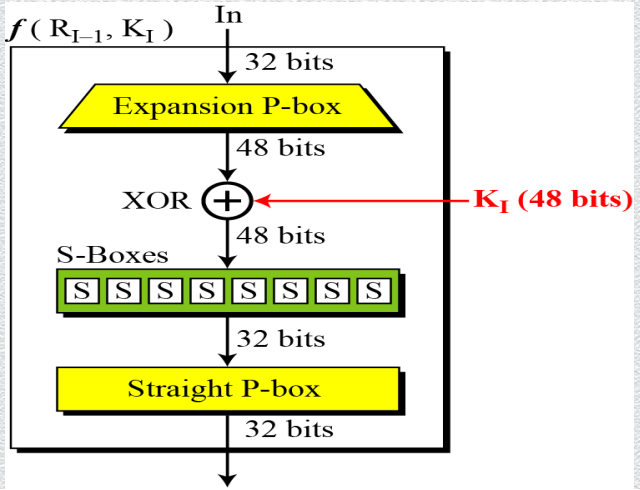


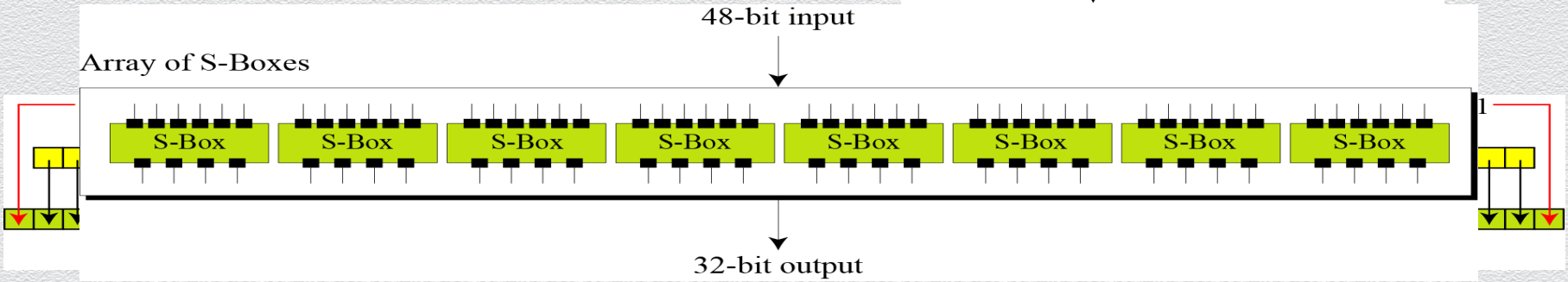


- DES uses 16 rounds, each applying a Feistel cipher
  - L(i) = R(i-1)
  - R(i) = L(i-1) XOR *f* (K(i),R(i-1)),
    where *f* applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output

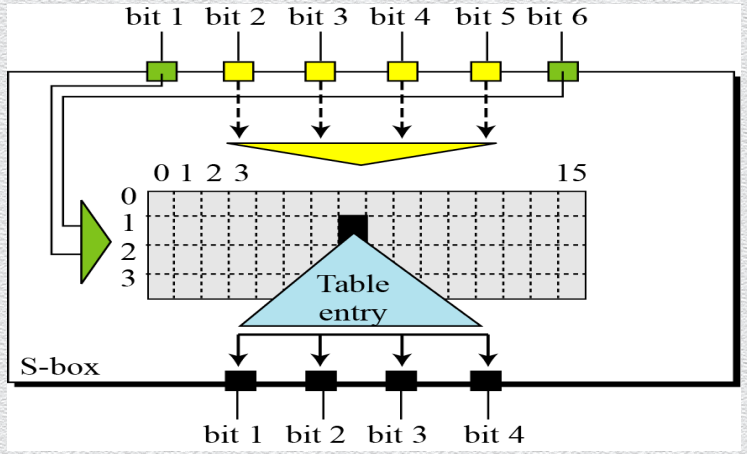
Low-level DES view

- expansion box
  - since R_{I-1} is a 32-bit input & K_I is a 48-bit key, we first need to expand R_{I-1} to 48 bits
- S-box
  - where real mixing (confusion) occurs
  - DES uses 8 6-to-4 bits S-boxes





