PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Private Key System

Lecture 13

Private key cryptography

One key between sender and receiver

Perfect secrecy (formal)

- Encryption scheme (Gen, Enc, Dec) with message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$ is *perfectly secret* if for every distribution over $\mathcal{M}$, every m $\in$ $\mathcal{M}$, and every c $\in$ $\mathcal{C}$ with Pr[C=c] > 0, it holds that

$$\Pr[M = m \mid C = c] = \Pr[M = m].$$

- I.e., the distribution of M does not change conditioned on observing the ciphertext

Perfect secrecy

- Requires that ***absolutely no information*** about the plaintext is leaked, even to eavesdroppers *with* ***unlimited computational power***

  - Has some inherent drawbacks
  - Seems unnecessarily strong

Computational security

- **Small probability of information leakage to eavesdroppers with bounded computational resources**
- Led to two relaxation to notion of security

  to achieve key reusability

  1. Security is only guaranteed against efficient adversaries that run for some feasible amount of time.
  2. Adversaries can break the scheme with small probability as most system does not require ever-lasting security.

Small probability?

- Consider the following four events:

1. Winning a lottery with 1 million contestants
2. Winning a lottery with 1 million contestants 5 times in a row
3. Winning a lottery with 1 million contestants 6 times in a row.
4. Winning a lottery with 1 million contestants 7 times in a row.

- What is the order of these events from most likely to least likely?

Computational security

- Two approaches
  - Concrete security
  - Asymptotic security

Concrete Approach

- A scheme is (t, ε)-secure if any adversary running for time at most t succeeds in breaking the scheme with probability at most ε .
- Disadvantages:
  - Will not consider the progress in the computing speed.

Asymptotic Approach

- Measure algorithms efficiency with respect to basic step depending on input size.
- Various notations are big O, omega and theta

Asymptotic in context of cryptography

- Security parameter: n (Size of secret key)
- All algorithms are expressed as function of security parameter
- Running time of user (encryption/decryption function)
- Running time of adversary
- Success rate of attacker

- AES security parameter n= 128, 192, 256

Polynomial-time algorithm

- Defining efficient algorithm

"Algorithm A has a polynomial running time, if there exists a polynomial p(.), such for every input $x \in \{0,1\}^*$, the computation of A(x) terminates within p(|x|) steps, where |x| denotes the length of the string x"

A polynomial-time algorithm is an algorithm whose execution time is either given by a polynomial on the size of the input or can be bounded by such a polynomial.

Negligible functions

- " A negligible function Is one that is asymptotically smaller than any inverse polynomial function"
- Definition 1:
  - ***A function f from the natural number to a non negative real number is negligible if every positive polynomial p there is an N such that for all integers*** $n > N$ ***it holds that*** $f(n) < \frac{1}{p(n)}$

Security parameter n

Negligible function

- Definition 2:

For every constant c, there exists some N such that $f(n) < n^{-c}$, $for\ all\ n > N$

Therefore

$$f(n) < \frac{1}{p(n)} < \frac{1}{n^c}$$

Example negligible functions

- $2^{-n}, 2^{-\sqrt{n}}, n^{-\log n}$ are all negligible functions
- Consider a function $f(n) = 2^{-n}$

  then $2^{-n} < 1/n^5$ for all n > 23

Carefully select the value of n for meaningful security

Closure properties

- For polynomials

  $if\ p_1(n) and\ p_2(n)\ are\ polynomial\ function\ then$

  $p_1(n) + p_2(n)$ and $p_1(n) X\ p_2(n)$ are polynomial

- for negligible functions

  $negl_3(n) = negl_1(n) + negl_2(n)$ is negligible

  $negl_4(n) = negl_1(n).\mathrm{p}(\mathrm{n})$ is negligible

Private key system

- The key-generation algorithm Gen takes as input 1^n and outputs a key k; we write k ← Gen(1^n)(emphasizing that Gen is a randomized algorithm). We assume without loss of generality that any key k output by Gen(1^n) satisfies |k|>= n.
- 2. The encryption algorithm Enc takes as input a key k and plaintext message m $\in\{0,1\}^*$, and outputs a ciphertext c. Since Enc may be randomized, we write this as c ← Enc_k(m).
- 3. The decryption algorithm Dec takes as input a key k and a ciphertext c, and outputs a message m or an error. We assume that Dec is deterministic, and so write m := Dec_k(c)

The indistinguishability experiment

$$PrivK^{eav}_{A,\Pi}(n)$$

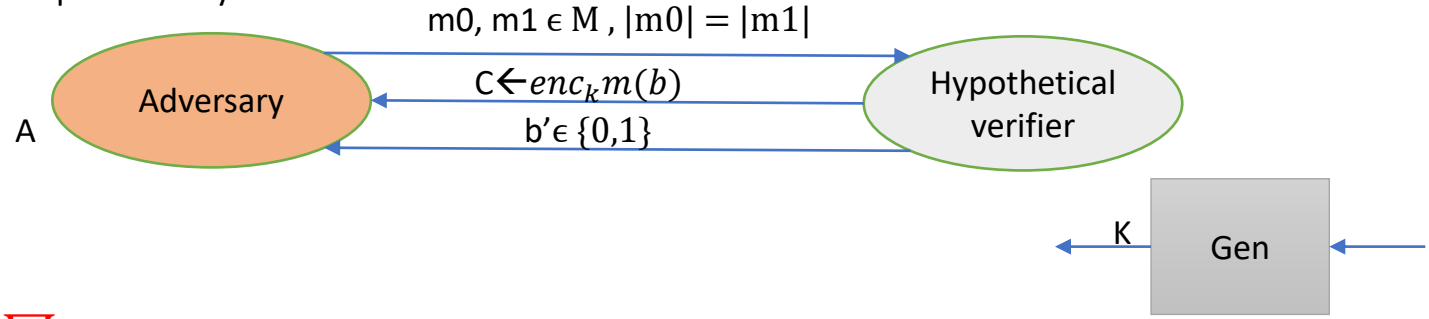
- PrivK denotes an experiment, in context of a private key or symmetric encryption,
- eav means we are considering an adversary who is an eavesdropper,
- A is the name of the adversarial algorithm
- Π is the name of the scheme,

In this experiment, the rules are as follows.

- Adversary can submit any pair of messages (m0, m1) from the plaintext space with the restriction that the size of the two plaintext should be same |m0| = |m1|
- The hypothetical verifier does the following:
  - It randomly generates a key by running a key generation algorithm and it randomly encrypts one of the messages (m0 | m1) using the key,
- The challenge for the adversary is to identify what plaintext has been encrypted in the challenge ciphertext c, whether it is m0 or m1.

The indistinguishability experiment

Computationally bounded



$\prod$ ***is perfectly indistinguishable if for every A***

$$pr(PrivK^{eav}_{A,\prod}(n)=1)=\frac{1}{2}$$

Adversary does

- Outputs a bit, namely its guess about what exactly has been encrypted in the challenge ciphertext.
- The scheme Π is perfectly secure or we say that a scheme is perfectly indistinguishable if the probability with which adversary could successfully Identify what message has been encrypted is upper bounded by half.

Semantic security in COA model

Enc is semantically secure, if the ciphertext does not reveal any additional information about the underlaying plaintext

- Adversary has access to abstract function h(m)
  - h(m): any prior information about plaintext obtained by other means
- Goal of the adversary is to compute f(m) of the underlying plaintext
  - f(m): Addition information that adversary wants to learn about message m

Semantic security in COA model

Semantic security is chances that adversary could compute f(m) using c and h(m) is same as adversary compute f(m) without c.

- $|pr(A(enc_k(m), h(m)) - pr(A'(h(m))| \leq negl(n)$

Let $R := \{0,1\}^4$ and consider the following PRF $F : R^5 \times R \to R$ defined as follows:

$$F(k,x) := \left\{ \begin{array}{l} t = k[0] \\ \text{for i=1 to 4 do} \\ \qquad \text{if } (x[i-1] == 1) \quad t = t \oplus k[i] \\ \text{output } t \end{array} \right.$$

That is, the key is $k = (k[0], k[1], k[2], k[3], k[4])$ in R^5 and the function at, for example, 0101 is defined as $F(k, 0101) = k[0] \oplus k[2] \oplus k[4]$.

For a random key k unknown to you, you learn that

$$F(k, 0110) = 0011 \text{ and } F(k, 0101) = 1010 \text{ and } F(k, 1110) = 0110\,.$$

What is the value of $F(k, 1101)$? Note that since you are able to predict the function at a new point, this PRF is insecure.

Thank You!

Next Class

- ☞ Mandatory reading for the next class
- ☞ https://en.wikipedia.org/wiki/Pseudorandom_number_generator

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Private key systems

Lecture 2

Pseudo Random Numbers

generating a sequence of numbers

Pseudorandom generators

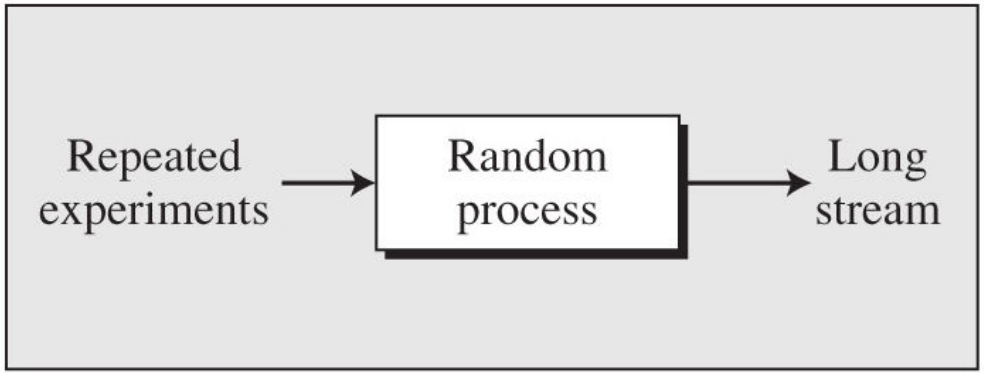
- A pseudorandom generator G is an efficient deterministic algorithm for transforming a short, uniform string called seed into a longer "uniform-looking" output string
- Let G: $\{0,1\}^n \rightarrow \{0,1\}^l$ be a function and define Dist to be the distribution on l-bit string obtained by choosing a uniform s$\in \{0,1\}^n$ and outputting G(s)

Randomness

- Truly random
- Pseudo random

Truly random

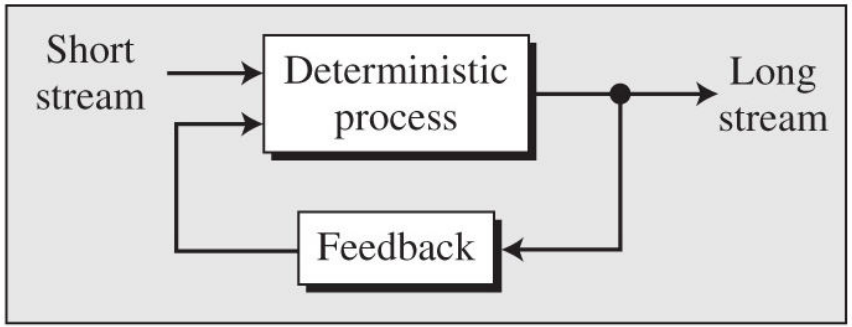
- TRG G' is a randomised algorithm
- Output $R \in \{0,1\}^L$
- Uniformly randomly output a bit string of length L bits



a. TRGN

Pseudorandom generators

- G should be an ***efficient algorithm***
- ***Expansion:*** $L > l$
- ***Pseudo randomness:*** No efficient statistical test should significantly separate an output of G from L bit truly random generator



b. PRNG

Generator Algorithm

- Function G is called a **pseudorandom generator**.
- G is a deterministic algorithm

Input: s$\in \{0,1\}^l$

Output: G(s) $\in \{0,1\}^L$

Note: G should be a polynomial function of a security parameter (efficient).

Seed

- Kept secret
- Chosen uniformly

ALGORITHM 3.16
Constructing G_ℓ from (Init, GetBits)

Input: Seed s and optional initialization vector IV
Output: $y_1, \ldots, y_\ell$

$\mathsf{st}_0 := \mathsf{Init}(s, IV)$
for $i = 1$ **to** ℓ:
  $(y_i, \mathsf{st}_i) := \mathsf{GetBits}(\mathsf{st}_{i-1})$
return $y_1, \ldots, y_\ell$

Encrypting long messages using short keys

$$M = K = C = \{0,1\}^L \quad G\{0,1\}^l \rightarrow \{0,1\}^L \qquad l < L$$

Consider mϵM= $\{0,1\}^L$

Encryption done by computing m$\oplus$G(s).

s is the seed = $\{0,1\}^l$

- A computationally bounded adversary will not be able to distinguish between G(s) and uniformly random string from $\{0,1\}^L$
- Both l and L are polynomial functions of security parameter n

PRG indistinguishability game

1. Hypothetical verifier:

   Challenges the distinguisher by a string or a sample of length L bits.

