

1-Mar-04 (1)

# CEG5010 Data Encryption Standard (DES)

1-Mar-04 (2)

## History

- Based on Lucifer developed at IBM in early 70's
- National Security Agency
  - reduced key size from 128 to 64 bits
  - modified the S-boxes (why?)
  - always been speculation added a trapdoor (but none has ever been found)

1-Mar-04 (3)

## Adoption

- Made federal standard in 1976 for unclassified government communications
- ANSI Standard in 1981 (Data encryption algorithm - DEA)
  - retail and wholesale banking
  - networks
- Believed to be the most widely use cryptosystem in the world

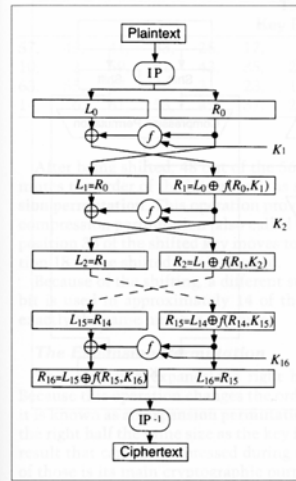
1-Mar-04 (4)

## Algorithm

- Encrypts data in 64 bit blocks
- Uses 64 bit key but every 8th bit ignored

1-Mar-04 (5)

## DES block diagram



1-Mar-04 (6)

## Key schedule (PC-1)

- Parity bits are discarded, reducing the key to 56 bits
- Key schedule  $K[i]$  is computed by
  - Perform the PC-1 permutation (bit1=msb)
  - For  $i=1$  to 15
    - Split result into 2 halves,  $C[0]$  and  $D[0]$
    - $C[i+1] = C[i]$  rol  $x$ ,  $D[i+1] = D[i]$  rol  $x$
    - Perform PC-2 on  $C[i]D[i]$  to get a 48 bit  $K[i]$

1-Mar-04 (7)

## Key schedule (PC-1 & shifts)

- PC-1 (56 bit output)

```

57 49 41 33 25 17  9
 1 58 50 42 34 26 18
 0  2 59 51 43 35 27
19 11  3 60 52 44 36
63 55 47 39 31 23 15
 7 62 54 46 38 30 22
14  6 61 53 45 37 29
21 13  5 28 20 12  4

```

- Rotate left by

```

Iter#  1 2 3 4 5 6 7 8
rol    1 1 2 2 2 2 2 2

```

```

Iter#   9 10 11 12 13 14 15 16
rol    1 2  2  2  2  2  2  1

```

- `c1x<=c0x((c0x'left + 1) to c0x'right) & c0x(c0x'left to (c0x'left + 0));`

1-Mar-04 (8)

## Key schedule PC-2

- Compression permutation PC-2

```

14 17 11 24  1  5
 3 28 15  6 21 10
23 19 12  4 26  8
16  7 27 20 13  2
41 52 31 37 47 55
30 40 51 45 33 48
44 49 39 56 34 53
46 42 50 36 29 32

```

- Has a 48 bit output

1-Mar-04 (9)

## Encrypting a block of data

- Initial permutation (IP)
- Split block into 2 halves L[0] and R[0]
- for  $i=1$  to 16
  - Expansion function (E)
  - xor with  $K[i]$
  - apply S-boxes bits b1 and b6 are rows, b2-b5 are columns in the table
  - Permute output with P
  - xor resulting value with L[i-1] to get R[i]
  - $L[i] = R[i-1]$
- Apply final permutation  $IP^{-1}$

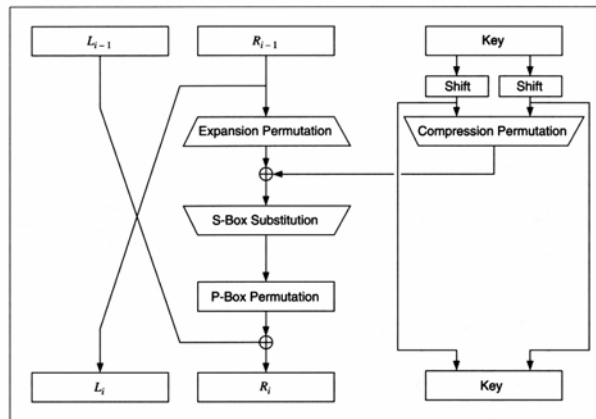
1-Mar-04 (10)

## Decrypting a block of data

- Same as encryption but substitute  $K[i]$  for  $K[17-i]$

1-Mar-04 (11)

## One round of DES



1-Mar-04 (12)