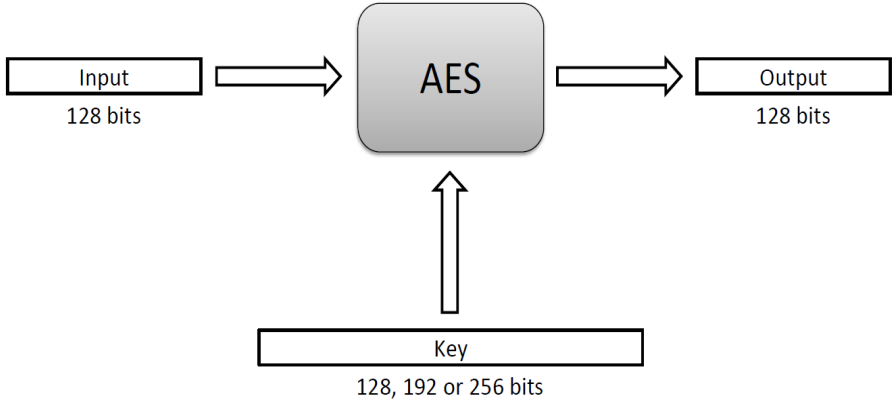
S-box in detail



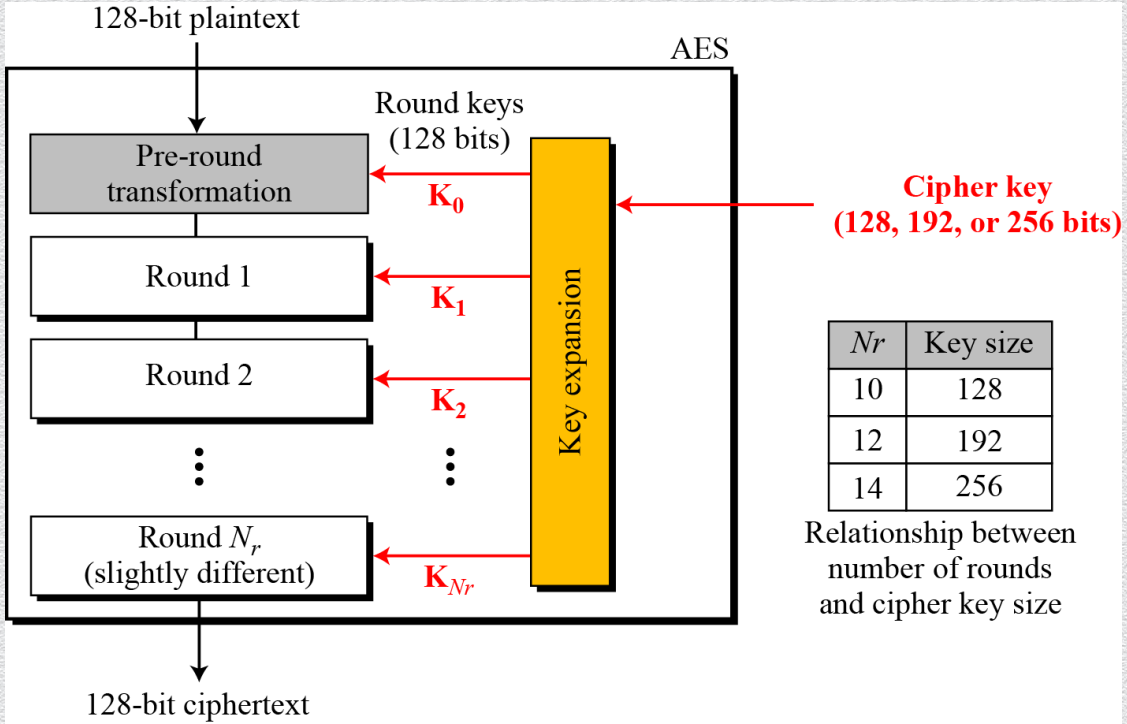
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

AES: Advanced Encryption System

- symmetric block cipher (called Rijndael)
  - developed in 1999 by independent Dutch cryptographers in response to the 1997 NIST's public call for a replacement to DES
- employs substitution, confusion & diffusion
  - on blocks of 128 bits, in 10, 12 or 14 rounds for keys of 128, 192, 256 bits
  - depending on key size, yields ciphers known as AES-128, AES-192, and AES-256
- still in common use
  - on the longevity of AES
    - larger key sizes possible to use
    - not known serious practical attacks