2. Distinguisher D

   Distinguish apart a sample generated by the pseudorandom generator from a sample generated by a truly random generator

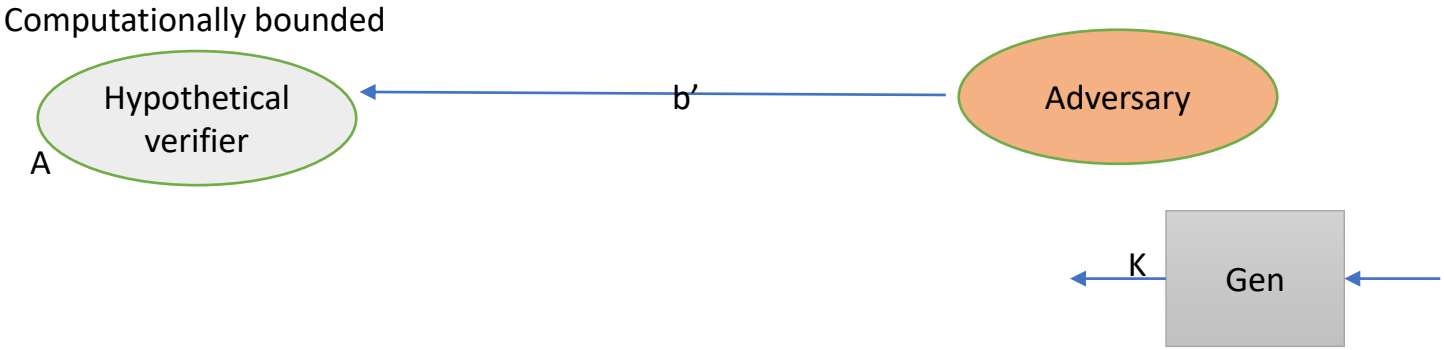
Hypothetical verifier

Uses two method to generate string of L bits

1. Truly random number generator (b=0) $y_R \in \{0,1\}^L$
2. Pseudorandom number generator (b=1) $s\{0,1\}^l_{\rightarrow} y_p \in \{0,1\}^L$

And sends the y bits to the distinguisher

The indistinguishability experiment



Distinguisher D

Should find how y bits are generated

If for every distinguisher D participating in this experiment

$$pr(D\ outputs\ b' = b) \leq \frac{1}{2} + negl(n)$$

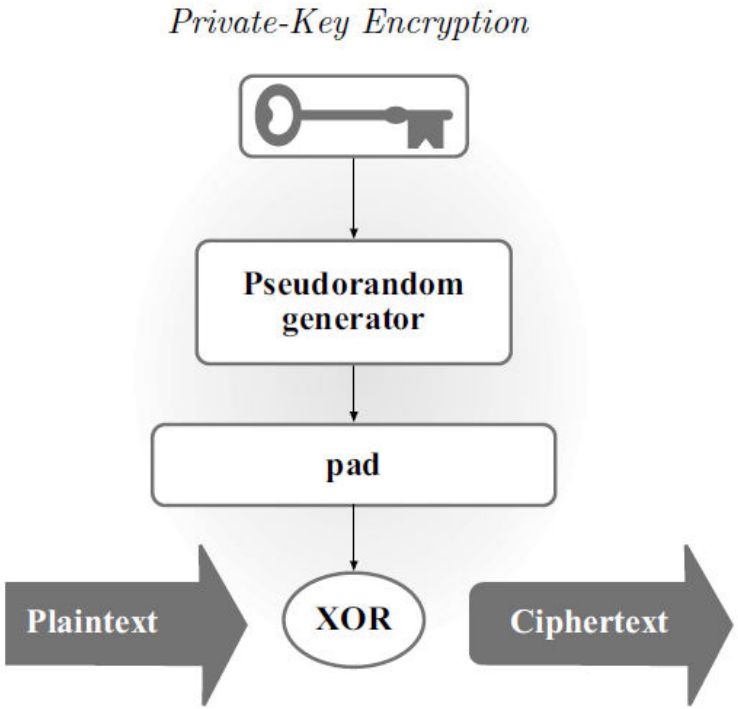
$$pr(D\ outputs\ b' = 1 | b = 0)$$

-

$$pr(D\ outputs\ b' = 1 | b = 1) \leq negl(n)$$

Probability of D labelling y as outcome of PRG given that y is generated by TRG - Probability of D labelling y as outcome of PRG given that y is generated by PRG $\leq negl(n)$

Encryption with a Pseudorandom generator



Pseudorandom Functions

- PRF does not require a one-to-one mapping between the input space and output space.
- A Pseudo Random Permutation is a PRF that happens to have the property that every element in the input domain has a single associated member in the output co-domain
- PRP is a bijection function (one-to-one mapping)

PRP (pseudorandom permutation)

- PRP is invertible.

*Let $F, F^{-1} : \{0,1\}^{\lambda} \times \{0,1\}^{blen} \rightarrow \{0,1\}^{blen}$ be deterministic and efficiently computable functions. Then F is a **pseudorandom permutation (PRP)** if for all keys $k \in \{0,1\}^{\lambda}$, and for all $x \in \{0,1\}^{blen}$, we have:*

- $F^{-1}(k,F(k,x)) = x$.

For example initial permutation and final permutation in DES

Thank you

Next Class

☞ Mandatory reading for the next class

- ☞ https://www.coursera.org/lecture/symmetric-crypto/feistel-cipher-YgMcO

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Private key Systems

Lecture 3

Feistel Cipher

Used by DES

Types of Symmetric key cipher

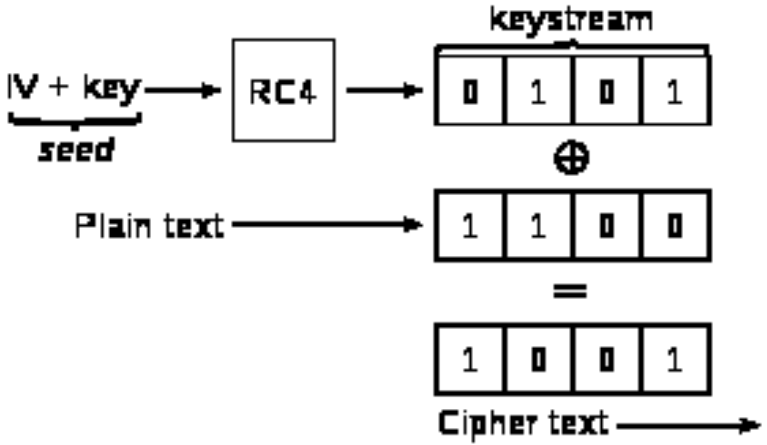
- ***Stream cipher:***
  - algorithm operates on individual bits (or bytes) one at a time
  - Example RC4 cipher system
- ***Block cipher:***
  - operates on fixed-length groups of bits called blocks
  - Example DES, Triple DES and AES

Stream Cipher (Rivest Cipher 4)

- Key stream
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
  - Can be generated on-line one bit (or byte) at the time
- Stream cipher
  - XOR the plaintext with the key stream C[i] = S[i] ⊕ P[i]
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream

RC4

- **Wired Equivalent Privacy** (WEP deprecated in 2004) used the stream cipher RC4 for confidentiality.



Limitations of stream cipher

- Keystream must have a large ***period*** and it must be impossible to ***recover the cipher's key*** or internal state from the keystream.
- One never reuse the same keystream twice
    - different ***nonce or key*** must be supplied

Block cipher

- Partition the text into relatively large (e.g. 128 bits) blocks and encode each block separately.
- The encoding of each block generally depends on at most one of the previous blocks.
- The same “key” is used at each block.

Difference between block and stream ciphers

- Block ciphers work a on block / word at a time, which is some number of bits. All of these bits have to be available before the block can be processed.
- Block cipher uses either 64 bits or more than 64 bits.
- The complexity of block cipher is simple.

- Stream ciphers work on a bit or byte of the message at a time, hence process it as a “stream”.
- While stream cipher uses 8 bits.
- While stream cipher is more complex.

Difference between block and stream ciphers

- Block cipher Uses confusion as well as diffusion.
- In block cipher, reverse encrypted text is hard.
- The algorithm modes which are used in block cipher are: ECB (Electronic Code Book) and CBC (Cipher Block Chaining).

- While stream cipher uses only confusion.
- While in stream cipher, reverse encrypted text is easy.
- The algorithm modes which are used in stream cipher are: CFB (Cipher Feedback) and OFB (Output Feedback).

Confusion and diffusion

- Diffusion:
  - Refers to dissipating the statistical structure of plaintext over the bulk of ciphertext.
  - Makes statistical relationship between the plaintext and ciphertext as complex as possible

- Confusion:
  - Refers to making the relationship between the ciphertext and the symmetric key as complex and involved as possible;
  - Makes relationship between ciphertext and key as complex as possible

Block cipher design principle

- ***Block size***
  - increasing size improves security, but slows cipher
- ***Key size***
  - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- ***Number of rounds***
  - increasing number improves security, but slows cipher
- ***Subkey generation***
  - greater complexity can make analysis harder, but slows cipher
- ***Round function***
  - greater complexity can make analysis harder, but slows cipher

Feistel cipher

- Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived.
- DES is just one example of a Feistel Cipher.
- DES is a cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption

History

- Feistel Cipher: the fundamental building block of DES designed by IBM.
- DES was adopted as a US federal standard for commercial encryption in 1975.
- Design requirements:
  - must provide high level of security (commercial standard)
  - Security must not depend on secrecy of algorithm (Kerckhoff's principle)
  - Must be easily and economically implemented

Feistel cipher structure

- Horst Feistel derived the Feistel cipher based on invertible product cipher
- process
  - Partitions input block into two halves
  - process through multiple rounds which perform a substitution on left data half based on round function of right half and subkey permutation
  - swapping both left and right partition
- Implements Shannon's SP net concept

Feistel Cipher: a cipher design pattern

- Encryption :N rounds
- Plaintext = (L0, R0)

  1 <= i <= n

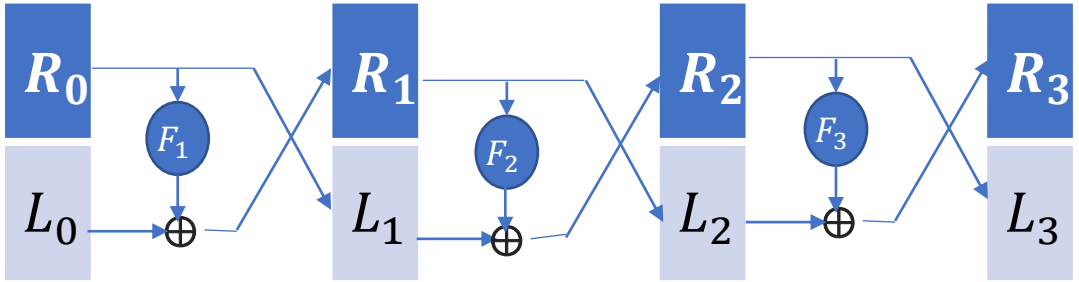
  Li = Ri-1

  Ri = Li-1 xor f(Ri-1 , Ki)

  Subkeys Ki derived from key K

  Ciphertext = (Rn, Ln) Note: swapped halves
- Decryption: As Encryption above, but subkeys applied in reverse order: N, N-1, N-2, …

Feistel Cipher for 3 rounds



F: K^3 X $\{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ *is a secure PRP*

Page 212 FIGURE 6.5: A three-round Feistel network.

Thank you

Next Class

☞ Mandatory reading for the next class

- ☞ https://www.oreilly.com/library/view/computer-security-and/9780471947837/sec9.3.html

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

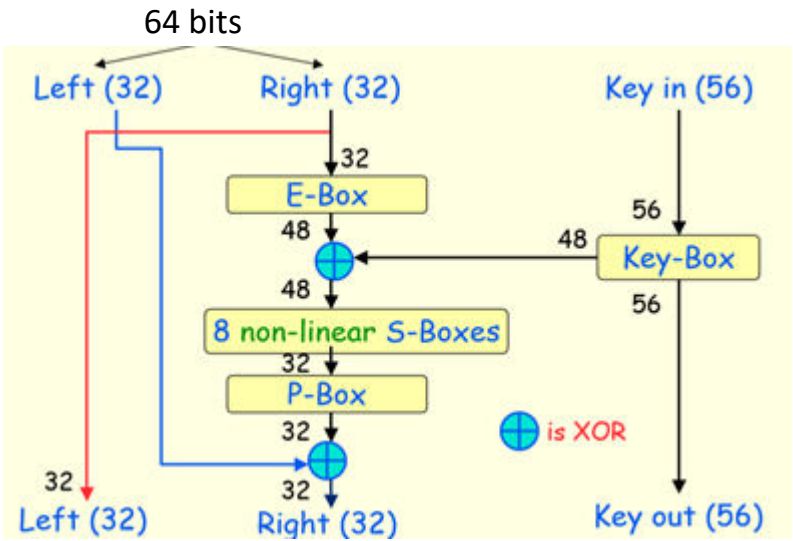
Private key Systems

Lecture 4

S-box and E-box

Substitution and extension box

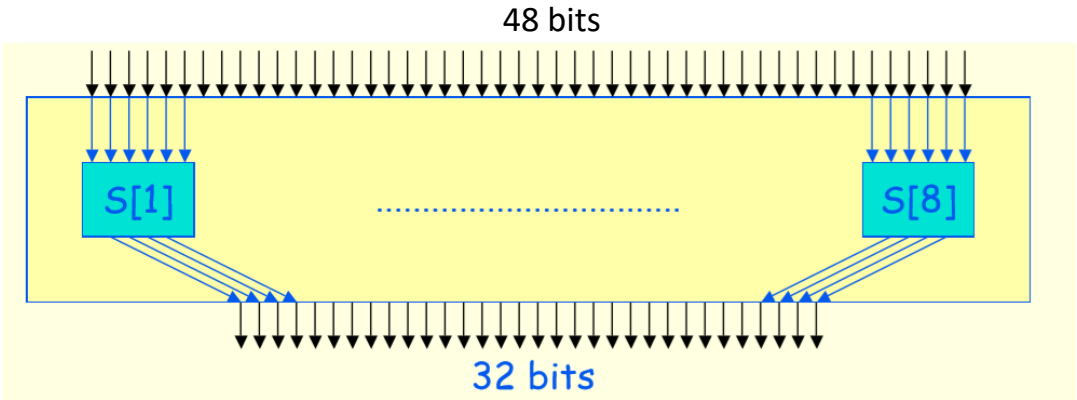
One round of Fiestel cipher



- 8 s-boxes
- 1 e-box
- *The round function in DES is a substitution-permutation network*
- *Block length of 64 bits and a key length of 56 bits are the shortcomings of DES*
- *Each S-box defines a 4 to 1 function*
- *Even though the best-known attack on DES is an exhaustive search, DES is insecure*