## Initial and final permutation

- IP

```

58 50 42 34 26 18 10  2
60 52 44 36 28 20 12  4
62 54 46 38 30 22 14  6
64 56 48 40 32 24 16  8
57 49 41 33 25 17  9  1
59 51 43 35 27 19 11  3
61 53 45 37 29 21 13  5
63 55 47 39 31 23 15  7

```

- IP-1

```

40  8 48 16 56 24 64 32
39  7 47 15 55 23 63 31
38  6 46 14 54 22 62 30
37  5 45 13 53 21 61 29
36  4 44 12 52 20 60 28
35  3 43 11 51 19 59 27
34  2 42 10 50 18 58 26
33  1 41  9 49 17 57 25

```

1-Mar-04 (13)

# IP VHDL

```

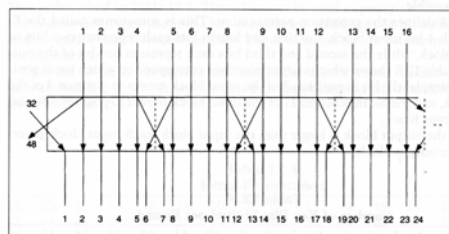
entity ip is port
(
  pt      :      in
  std_logic_vector(1 TO 64);
  l0x     :      out
  std_logic_vector(1 TO 32);
  r0x     :      out
  std_logic_vector(1 TO 32)
);
end ip;

architecture behaviour of ip is
begin
  l0x(1)<=pt(58);
  l0x(2)<=pt(50);
  l0x(3)<=pt(42);
  l0x(4)<=pt(34);
  l0x(5)<=pt(26);
  l0x(6)<=pt(18);
  ...
  r0x(25)<=pt(63);
  r0x(26)<=pt(55);
  r0x(27)<=pt(47);
  r0x(28)<=pt(39);
  r0x(29)<=pt(31);
  r0x(30)<=pt(23);
  r0x(31)<=pt(15);
  r0x(32)<=pt(7);
end behaviour;

```

1-Mar-04 (14)

# Expansion function



# Expansion function and P permutation

- Expansion function

```

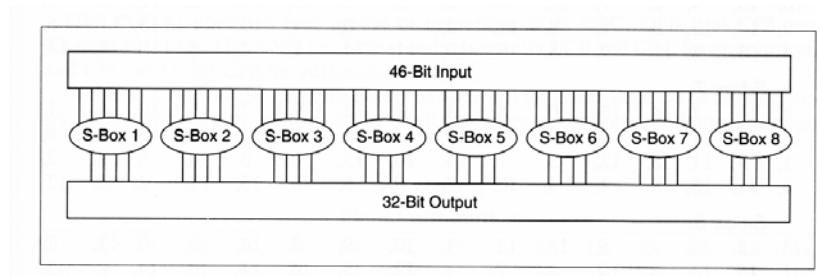
32  1  2  3  4  5
   4  5  6  7  8  9
   8  9 10 11 12 13
  12 13 14 15 16 17
  16 17 18 19 20 21
  20 21 22 23 24 25
  24 25 26 27 28 29
  28 29 30 31 32  1
    
```

- P permutation

```

16  7 20 21
29 12 28 17
 1 15 23 26
 5 18 31 10
 2  8 24 14
32 27  3  9
19 13 30  6
22 11  4 25
    
```

# S-boxes



1-Mar-04 (17)

# Substitution box (S-box)

Substitution Box 1 (S[1])

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

S[3]

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

S[2]

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

S[4]

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

1-Mar-04 (18)

# S-box

S[5]

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

S[7]

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

S[6]

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

S[8]

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

1-Mar-04 (19)

## S-box VHDL

```

architecture behaviour of s1 is
...
  attribute init: string;
  attribute init of u1: label is "86e67619";
...
  attribute init of u8: label is "917be906";

  signal
    lower1,upper1,lower2,upper2,lower3,upper3,lowe
    r4,upper4 : std_logic;
  signal d1,d2,d3,d4 : std_logic;

...
  u9: m2_1 port map (
    d0=>lower1, d1=>upper1, s0=>i(1),
    o=>d1 );
  u10: m2_1 port map (
    d0=>lower2,
    d1=>upper2, s0=>i(1), o=>d2 );
  u11: m2_1 port map (
    d0=>lower3,
    d1=>upper3, s0=>i(1), o=>d3 );
  u12: m2_1 port map (
    d0=>lower4,
    d1=>upper4, s0=>i(1), o=>d4 );

end: -- s1

begin
  u1: rom32x1 generic map (
    init=>"1000011011100110011101100001100
  1" ) port map (
    a0=>i(6),
    a1=>i(5), a2=>i(4), a3=>i(3),
    a4=>i(2), o=>lower1 );
  u2: rom32x1 generic map (
    init=>"1000011010011101010010010111101
  0" ) port map (
    a0=>i(6),
    a1=>i(5), a2=>i(4), a3=>i(3),
    a4=>i(2), o=>upper1 );