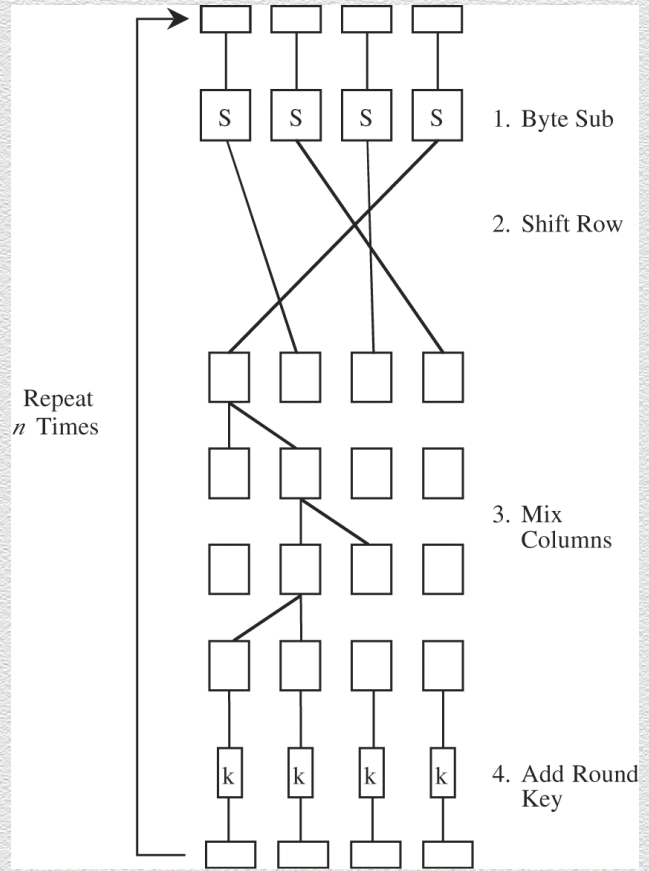
AES structure



Cipher key
(128, 192, or 256 bits)

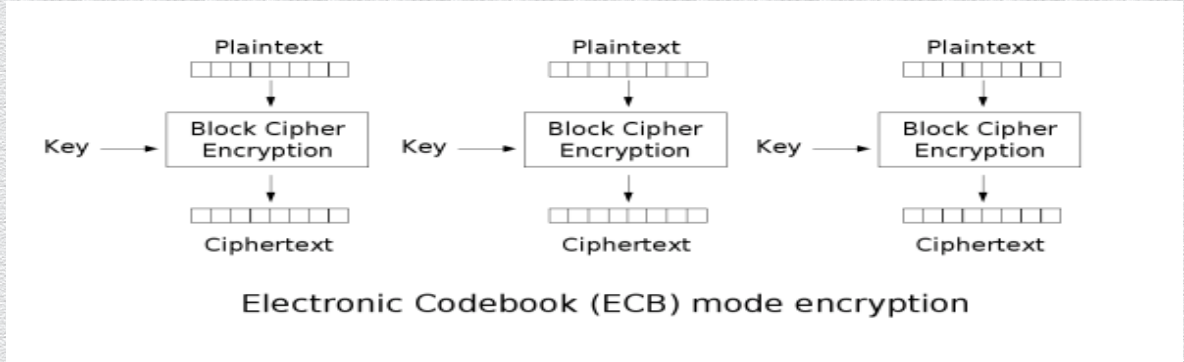
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size



Block cipher modes

- describe the way a block cipher encrypts or decrypts a sequence of message blocks
- electronic Code Book (ECB) mode (is the simplest):
  - block P[i] encrypted into ciphertext block C[i] = $E_K(P[i])$
  - block C[i] decrypted into plaintext block M[i] = $D_K(C[i])$



Electronic Codebook (ECB) mode encryption

Strengths & weaknesses of ECB

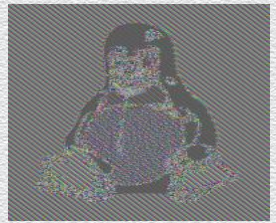
Strengths

- very simple
- allows for parallel encryptions of the blocks of a plaintext
- can tolerate the loss or damage of a block

Weaknesses

- documents and images are not suitable for ECB encryption, since patterns in the plaintext are repeated in the ciphertext
- e.g.,:

ECB

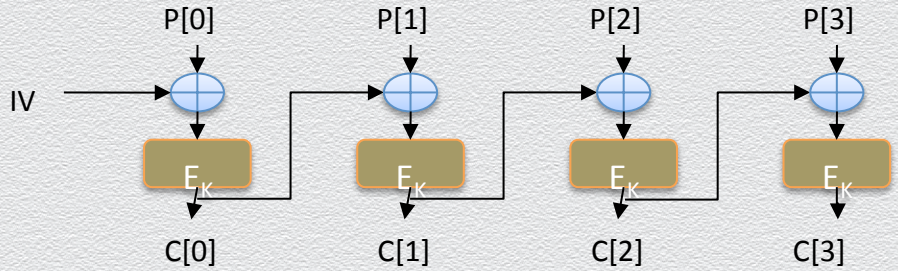


CBC

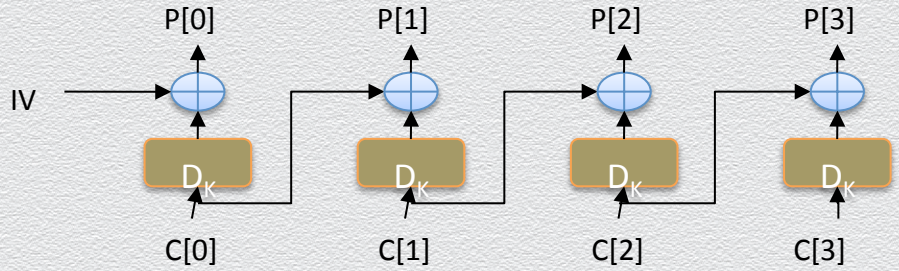
Cipher Block Chaining (CBC) [or chaining]

- ECB produces the same ciphertext on the same ciphertext (under the same key)
- alternatively, the ciphertext of the previous block can be mixed with the plaintext of the current block (e.g., by XORing); an initial vector is used as initial "ciphertext"
- Cipher Block Chaining (CBC) mode:
  - previous ciphertext block is combined with current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - C[-1] = IV, a random block separately transmitted encrypted (a.k.a. the initialization vector)
  - decryption: $P[i] = C[i-1] \oplus D_K(C[i])$

CBC Encryption:



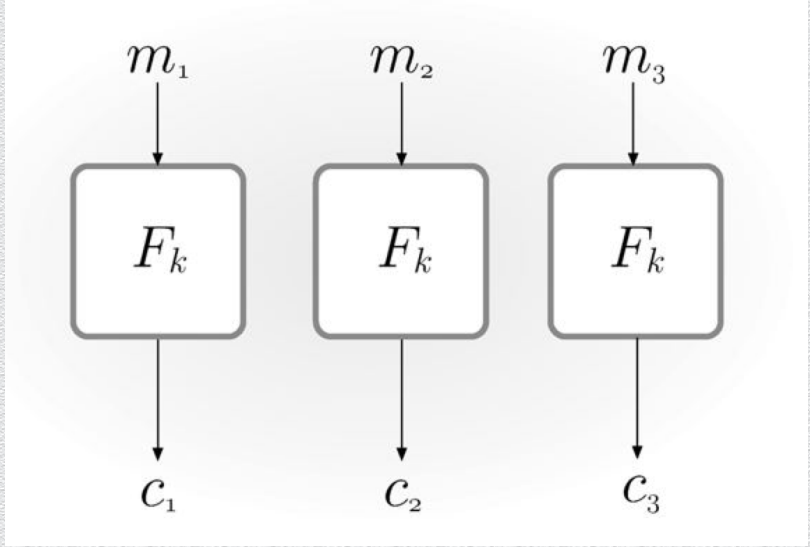
CBC Decryption:



3.4 Modes of operations

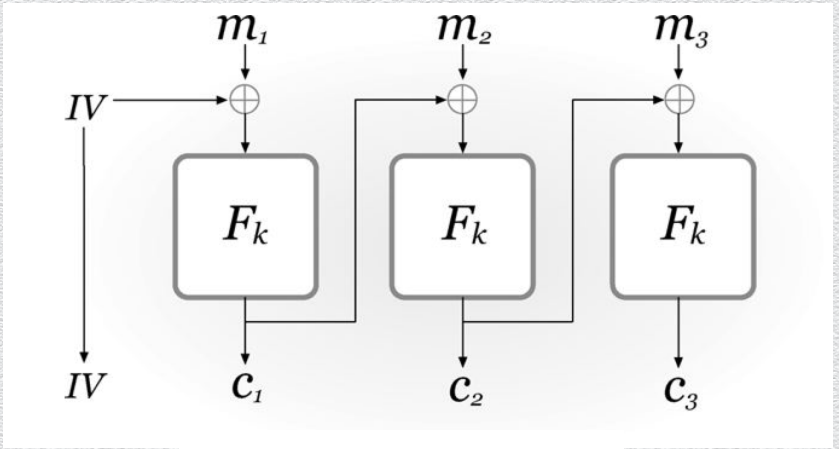
Modes of operation for block ciphers (I)

