# Encryption Algorithms & Protocols

## Symmetric key Crypto
- Block cipher

Dr. Omar Abusada
E-mail: abossada1@gmail.com

# Block Cipher (Iterated)

- Plaintext and Ciphertext consist of fixed-sized blocks.
- Ciphertext obtained from plaintext by iterating a round function.
- Input to round function consists of key and output of previous round.
- Usually implemented in software.

# Feistel Cipher: Encryption

- Feistel cipher is a type of block cipher, not a specific block cipher.
- Split plaintext block into left and right halves: $P = (L_0 , R_0)$
- For each round $i = 1,2,3, \ldots \ldots \ldots \ldots \ldots \ldots, n$ *compute*:
- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, where F is round function and $K_i$ is **subkey**
- Ciphertext: $C = (L_n , R_n)$.

# Feistel Cipher: Decryption

- Start with Ciphertext: $C = (L_n , R_n)$.
- For each round $i = n, n-1, \ldots\ldots\ldots\ldots\ldots, 2,1 compute$:
- $R_{i-1}$=$L_i$
- $L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$, where F is round function and $K_i$ is **subkey**
- Plaintext: $P = (L_0 , R_0)$.
- Formula "works" for any function F
- But only secure for certain functions F

# Feistel Cipher: Example

- Plain text [0 1 1 1 1 0 1 0 0 0 0 1].
- key = [1st to 2nd -- 2nd to 3rd -- 3rd to 1st ]
- $L_0 = [0\ 1\ 1\ 1\ 1\ 0]$, $R_0 = [\underline{1\ 0\ 0}\ \ \underline{0\ 0\ 1}]$,
- **1st iteration :**
- $L_1 = R_0 = [1\ 0\ 0\ 0\ 0\ 1]$, $R_0 = [1\ 0\ 0\ 0\ 0\ 1]$,
- $F(R_0, K_1)$= $F(R_0, K_1)$=[0 1 0  1 0 0 ]  **[1st to 2nd -- 2nd to 3rd -- 3rd to 1st ]**
- $R_1 = L_0 \oplus F(R_0, K_1) = [0\ 1\ 1\ 1\ 1\ 0] \oplus [0\ 1\ 0\ 1\ 0\ 0] = [0\ 0\ 1\ 0\ 1\ 0]$
- **Ciphertext= [1 0 0 0 0 10 0 1 0 1 0]**