```

1-Mar-04 (20)

## DES VHDL

```

begin
  Ukeysched: keysched port map
    (key=>key, k1x=>k1x,
    k2x=>k2x, k3x=>k3x, k4x=>k4x,
    k5x=>k5x, k6x=>k6x, k7x=>k7x,
    k8x=>k8x, k9x=>k9x,
    k10x=>k10x, k11x=>k11x,
    k12x=>k12x, k13x=>k13x,
    k14x=>k14x, k15x=>k15x,
    k16x=>k16x);

  Uip: ip port map(pt=>pt,
    10x=>10x, r0x=>r0x );

  round1:roundfuncport map (
    clk=>clk, ce=>ce, li=>10x,
    ri=>r0x, lo=>11x, ro=>r1x,
    k=>k1x);
  ...
  round16:roundfunc port map(
    clk=>clk, ce=>ce, li=>115x,
    ri=>r15x, lo=>116x, ro=>r16x,
    k=>k16x);
  Ufp:fp port map (l=>r16x, r=>116x,
    ct=>ct);

end behaviour;

```

1-Mar-04 (21)

## Modes of operation

- Electronic codebook (ECB)
  - simplest method, each block encrypted independently with key
- Cipher block chaining mode (CBC)
  - plaintext is xor'ed with ciphertext of previous block before it is encrypted
  - $ct_i = \text{DES}(pt_i \text{ xor } ct_{i-1})$  instead of  $ct_i = \text{DES}(pt_i)$
  - much more secure than ECB

1-Mar-04 (22)

## Pipelining

- Can insert latch after every round
- Most previous implementations could not fit all 16 rounds of DES on a chip
  - require 16 cycles to compute DES
- Easy for ECB but cannot use for CBC mode

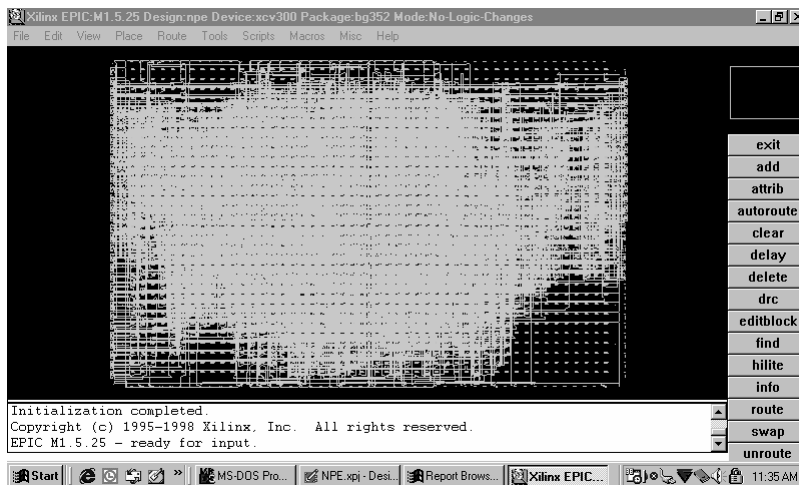
1-Mar-04 (23)

## Our implementation

- Synthesized from VHDL without floorplanning
- On an XCV300-4 FPGA
  - Slices: 1981/3072 (64% slices)
  - Max freq: 34.4 MHz
  - Can perform a 64 bit encryption every cycle (latency 16 cycles)

1-Mar-04 (24)

## FPGA Layout



1-Mar-04 (25)

## Software DES Implementations

Processor	CLOCK	DATA RATE
HP 9000/887	125MHz	1.6 MB/s
Sun Ultra 5	333MHz	4.0 MB/s
Intel Pentium III	750MHz	14.0 MB/s

1-Mar-04 (26)