S-box

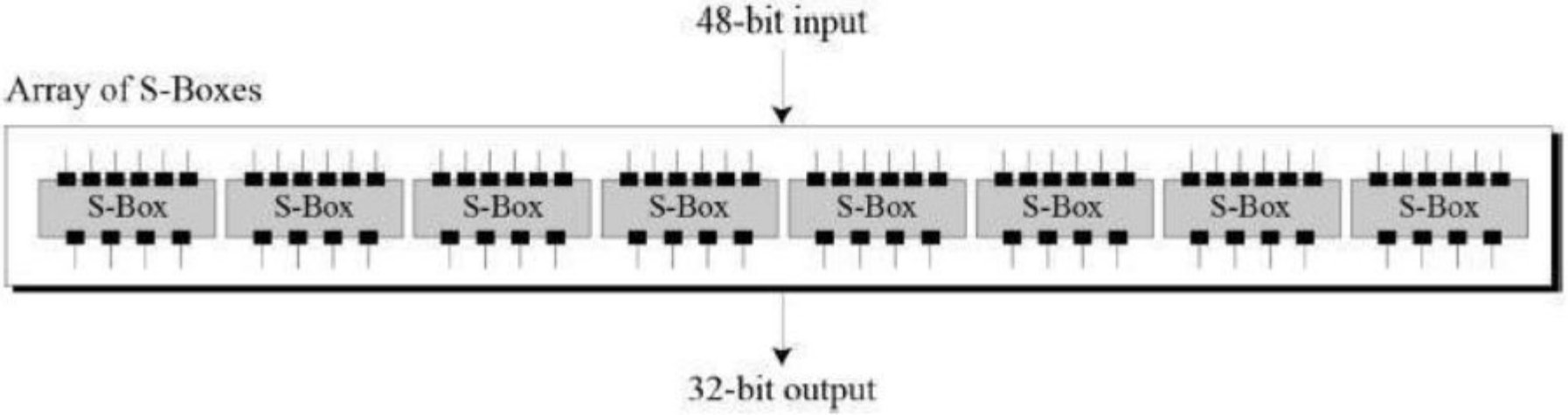
- ***The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.***



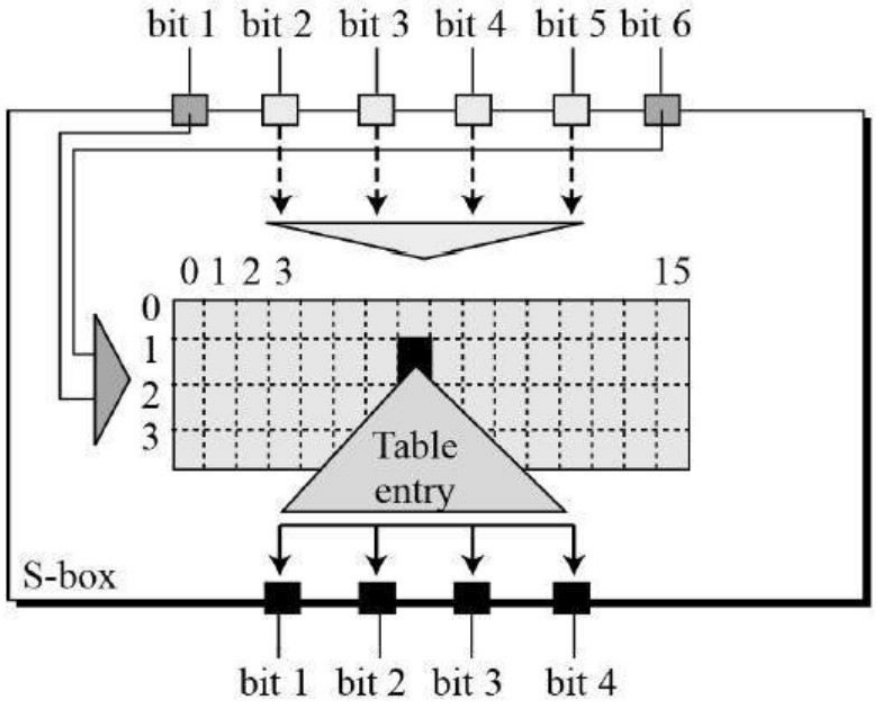
S box

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

8 S-Boxes in DES



Working of s-box



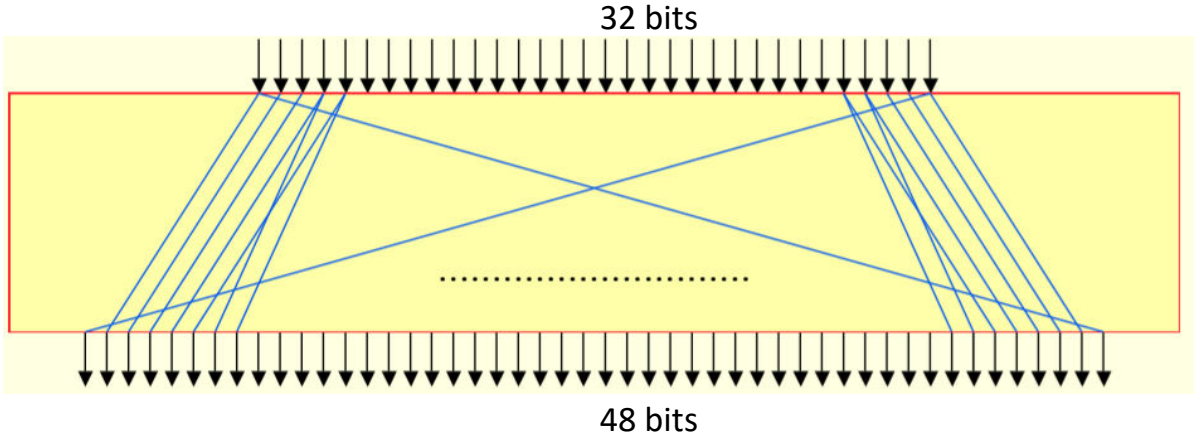
- Each s-box takes 6 bit input
- First(bit1) and last(bit6) bit forms row value
- Remaining 4 bits(bit2-bit5) forms column value
- Output is 4 bit

The input to S-box 1 is 100011. What is the output?

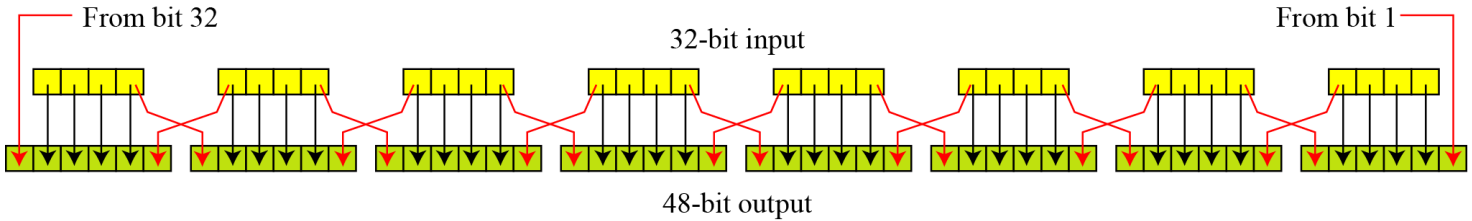
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

E-box

- E-box expands and permutates 32 bits to 48 bits



Expansion box (32 bit input 48 bit output)



1. E box take the left 32(left half) bits and expands it to 48 bits so that 48 bit key can be xor to it.

2. The o/p of step1(48 bits) is fed as i/p to 8 s-boxes (6 bits *8 = 48)

3. Each s-box produce 4 bits (4*8 = 32bits)

4. 32 bits will be concatenated to output

Avalanche effect

- Design S-boxes and mixing permutation (E-box) to ensure avalanche effect
  - Small differences should eventually propagate to entire output
- S-boxes: 1-bit change in input causes ≥2-bit change in output
  - Not so easy to ensure!
- Mixing permutation
  - Each bit output from a given S-box should feed into a *different* S-box in the next round

Thank you

Next Class

- ☞ Mandatory reading for the next class
  - ☞ https://link.springer.com/content/pdf/10.1007%2F3-540-46885-4_71.pdf

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

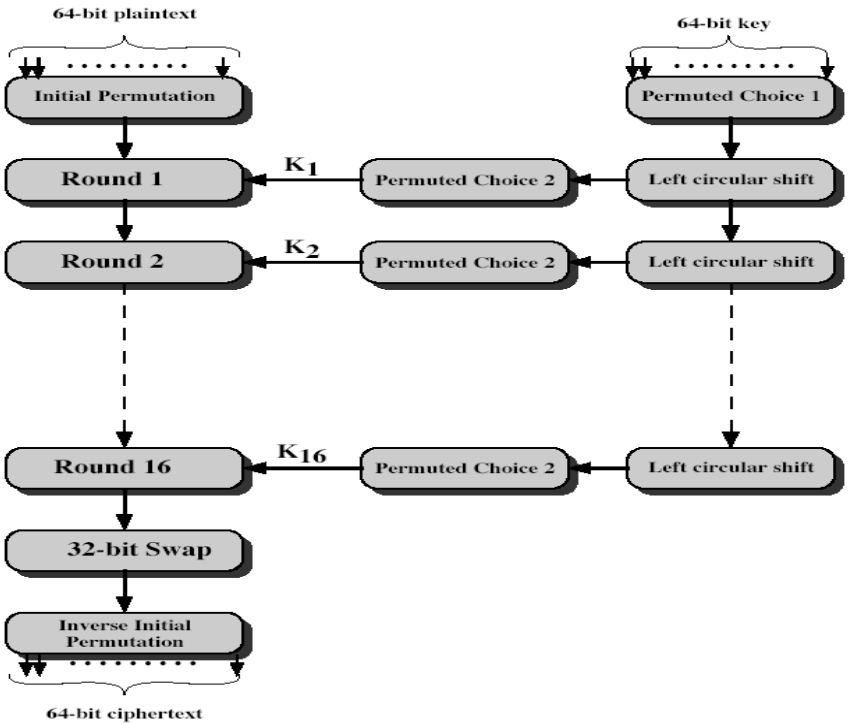
Private key Systems

Lecture 5

Initial and Final permutations

Change in bit position changes a lot in output

Initial permutation



- Initial permutation IP which shuffles the 64 bit input block
- A final permutation being inverse of Initial permutation

Initial permutation tables

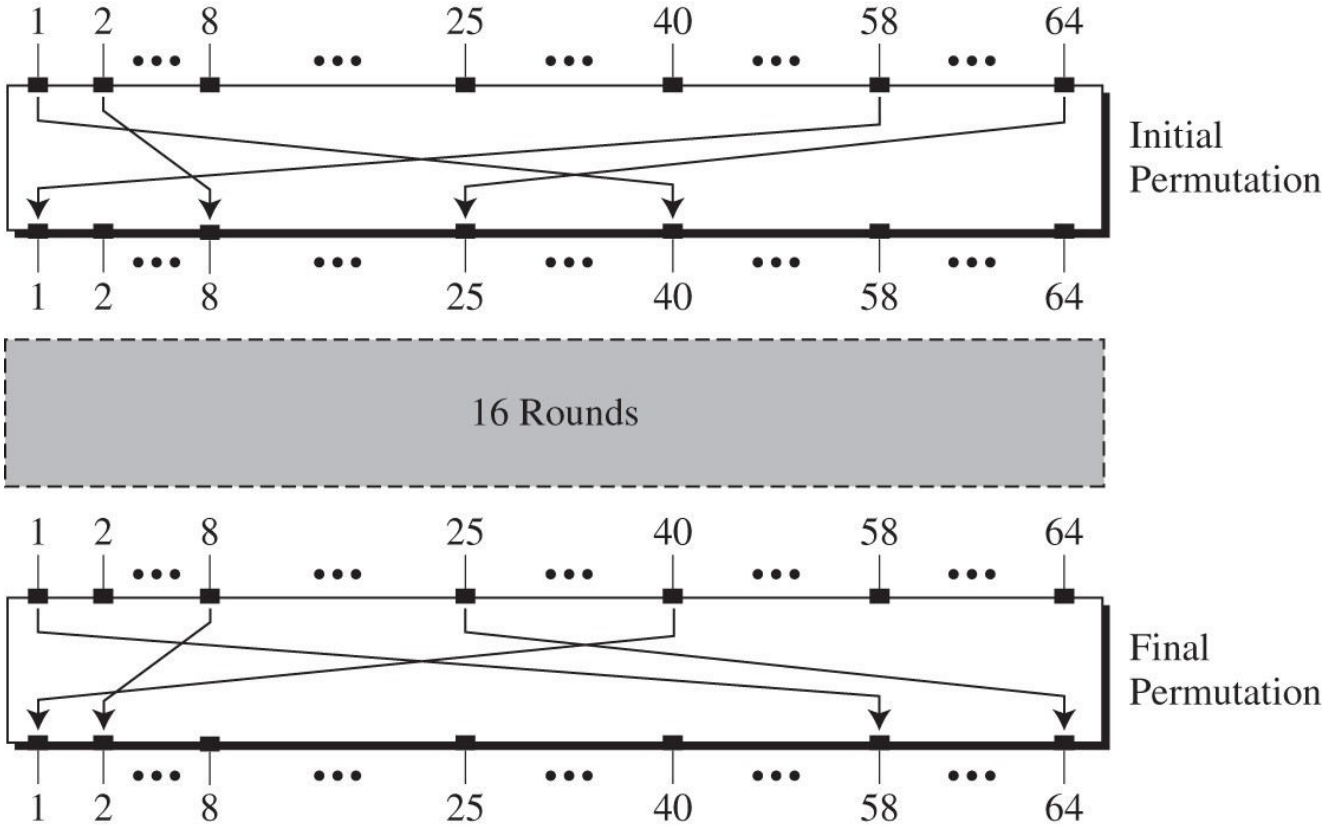
Initial Permutation							
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

- Means a value which is present a 1 bit at input of p-box (0 or 1) should be moved to 58 bit in output of p-box.