- ECB - electronic code book
  - insecure, of only historic value
  - deterministic, thus not CPA-secure
  - actually, not even EAV-secure

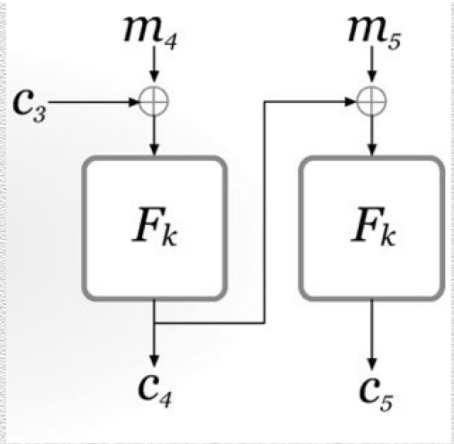


Modes of operation for block ciphers (II)

- CBC – cipher block chaining
  - CPA-secure if F_k a permutation
  - uniform IV
    - otherwise security breaks

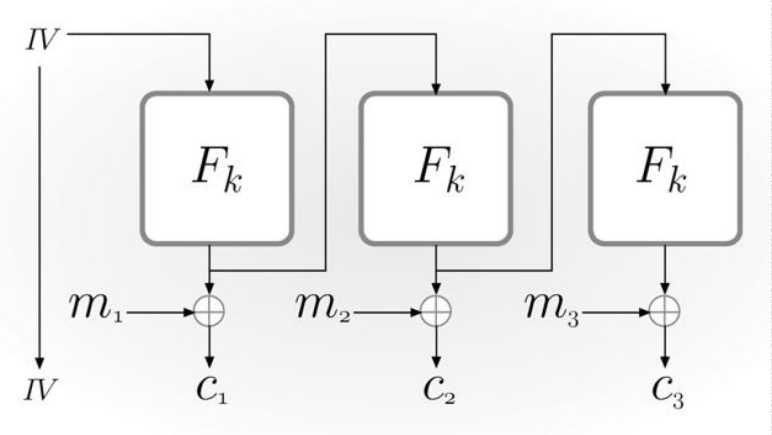


- Chained CBC
  - use last block ciphertext of current message as IV of next message
  - saves bandwidth but not CPA-secure



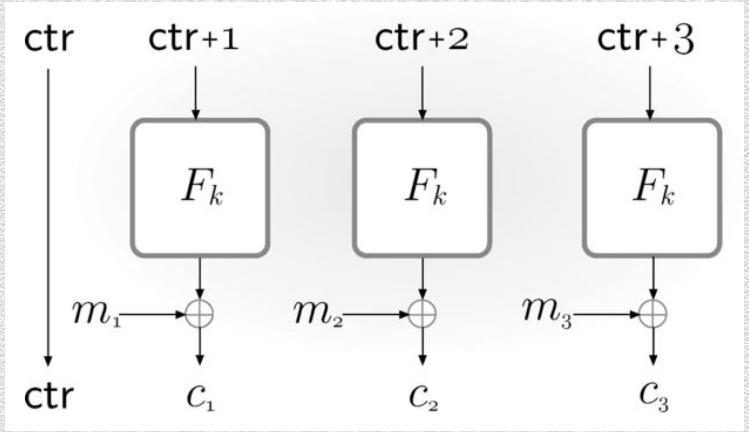
Modes of operation for block ciphers (III)

- OFB – output feedback
  - IV uniform
  - no need message length to be multiple of n
  - resembles synchronized stream-cipher mode
  - stateful variant (chaining) is secure
  - CPA-secure if F_k is PRF



Modes of operation for block ciphers (IV)

- CTR – counter mode
  - ctr uniform
  - no need message length to be multiple of n
  - resembles synchronized stream-cipher mode
  - CPA-secure if F_k is PRF
  - no need for F_k to be invertible
  - parallelizable



Notes on modes of operation

- block length matters
  - if small, IV or ctr can be “recycled”
- IV are often misused
  - e.g., reused or not uniformly random
  - in this case, CBC is a better option than OFB/CTR