**Same procedure can go to the 2nd , 3rd and so on iterations**

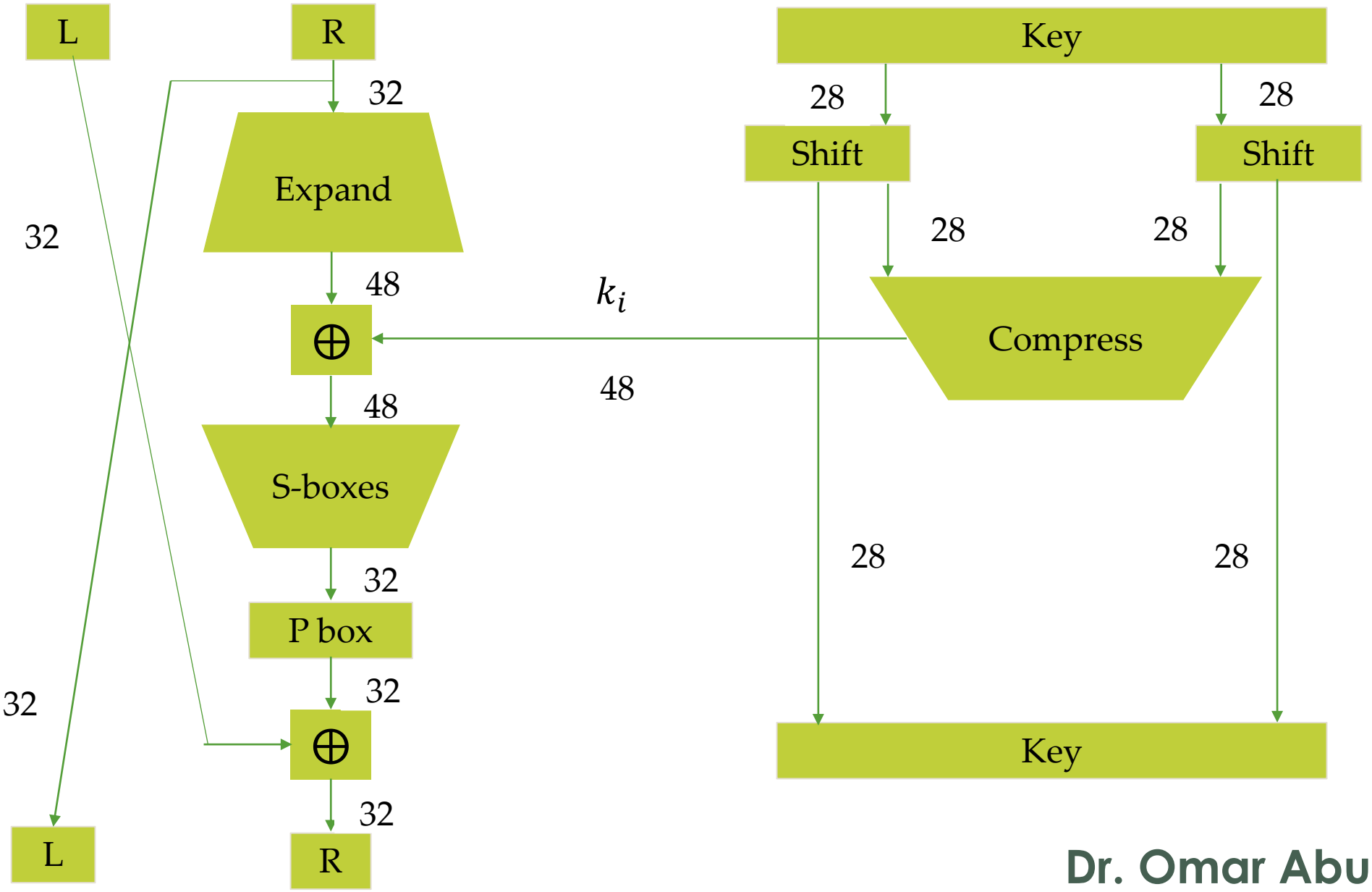
# Data Encryption Standard

- DES developed in 1970's.
- Based on IBM's Lucifer cipher.
- DES was U.S. government standard.
- DES development was controversial.
  - NSA secretly involved.
  - Design process was secret.
  - Key length reduced from 128 to 56 bits.
  - Subtle changes to Lucifer algorithm.

# DES Numerology

- DES is a Feistel cipher with…
  - 64 bit block length.
  - 56 bit key length.
  - 16 rounds.
  - 48 bits of key used each round (Subkey).
- Each round is simple (for a block cipher).
- Security depends heavily on “S-boxes”.
  - Each S-boxes maps 6 bits to 4 bits.

# One Round of DES



# Triple DES

- Today, 56 bit DES key is too small.
  - Exhaustive key search is feasible.
- But DES is everywhere, so what to do?
- Triple DES or 3DES (112 bit key).
  - $\boldsymbol{C = E(D(E(P, K_1), K_2), K_1)}$.
  - $\boldsymbol{P = D(E(D(C, K_1), K_2), K_1)}$.
- Encrypt-Decrypt-Encrypt with 2 keys

# Advanced Encryption Standard AES

- The AES algorithm, also known as (Rijndael Algorithm) is a symmetric block cipher algorithm that takes block size of 28 bits and converts them into ciphertext using Key of 128, 192 or 256 bits (independent of block size)
- AES performs on byte data, instead of bit data.
- Number of rounds depends on Key length
  - 128 bit Key length_____ uses _____ 10 rounds
  - 192 bit Key length_____ uses _____ 12 rounds
  - 256 bit Key length_____ uses _____ 14 rounds

# DES vs AES

## DES

- Key length 56 bits
- Block size 64 bits
- Fixed number of rounds (16)
- Implemented slower

## AES

- Key length 128/192/256 bits
- Block size 128 bits
- Number of rounds dependent on key length
- Implemented faster

# A Few Other Block Ciphers

- **International Data Encryption Algorithm ( IDEA)**
- Invented by James Massey
- One of the giants of modern crypto
- IDEA has 64-bit block, 128-bit key
- IDEA uses mixed-mode arithmetic
- Combine different math operations
  - IDEA the first to use this approach
  - Frequently used today

# Blowfish

- Blowfish encrypts 64-bit blocks
- Key is variable length, up to 448 bits
- Invented by Bruce Schneier
- Almost a Feistel cipher
- $Ri = Li\text{-}1 \oplus Ki$
- $Li = Ri\text{-}1 \oplus F(Li\text{-}1 \oplus Ki)$
- The round function F uses 4 S-boxes
  - Each S-box maps 8 bits to 32 bits
- Key-dependent S-boxes
  - S-boxes determined by the key

# … Thank you …