Final *permutation tables*

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

- Inverses the initial permutation values
- Observe that a value at 58 position will be moved to 1 position which is nothing but reverse of initial permutation.


1 2 8 25 40 58 64
Initial Permutation
1 2 8 25 40 58 64
16 Rounds
1 2 8 25 40 58 64
Final Permutation
1 2 8 25 40 58 64

Find the output of the initial permutation box when the input is 0x0000 0080 0000 0002

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

- Input is in hexadecimal number
- Convert it into binary format
- Place the bit values according to initial permutation table
- Convert back the result to hexadecimal number

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0				0				0				0				Data
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Binary
58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04	IP
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	IP o/p

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0				0				8				0			
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
0	0	0	0	0	00	0	0	0	0	0	0	0	0	0	0

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
0				0				0				0			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
0				0				0				2			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

when the input is 0x0000 0080 0000 0002
Output is 0x0002 0000 0000 0001

Prove that the initial and final permutations are the inverse of each other (Assignment for Learners)

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

- From the previous slide input for final permutation is

  0x0002 0000 0000 0001

When bit value changes according to final permutation table the result is

0x0000 0080 0000 0002

Which is same as input to initial permutation

- Therefore initial and final permutation are inverse of each other

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Private key systems

Lecture 18

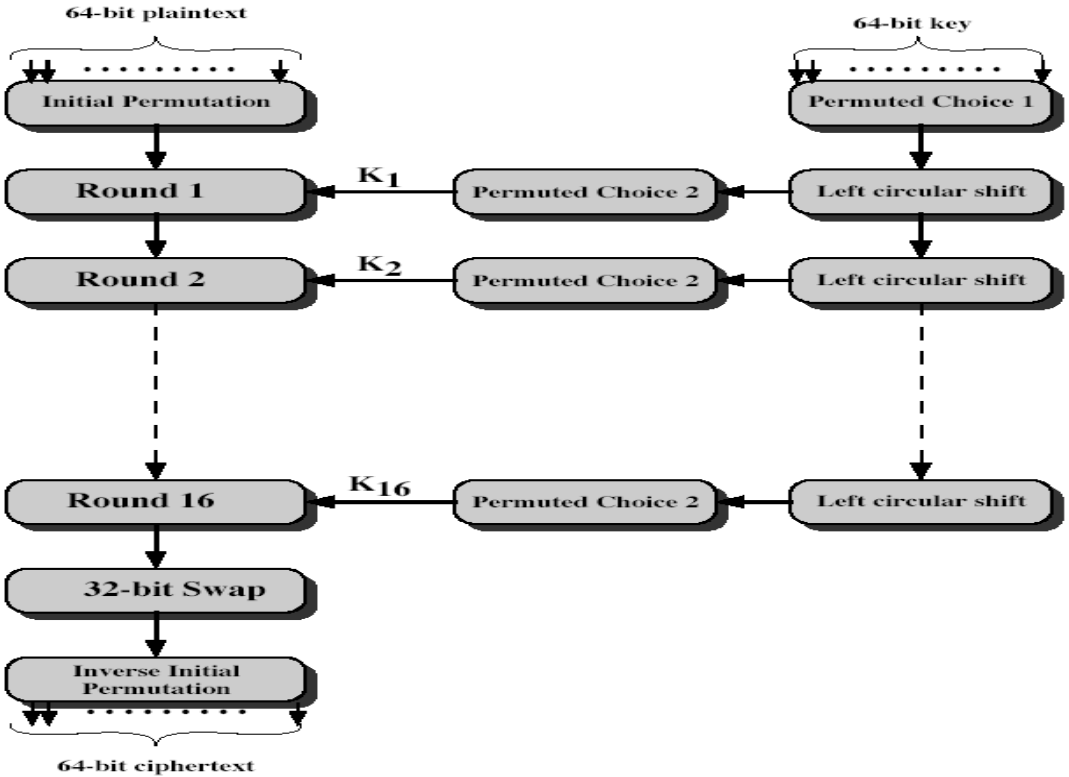
DES

Data Encryption Standard

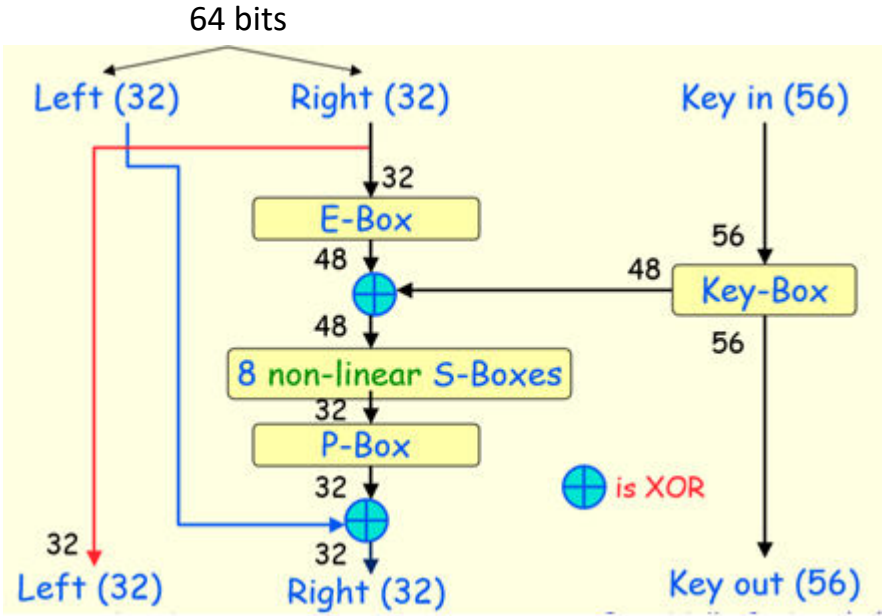
Process

- The process of enciphering a 64 bit input block consists of
  - Initial permutation IP which shuffles the 64 bit input block
  - 16 rounds of complex key dependent round function involving substitution and permutation.
  - A final permutation being inverse of Initial permutation

DES

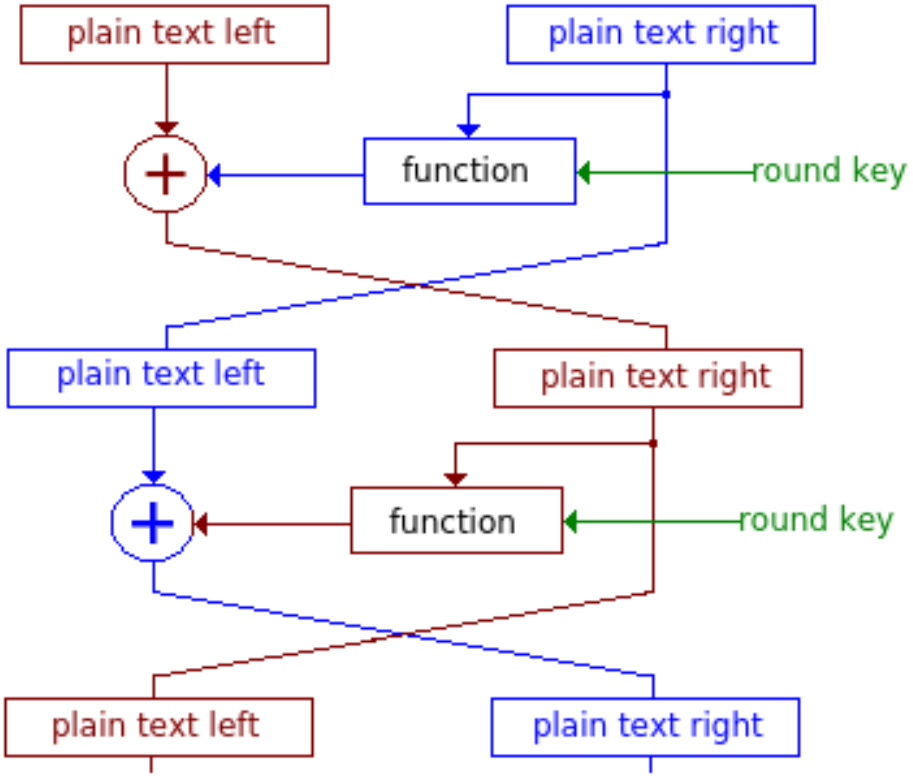

64-bit plaintext
64-bit key
Initial Permutation
Permuted Choice 1
Round 1
K1
Permuted Choice 2
Left circular shift
Round 2
K2
Permuted Choice 2
Left circular shift
Round 16
K16
Permuted Choice 2
Left circular shift
32-bit Swap
Inverse Initial Permutation
64-bit ciphertext

The DES mangler function (one round)



- Expands right 32 bits to 48 bits
- This 48 bits xor with 48 bits of key given to 8 s boxes
- Each s – box produces 4 bit output

Output of first round is input for second round



- Separate message block into two 32-bit halves, L_i and R_i
  - Introduce **confusion** by using a "complex" nonlinear function f
  - f has two inputs: R_i and a 48-bit round key, K_i
  - Introduce **diffusion** by "adding" L_i and the output of f

$L_{i+1} = R_i$

$R_{i+1} = L_i \oplus f(R_i, K_{i+1})$

Bit change in input

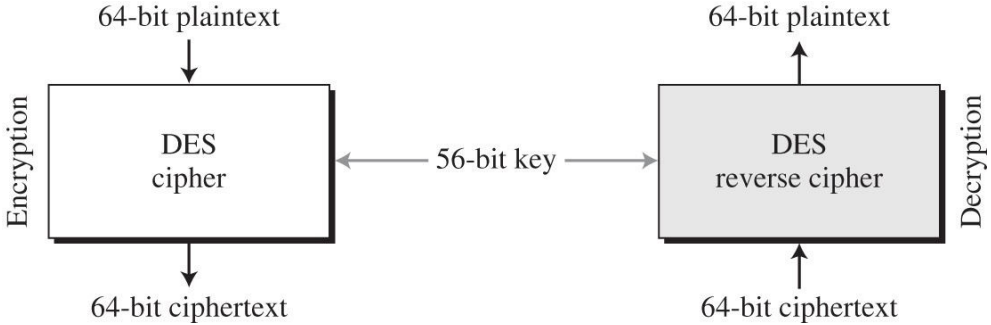
- To check the avalanche effect in DES, encrypt two plaintext blocks (with the same key) that differ only in one bit and observe the differences in the number of bits in each round.

Plaintext: 0000000000000000 Key: 22234512987ABB23
Ciphertext: 4789FD476E82A5F1

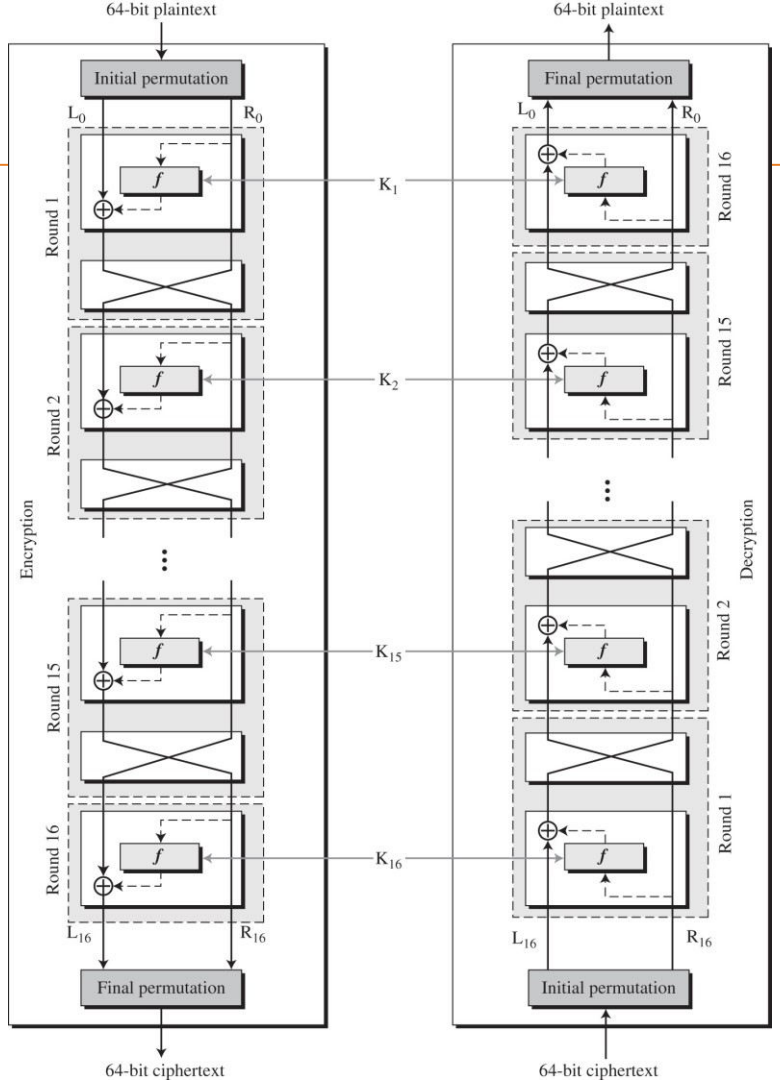
Plaintext: 0000000000000001 Key: 22234512987ABB23
Ciphertext: 0A4ED5C15A63FEA3

Reference 1: page 215

DES


64-bit plaintext
Encryption
DES
cipher
56-bit key
DES
reverse cipher
Decryption
64-bit plaintext
64-bit ciphertext
64-bit ciphertext

16 rounds


64-bit plaintext
Initial permutation
L_0
R_0
Round 1
Round 2
Round 15
Round 16
K_1
K_2
K_{15}
K_{16}
Encryption
L_{16}
R_{16}
Final permutation
64-bit ciphertext
64-bit plaintext
Final permutation
Round 16
Round 15
Round 2
Round 1
Decryption
Initial permutation
64-bit ciphertext

Multiple DES

- Major limitation of DES
- The major criticism of DES regards its key length.
- Key length is too short (56 bits).