## VLSI DES Implementations

COMPANY	CHIP	YEAR	CLOCK	DATA RATE	NOTES
AMD	AmZ8068	1982	4MHz	1.7 Mbyte/s	
Western digital	WD2002	1984	3MHz	0.23 Mbyte/s	
CE-Infosys	CE99C003A	1994	30MHz	20 Mbyte/s	
VLSI Technology	6868	1995	32MHz	64 Mbyte/s	
CUHK (core)	XVC300-6	1999	33MHz	264 Mbyte/s	
CUHK (with bus)	XVC300-6	1999	33MHz	1.9 Mbytes/s	
Wilcox	0.6um CMOS	1999	156 MHz	1250 Mbytes/s	
Patterson	XCV150-6	2000	168 MHz	1344 Mbytes/s	floorplanned
Trimberger	XCV300E-8	2000	190 MHz	1500 Mbytes/s	no floorplan

1-Mar-04 (27)

## DES Cracking

- Involves searching  $2^{56}$  (72,057,594,037,927,936) keys
  - no better way has yet been found
- Two successful approaches
  - distributed computing
  - custom DES hardware (note this is a different problem to encryption since we need to be able to change the keys quickly and check the plaintext)

1-Mar-04 (28)

## Distributed computing approach (1998)

- DES-II-1 is a competition from RSA labs to decrypt a message encrypted with DES
- Monday, 23-Feb-1998 a solution was found from a distributed computing effort
  - took 39 days

1-Mar-04 (29)

## Project statistics

- Start of contest: January 13, 1998
- End of Contest: February 23, 1998
- Size of keyspace: 72,057,594,037,927,936
- Approximate keys tested:  
63,686,000,000,000,000
- Peak keys per second: 34,430,460,000

1-Mar-04 (30)

## Equivalent computing power

- Distributed.net is equivalent in processing power to:
  - 11,264 DEC Alpha 21064 533s
  - 15,316 Sun Ultra I 167s
  - 22,393 Intel Pentium II 333s

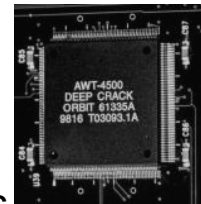
1-Mar-04 (31)

## “Deep crack” Hardware cracker

- Developed by the Electronic Frontier Foundation
- Cost US\$210,000
  - \$80,000 design
  - \$210,000 materials (chips, boards, chassis etc)

1-Mar-04 (32)

## VLSI Chip



- Developed by Advanced Wireless Technologies
  - 24 search units per chip, developed in VHDL
  - 40 MHz
  - 16 cycles per encryption
  - 2.5 million keys/s
  - FPGAs were considered but found to be too expensive

1-Mar-04 (33)

## Board

- Contains 64 chips



1-Mar-04 (34)

## Cabinets

- 6 cabinets holding 29 boards



1-Mar-04 (35)

## Deep crack system

- 90 billion keys/s
  - 37,000 search units
  - c.f. Distributed Net's 34 billion keys/s
- Controlled by PC
  - checks possible all ASCII candidate solutions from the search units
- Solved RSA's DES-III in 22 hours
  - Jan 18 1999

1-Mar-04 (36)

## Alternatives

- Triple DES (56\*3) bit key (ANSI X9.52)
  - $\text{DES}(\text{key3}, \text{DES}^{-1}(\text{key2}, \text{DES}(\text{key1}, \text{pt})))$
  - $\text{key1}=\text{key3}$  (2 key variant)
- Internet task force recommends triple DES, RC2 and RC4
- NIST developing AES (advanced encryption standard) but is many years from approval

## RC5 (Hardware friendly)

- Params:
    - w-bit wordsize (32,64)
    - r-rounds (12,16)
    - b-byte key (16)
  - In: 2w bit plaintext  $M=(A,B)$ ,  $K[0] \dots K[b-1]$
  - Out: 2w bit ciphertext C
  - Notes
    - invented by Rivest 1995
    - “+” is mod  $2^w$
    - only uses XOR, add and rot
    - decryption is inverse
- Encryption

  1. Compute  $K_0 \dots K_{2r+1}$  (w-bit)
  2.  $A=A + K_0; B=B + K_1$
  3. for  $i = 1$  to  $r$  {
    - $A = (A \text{ xor } B) \text{ rot } B + K_{2i};$
    - $B = (B \text{ xor } A) \text{ rot } A + K_{2i+1};$
  4.  $C = (A,B)$

## References

- B. Schneier, “Applied Cryptography”, 2nd Edition, Wiley 1996
- A. Menezes, P. Van Oorschot (Editor), Scott A. Vanstone (Editor), “Handbook of Applied Cryptography”, CRC Press Series on Discrete Mathematics and Its Applications, 1996
- J. Gilmore (Editor), “Cracking DES: Secrets of Encryption Research, Wiretap Politics & Chip Design”, Electronic Frontier Foundation, 1998