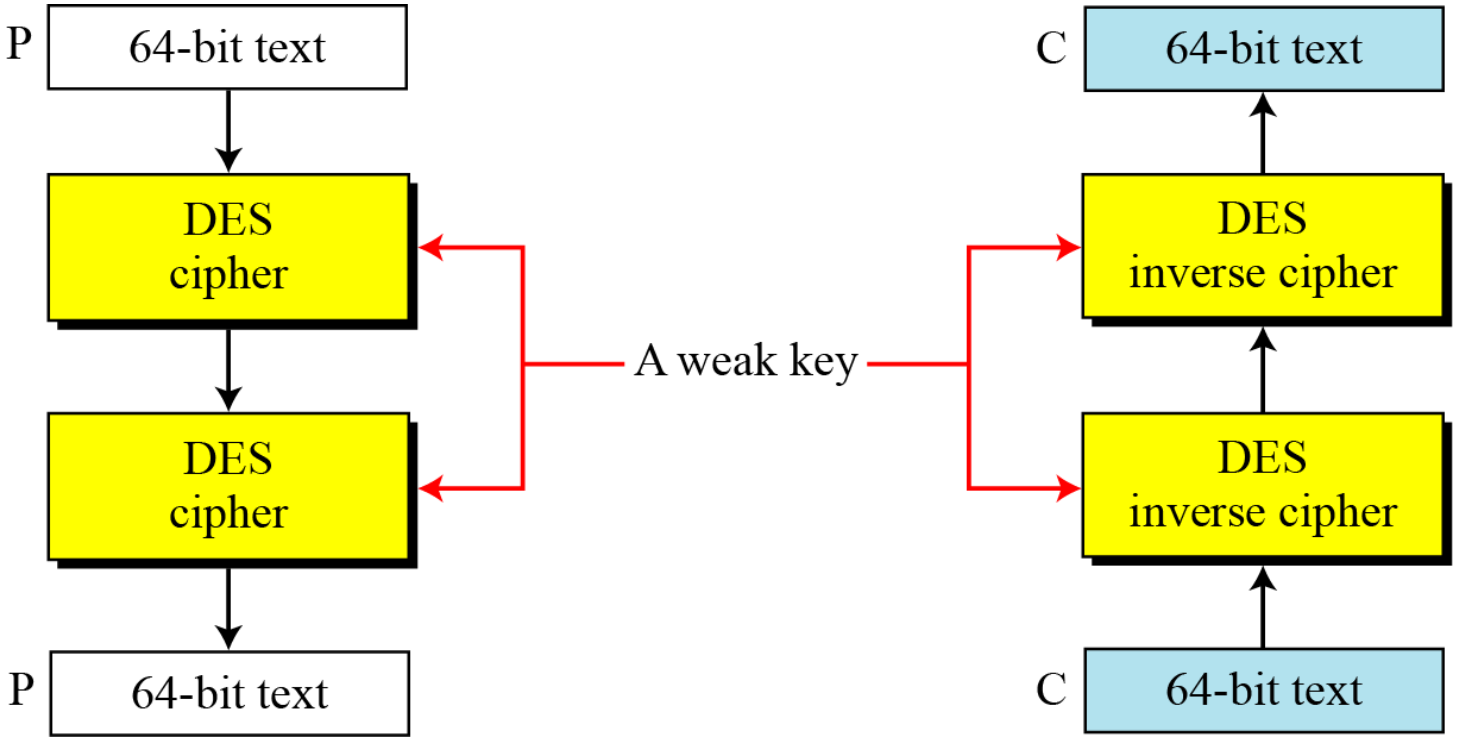
  Question:

  So can we apply DES multiple times to increase the strength of encryption?

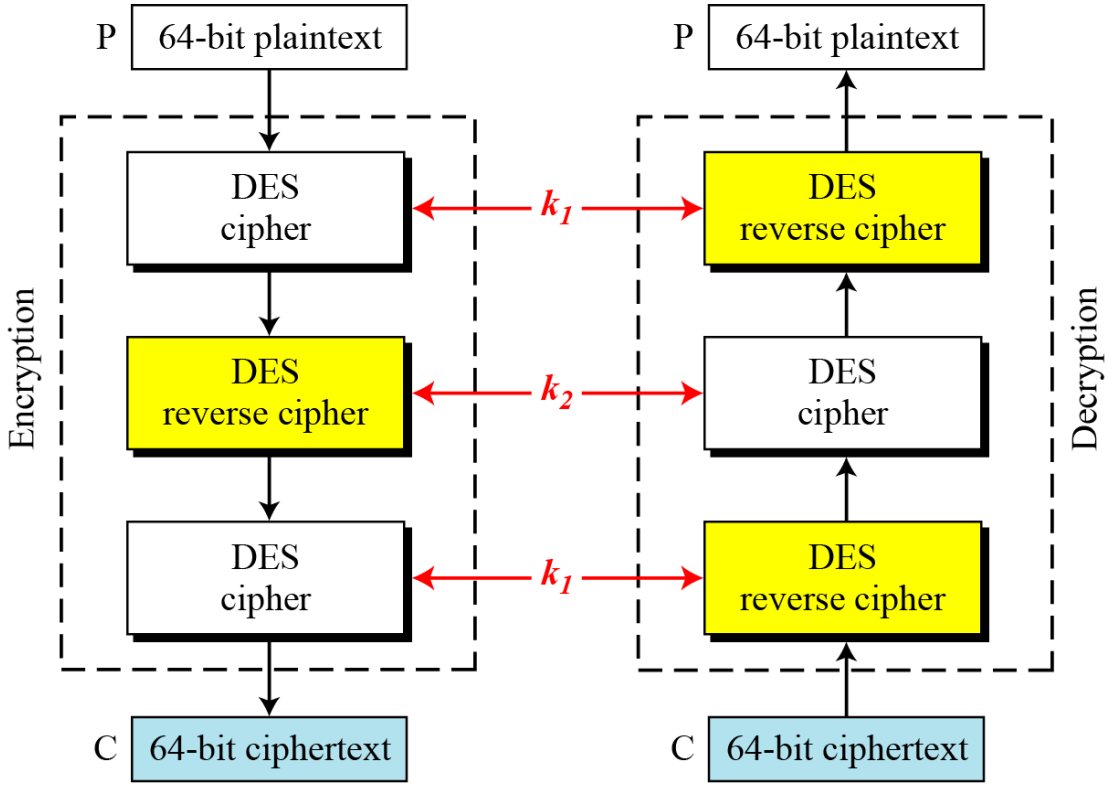
- Advantage:
  - We could then preserve the existing investment in software and equipment.
- Double DES
- Triple DES

Reference 1 page 221 to 223

Double DES


P
64-bit text
DES cipher
DES cipher
P
64-bit text
A weak key
C
64-bit text
DES inverse cipher
DES inverse cipher
C
64-bit text

Triple DES (with 2 key)



Triple DES

Triple DES with Three Keys

The possibility of known-plaintext attacks on triple DES with two keys has enticed some applications to use triple DES with three keys. Triple DES with three keys is used by many applications such as PGP

Thank you

Next Class

- ☞ Mandatory reading for the next class
  - ☞ https://en.wikipedia.org/wiki/Avalanche_effect#:~:text=In%20cryptography%2C%20the%20avalanche%20effect,half%20the%20output%20bits%20flip).

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Private key systems

Lecture 7

Cryptanalysis

Linear and differential

DES Weaknesses

- ***Weaknesses in Cipher Design***
    1. ***Weaknesses in S-boxes***
    2. ***Weaknesses in P-boxes***
    3. ***Weaknesses in Key***

Security of DES

- DES, as the first important block cipher, has gone through much scrutiny.
- The size of the key space, 64, is “too small” to be secure. Brute-Force Attack: Combining short cipher key in DES with the key complement weakness.
- Security of DES mainly relies on the nonlinearity of the function f

Security of DES

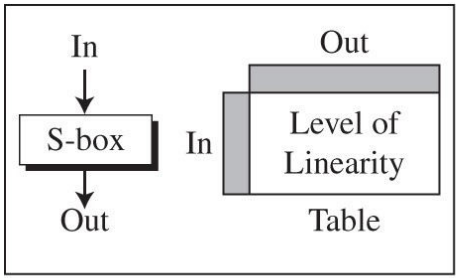
- Differential cryptanalysis: Designed S-boxes and 16 rounds aim to make DES specifically resistant to this type of attack.
- Linear cryptanalysis: DES is more vulnerable to linear cryptanalysis than to differential cryptanalysis. S-boxes are not very resistant to linear cryptanalysis. It has been shown that DES can be broken using ***243*** pairs of known plaintexts.

Linear Cryptanalysis

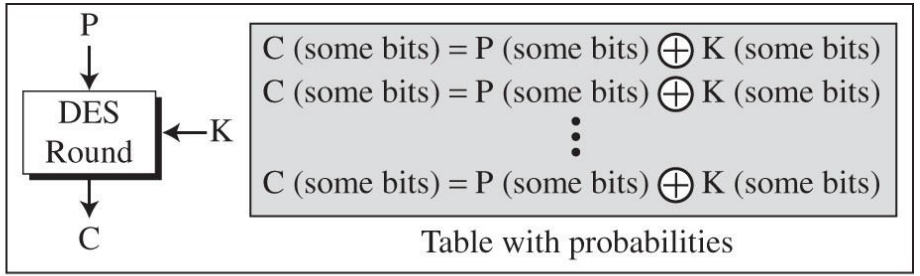
- Linear cryptanalysis first defined by Matsui and Yamagishi in 1992. It was extended Matsui later in 1993 published a linear attack on DES.
- Linear cryptanalysis is a known plaintext attack in which cryptanalyst access larger plaintext and ciphertext messages along with an encrypted unknown key.

Reference 1: section 6.2.6 Differential and Linear Cryptanalysis

Linear Cryptanalysis



a. Linearity Profile

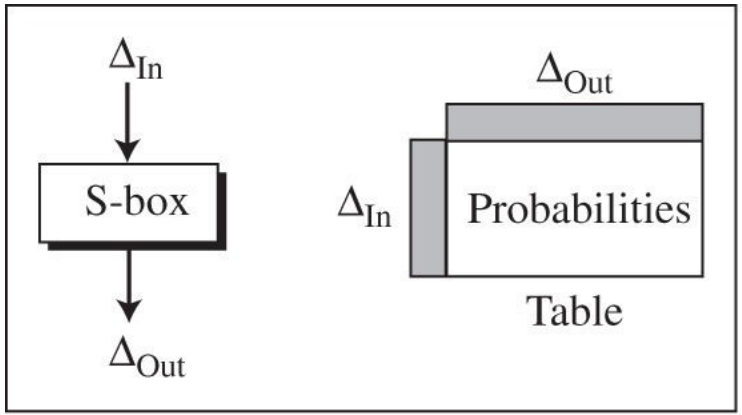


b. Round Characteristic

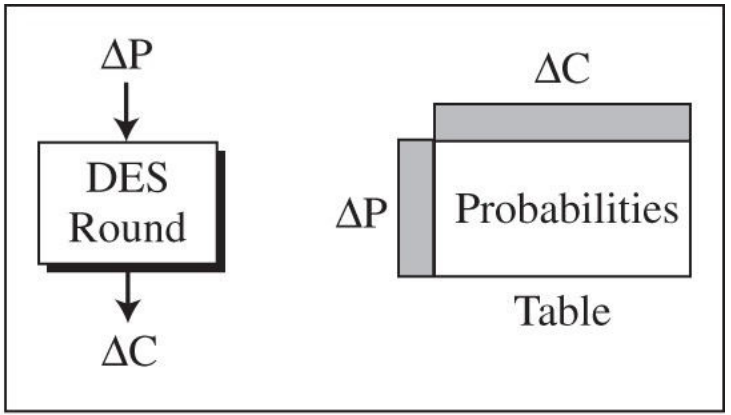
Differential Cryptanalysis

- is a method for breaking certain classes of cryptosystems. It was invented in 1990 by Israeli researchers Eli Biham and Adi Shamir.
- is available to obtain clues about some bits of the key, thereby shortening an exhaustive search. By analyzing the changes in some chosen plaintexts, and the difference in the outputs resulting from encrypting each one, it is possible to recover some properties of the key.

Differential Cryptanalysis



a. Differential Profile



b. Round Characteristic

For example, assume that the ciphertext obtained from one exclusive-or operation of plain text and key.

Without knowing the value of the key, the cryptanalyst can easily find the differences between plaintext and ciphertext. Plaintext difference is represented by P1⊕P2.

Whereas the ciphertext difference represented by C1⊕C2. The following proves that C1⊕C2 =P1⊕P2 First ciphertext C1 obtained = First plaintext P1 ⊕ Key K

Second ciphertext C2 obtained = Second plaintext P2 ⊕ Key K, if C1 and C2 obtained from XORing P1 and P2 and using Key K, can be represented by,

C1⊕C2=P1⊕K⊕P2⊕K=P1⊕P2

Difference between Linear and differential cryptanalysis

- Linear cryptanalysis
  first defined by Matsui and Yamagishi in 1992.
- The cryptanalyst decrypts each cipher text using all possible sub keys for one round of encryption and studies the resulting intermediate cipher text to analyze the random result

- Differential cryptanalysis is a method for breaking certain classes of crypto systemsis invented in 1990 by Israeli researchers Eli Biham and Adi Shamir.
- Cryptanalyst studies changes to the intermediate cipher text obtained between multiple rounds of encryption. The attacks can be combined, which is called differential-linear cryptanalysis.

Difference between Linear and differential cryptanalysis

- In linear cryptanalysis, the role of cryptanalyst is to identify the linear relation between some bits of the plaintext, some bits of the ciphertext and some bits of the unknown key
- Linear cryptanalysis focus on statistical analysis against one round of decrypted cipher text

- By analyzing the changes in some chosen plaintexts, and the difference in the outputs resulting from encrypting each one, it is possible to recover some of the key.
- Differential analysis focuses on statistical analysis of two inputs an two outputs of a cryptographic algorithm.

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

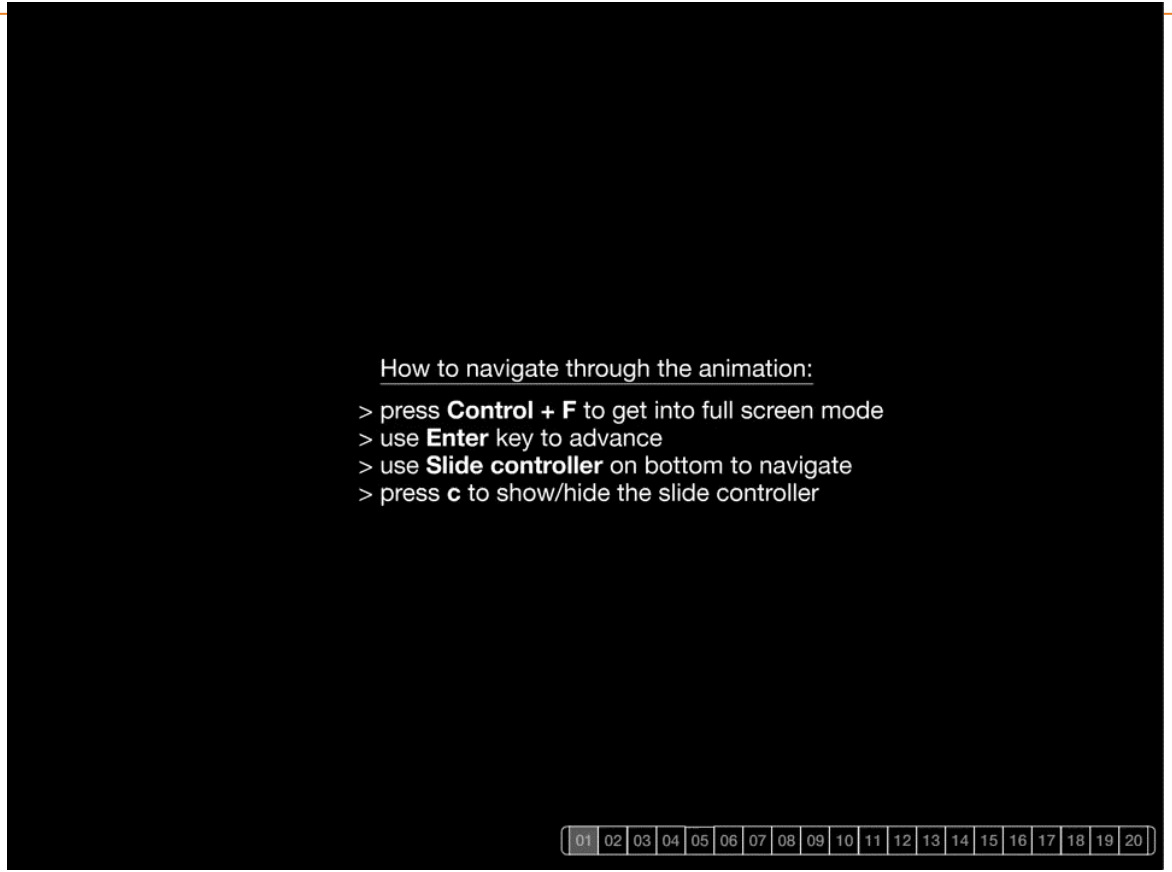
Private keys systems

Lecture 8

AES

Advanced Encryption Standard

AES animation



Origins

- A replacement for DES was needed
  - Key size is too small
- Can use Triple-DES – but slow, small block
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug 99

AES Competition Requirements

- Private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Provide full specification & design details
- Both C & Java implementations

AES Shortlist

- After testing and evaluation, shortlist in Aug-99
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin

- Found contrast between algorithms with
  - few complex rounds versus many simple rounds
  - Refined versions of existing ciphers versus new proposals

The AES Cipher - Rijndael

- Rijndael was selected as the AES in Oct-2000
  - Designed by Vincent Rijmen and Joan Daemen in Belgium
  - Issued as FIPS PUB 197 standard in Nov-2001

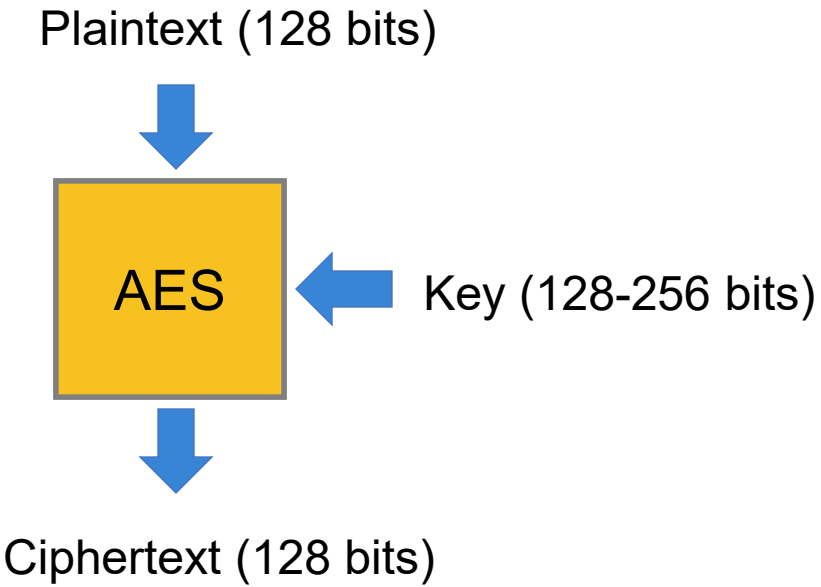
- An **iterative** rather than **Feistel** cipher
  - processes data as block of 4 columns of 4 bytes (128 bits)
  - operates on entire data block in every round

- Rijndael design:
  - simplicity
  - has 128/192/256 bit keys, 128 bits data
  - resistant against known attacks
  - speed and code compactness on many CPUs

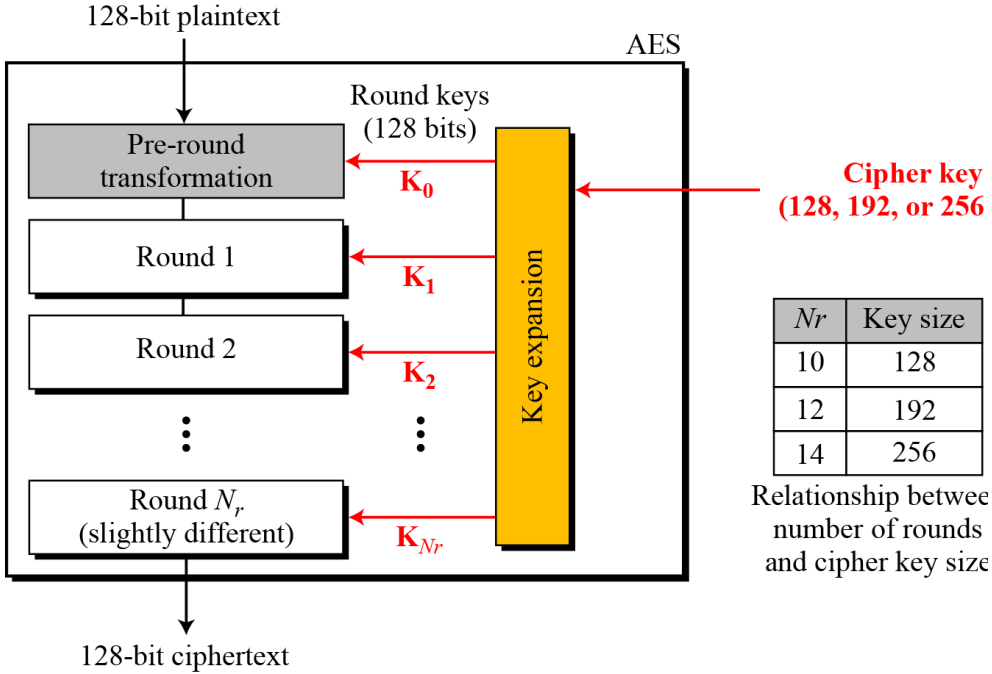
V. Rijmen

J. Daemen

AES Conceptual Scheme



Multiple rounds



Nr	Key size
10	128
12	192
14	256

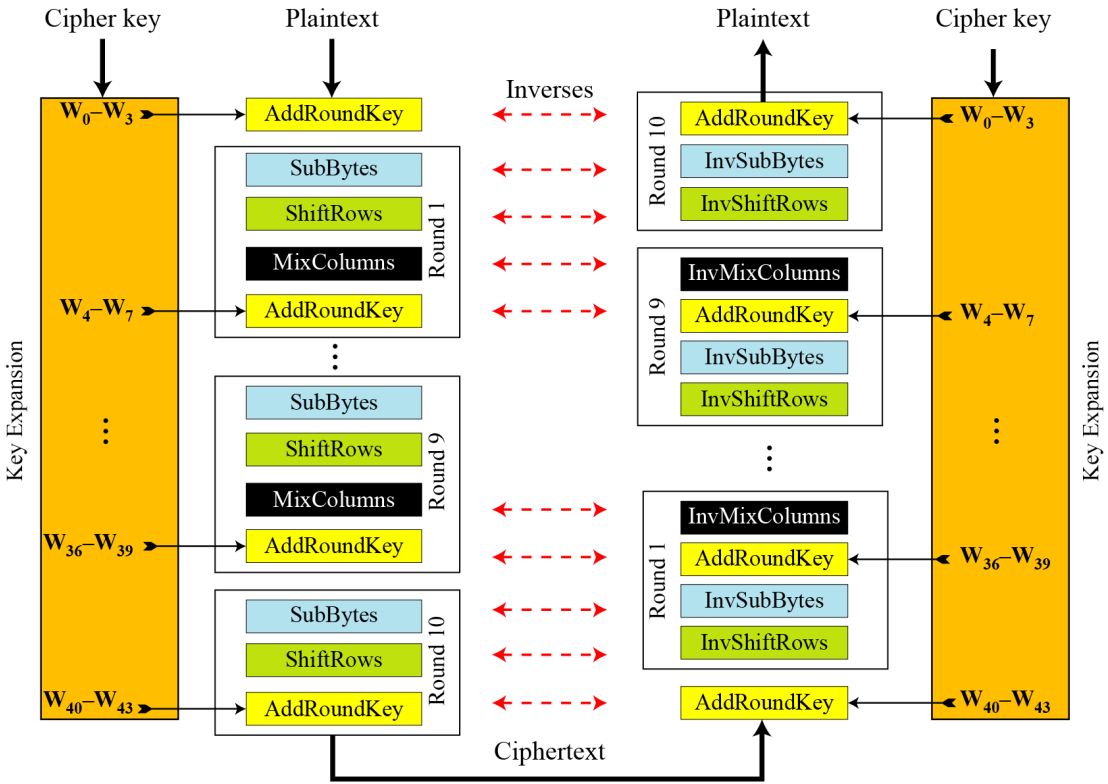
Relationship betwee
number of rounds
and cipher key size

- Rounds are (almost) identical
- First and last round are a little different

High Level Description

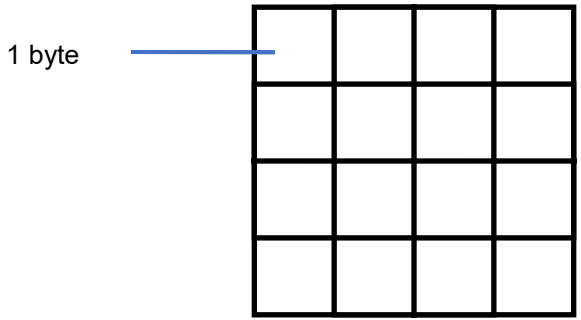
- Key Expansion: Round keys are derived from the cipher key using Rijndael's key schedule
- Initial Round: AddRoundKey : Each byte of the state is combined with the round key using bitwise xor
- Rounds
  - SubBytes : non-linear substitution step
  - ShiftRows : transposition step
  - MixColumns : mixing operation of each column.
  - AddRoundKey
- Final Round:
  - SubBytes
  - ShiftRows
  - AddRoundKey

Overall Structure

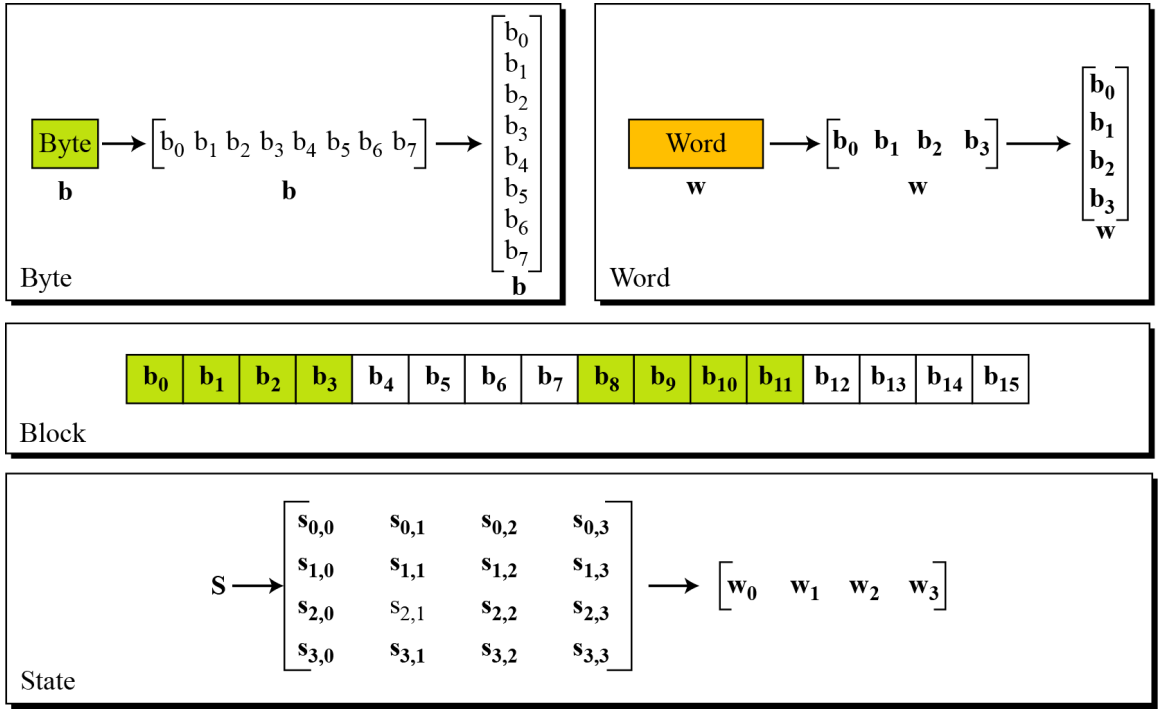


128-bit values

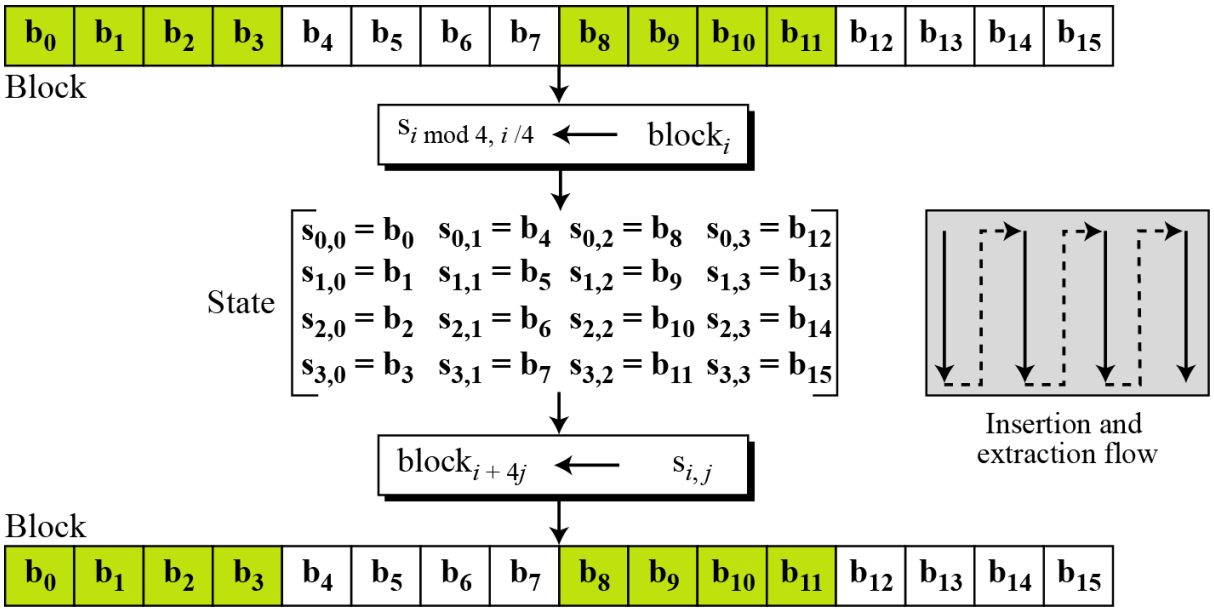
- Data block viewed as 4-by-4 table of bytes
- Represented as 4 by 4 matrix of 8-bit bytes.
- Key is expanded to array of 32 bits words



Data Unit



Unit Transformation

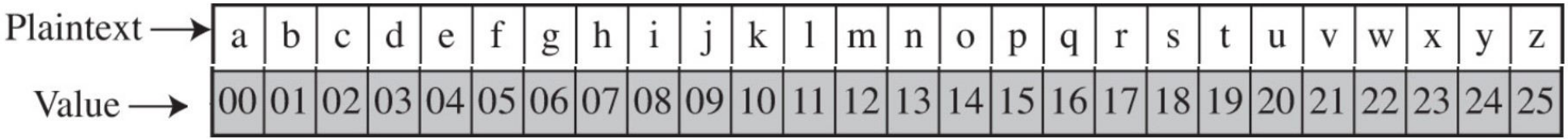


AES

Plaintext: AES USES MATRIX = 14 characters = 14*8 = 112

Converting plaintext to state which is 128 bits. 128-112 = 16bits needed

So add 2 extra character



Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

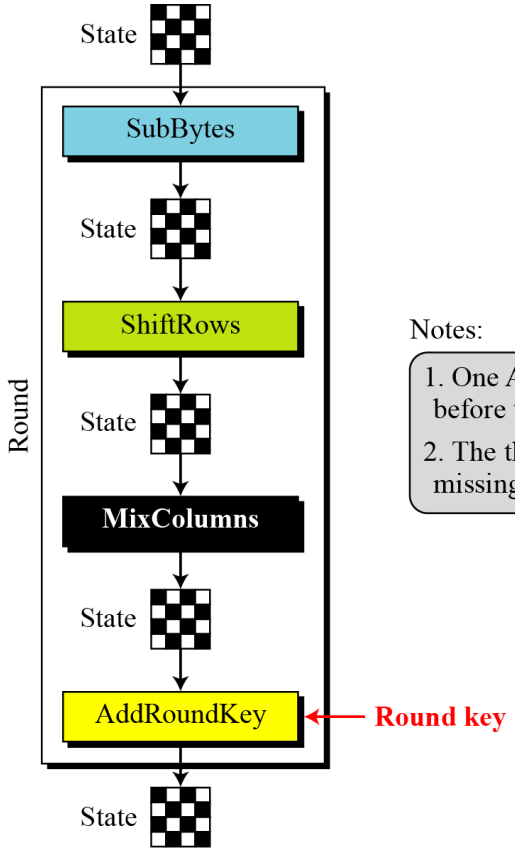
Changing Plaintext to State

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	17	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 17 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

Reference 2: example 7.1

Details of Each Round



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

Reference 2: figure 7.5

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Lecture 21

AES

Advanced encryption standard

Plaintext: AES USES MATRIX

- State

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	17	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 17 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$ State

SubBytes: Byte Substitution

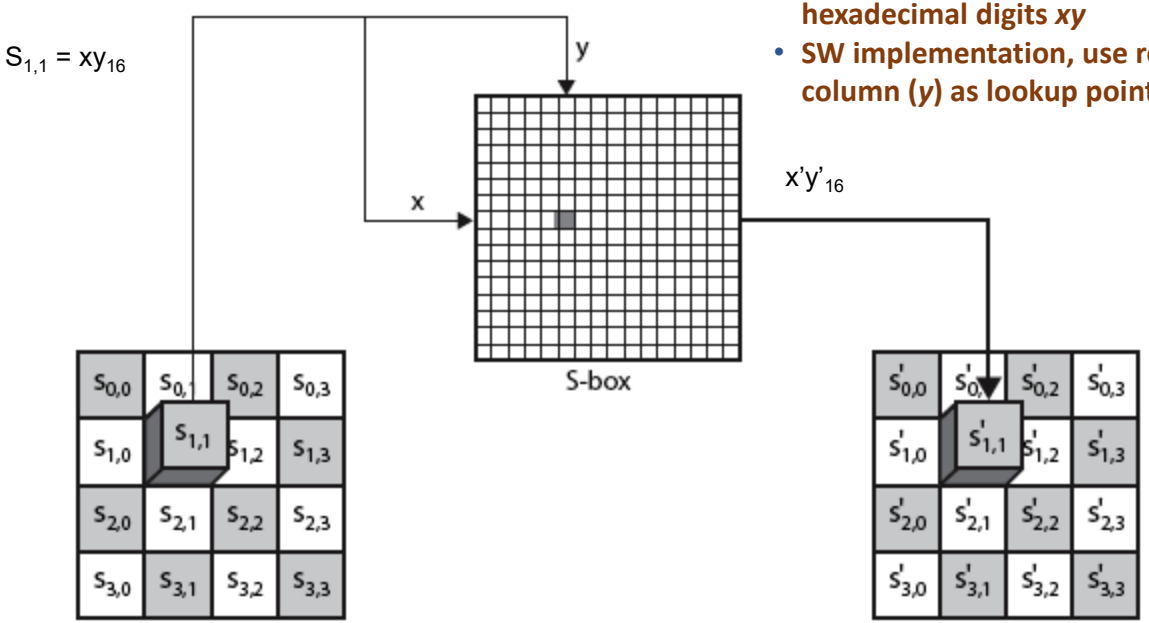
- A simple substitution of each byte
  - provide a confusion

- Uses one S-box of 16x16 bytes containing a permutation of all 256 8-bit values

- Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
- S-box constructed using defined transformation of values in Galois Field- GF(2^8)

SubBytes Operation

- The SubBytes operation involves 16 independent byte-to-byte transformations.



- **Interpret the byte as two hexadecimal digits *xy***
- **SW implementation, use row (*x*) and column (*y*) as lookup pointer**

Reference 2: figure 7.6

SubBytes Table

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 17 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$

AFTER SUBBYTES

		y														
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

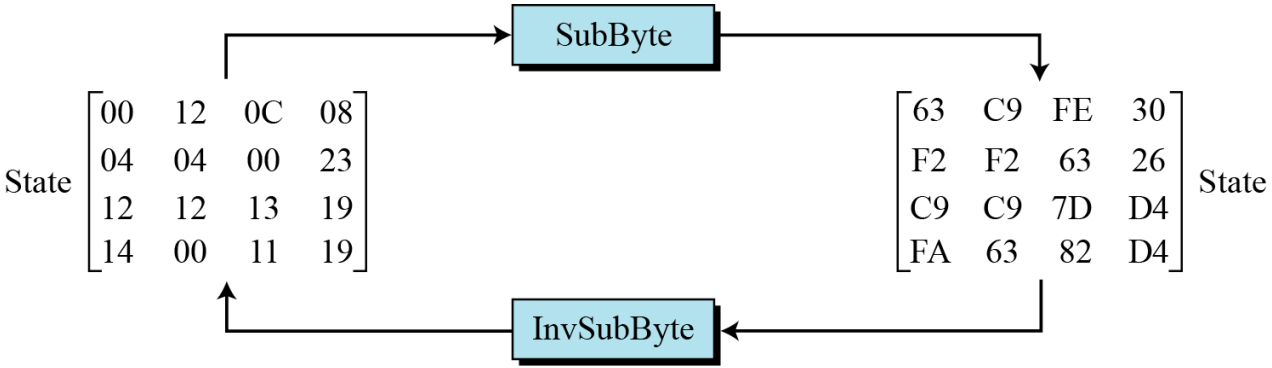
Reference 2: page197

InvSubBytes Table

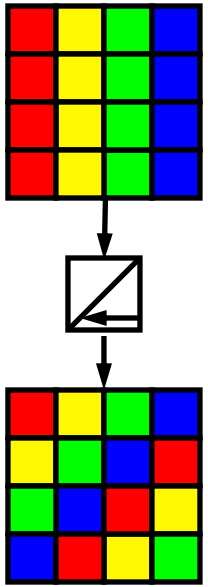
$$\begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}$$

		y														
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
x	**0**	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Sample SubByte Transformation

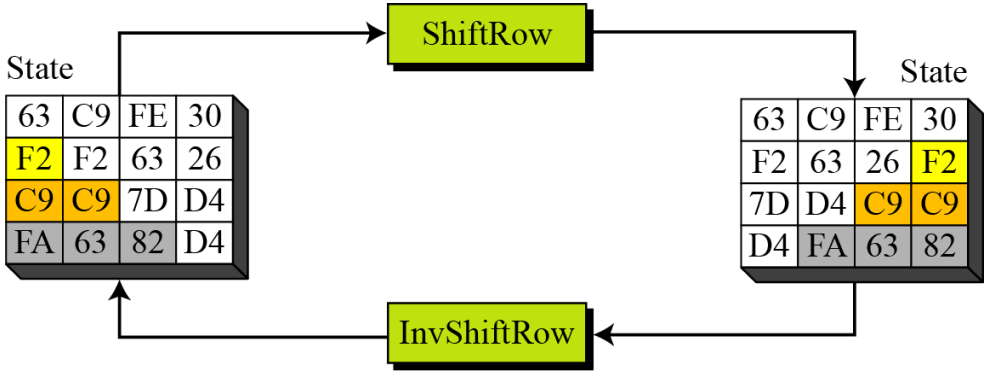


ShiftRows



- Shifting, which permutes the bytes.
- A circular byte shift in each each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- In the encryption, the transformation is called ShiftRows
- In the decryption, the transformation is called InvShiftRows and the shifting is to the right

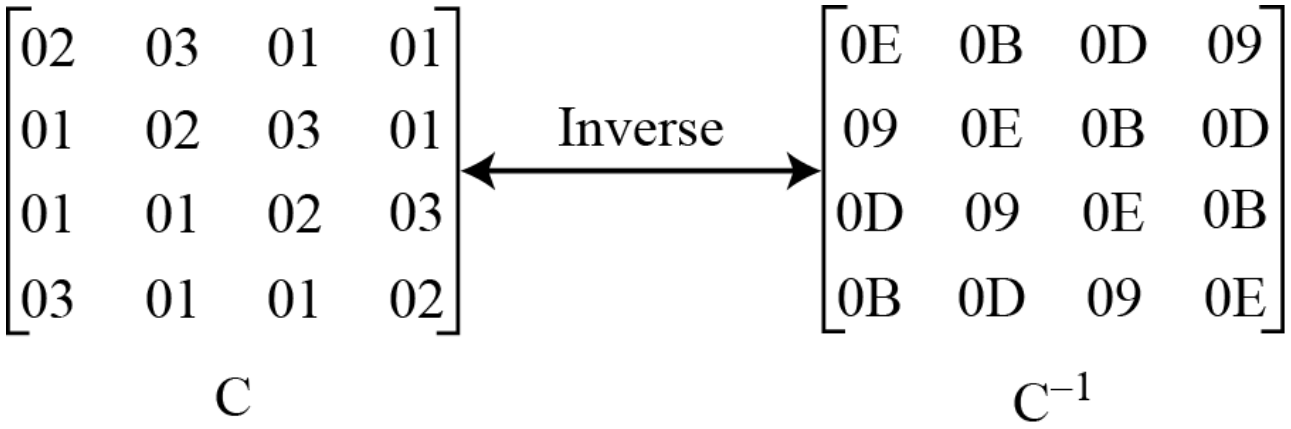
ShiftRows and InvShiftRows



MixColumns

- ShiftRows and MixColumns provide diffusion to the cipher
- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column
- Effectively a matrix multiplication in GF(2^8) using prime poly m(x) =$x^8+x^4+x^3+x+1$

MixColumn and InvMixColumn



63
f2 . [02 03 01 01]
7d
d4

Next topic

- Add round key
- AES security

Next Class

- ☞ Mandatory reading for the next class
  - ☞ https://en.wikipedia.org/wiki/AES_key_schedule#:~:text=AES%20uses%20a%20key%20schedule,keys%20from%20the%20initial%20key.

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things

PESU Center for
Information Security,
Forensics and
Cyber Resilience

Welcome to

PES University

Ring Road Campus, Bengaluru

PESU Center for
Information Security,
Forensics and
Cyber Resilience

PES
UNIVERSITY

APPLIED CRYPTOGRAPHY

Lecture 10

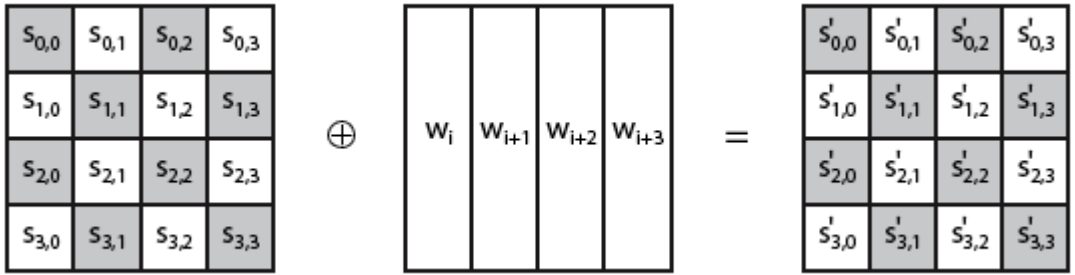
AES key scheduling

Subkey generation

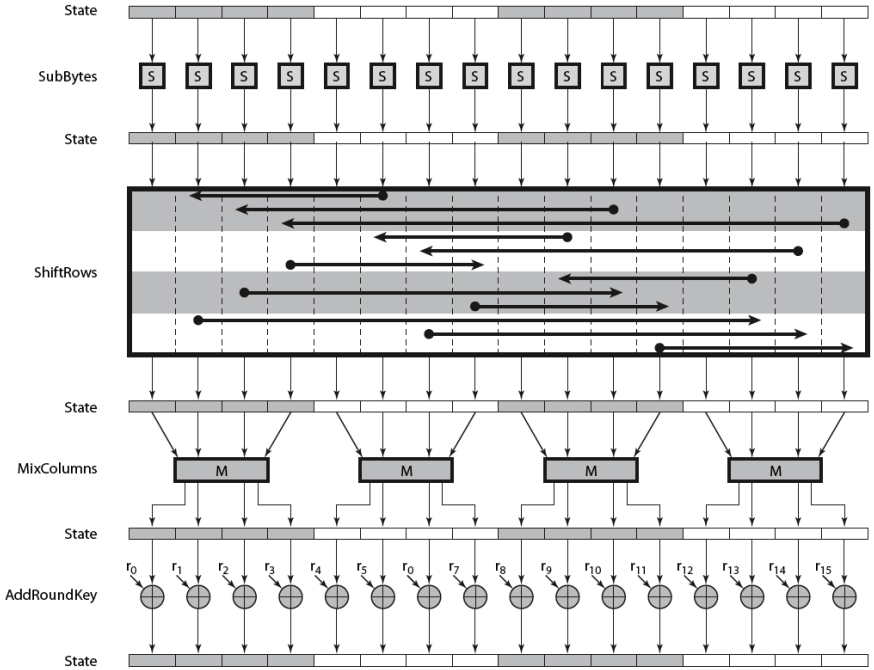
AddRoundKey

- XOR state with 128-bits of the round key
- AddRoundKey proceeds one column at a time.
  - adds a round key word with each state column matrix the operation is matrix addition
- Designed to be as simple as possible

AddRoundKey Scheme



AES Round

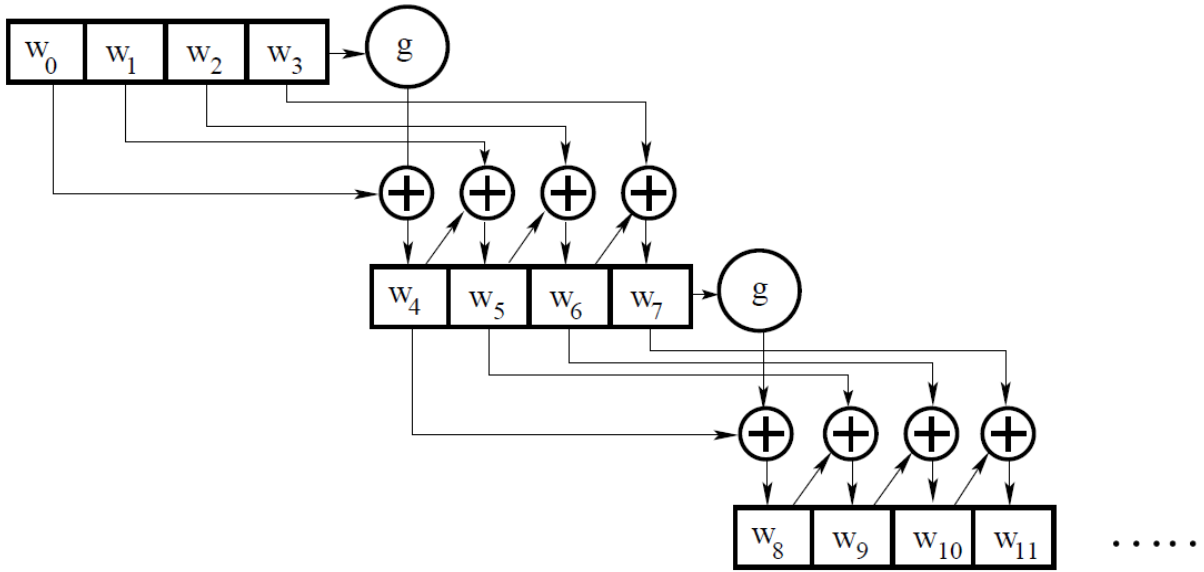

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
State

AES Key Scheduling

- takes 128-bits (16-bytes) key and expands into array of 44 32-bit words

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

Key generation



Rcon

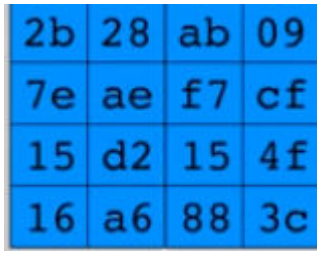
$$rc_i = \begin{cases} 1 & \text{if } i = 1 \\ 2 \cdot rc_{i-1} & \text{if } i > 1 \text{ and } rc_{i-1} < 80_{16} \\ (2 \cdot rc_{i-1}) \oplus 11B_{16} & \text{if } i > 1 \text{ and } rc_{i-1} \geq 80_{16} \end{cases}$$

Values of rc_i in hexadecimal

i	1	2	3	4	5	6	7	8	9	10
rc_i	01	02	04	08	10	20	40	80	1B	36

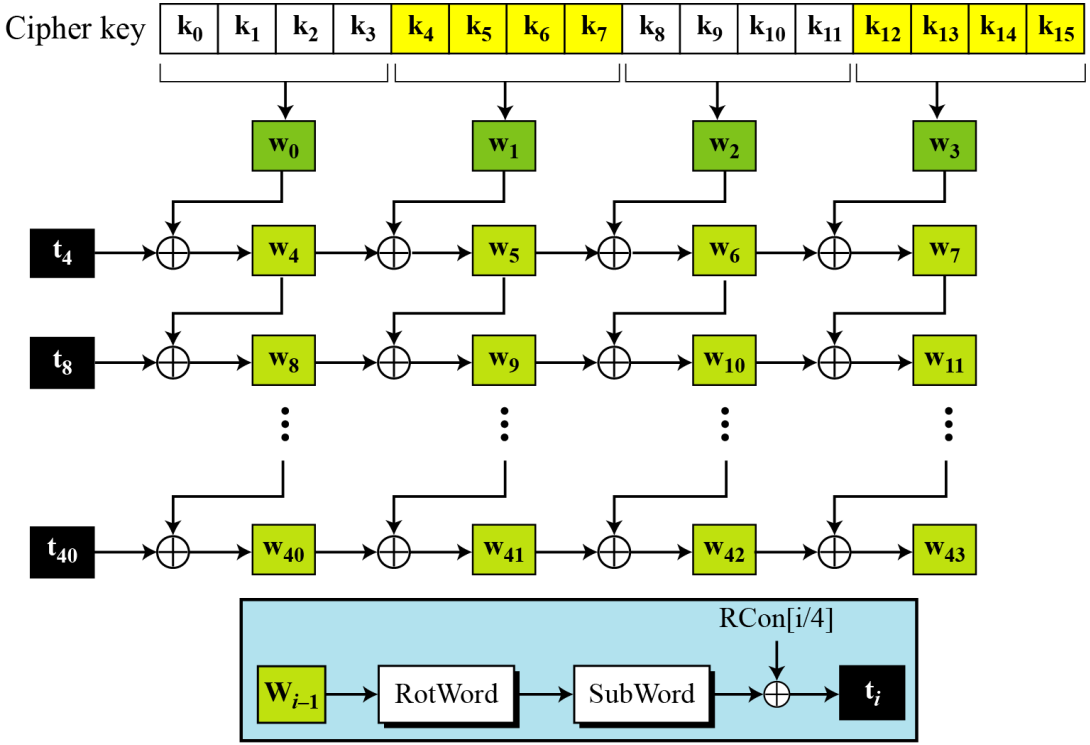
$$W_i = \begin{cases} K_i & \text{if } i < N \\ W_{i-N} \oplus \text{SubWord}(\text{RotWord}(W_{i-1})) \oplus rcon_{i/N} & \text{if } i \geq N \text{ and } i \equiv 0 \pmod{N} \\ W_{i-N} \oplus \text{SubWord}(W_{i-1}) & \text{if } i \geq N,\ N > 6, \text{ and } i \equiv 4 \pmod{N} \\ W_{i-N} \oplus W_{i-1} & \text{otherwise.} \end{cases}$$

AES key scheduling example



2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Key Expansion Scheme



Making of t_i (temporary) words $i = 4\ N_r$

Key Expansion Example (1st Round)

- Example of expansion of a 128-bit cipher key

  Cipher key = 2b7e151628aed2a6abf7158809cf4f3c

  w0=2b7e1516 w1=28aed2a6 w2=abf71588 w3=09cf4f3c

i	w_{i-1}	RotWord	SubWord	Rcon[i/4]	t_i	w[i-4]	w_i
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17	-	-	-	-	28aed2a6	88542cb1
6	88542cb1	-	-	-	-	Abf71588	23a33939
7	23a33939	-	-	-	-	09cf4f3c	2a6c7605

AES Security

- AES was designed after DES.
- Most of the known attacks on DES were already tested on AES.
- Brute-Force Attack
  - AES is definitely more secure than DES due to the larger-size key.
- Statistical Attacks
  - Numerous tests have failed to do statistical analysis of the ciphertext
- Differential and Linear Attacks
  - There are no differential and linear attacks on AES as yet.

Implementation Aspects

- The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory.

- Very efficient

- Implementation was a key factor in its selection as the AES cipher

- AES animation:
  - http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf

Thank you

Next Class

- ☞ Mandatory reading for the next class
  - ☞ https://seedsecuritylabs.org/Labs_16.04/Crypto/Crypto_Encryption/

S Rajashree

Computer Science and Engineering

PES University, Bengaluru

PESU Center for Information Security, Forensics and Cyber Resilience

PESU IoT PESU Center for Internet of Things