EECS 3481

APPLIED CRYPTOGRAPHY

MODERN SYMMETRIC CRYPTO

PROF H ROUMANI
Dept. of Electrical Engineering and Computer Science, York University

1

MODERN VS CLASSICAL

- *Alphabet is {0,1} instead of {A-Z}*
  - Continue with byte[] but as 8 bits, *not* char
- *To encrypt, digitize the PT: string-to-byte[]*
  - Use the getBytes() method in String
- *To display/transmit PT / CT, use a Hex String*
  - Use CryptoTools hexToBytes and bytesToHex
- *After decrypting the CT: byte[]-to-string*
  - Use the new String(byte[]) constructor

Continue classifying Sym/Asym, Stream/Block, Substit/Transpo, Group/Not, AttackTypes, but no more Mono/Poly (why?).

2

2

STREAM CIPHERS

3

OTP [One-Time Pad]

- What
  Vigenère with a random key (as long as PT) used only once
- Definition
  1. |K|=|P|
  2. K is random
  3. y = E(k,x) = k xor x *(bitwise ^)*
  4. K never re-used (hence the O in OTP)
- It boasts perfect secrecy
  Thwarts *exhaustive* attacks even if Eve had *infinite* classical or quantum computing power!

4

4

Example

```
byte[] ky = "Go Leafs Go!!".getBytes();

// Alice:
byte[] pt = "Meet Wed@1:30".getBytes();
byte[] ct = xor(pt, ky);
```

Send the string X = CryptoTools.bytesToHex(ct)

```
// Bob:
```

Receive ct = CryptoTools.hexToBytes(X)

```
byte[] bk = xor(ct, ky);
System.out.println("BACK: " + new String(bk));
```

Method byte[] xor(byte[] a, byte[] b)

```
byte[] result = new byte[a.length];
for (int i = 0; i < a.length; i++) result[i] = (byte) (a[i] ^ b[i]);
```

5

5

Questions About OTP

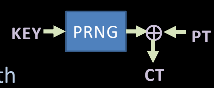
- Is it a stream or a block cipher?
- Does it rely on substitution or transposition?
- Is it a group cipher?
- Is it vulnerable to exhaustive KCA?
- Is it practical? Why shouldn't the key be reused?
- Can we send a new key through it?
- Does it offer content integrity or is it malleable?
- Is it vulnerable to KPA?

6

6

Stream Ciphers

- Practical approximations to OTP
- A short key (concatenated with a counter IV) is used as a seed to a PRNG
- CT = PRNG xor PT stream
- Fast and implementable in hardware. Used in WiFi, GSM, TLS, and Bluetooth
- Strength dependent on PRNG (period, algorithm) and the size / change-frequency of the key.



7

7

WEP (A stream cipher example)

- |K|=104b (26 hex or 13 ASCII), |IV|=24b,
- K || IV (128b) seeds PRNG (RC4)
- IV changed after each packet
- Even at only 11 Mbps, and with 1500B TCP packets, period is only ~5 hours
- Can accelerate by flooding with a replay (ARP)

8

8

PRNG

- **What is Random?**
  Uniformly distributed stats: bits, runs, etc.
  Independence: given a subsequence, cannot predict the next number
- **Linear Congruential Generator**
  $X_{n+1} = (a \times X_n) \bmod m$
  Select the seed X_0 so period is maximal, e.g. $m = 2^{31}-1$, $a = 48271$ or 7^5
- **Why Pseudo?**
  With well-chosen parameters, the sequence is uniform but predictable.

Java.util.Random

9

9

TRNG

- Natural Randomness
  Relies on a non-deterministic entropy source.
  Slow but unpredictable. Ideal for nonce/seed.
  Example: radiation, radio/thermal noise, RdRand
- Symmetric Ciphers or Hash Functions
  Hash or encrypt a counter starting with the seed.
  Unpredictable if the key or the seed is not known.
- Use Asymmetric Cryptography
  BBS: $x_i = x_{i-1}^2$ mod n, n=pq, and primes p,q=3 mod 4
  Unpredictable w/o factoring n. CSPRNG.

java.security.SecureRandom

10

10

BLOCK CIPHERS

11

Block Ciphers

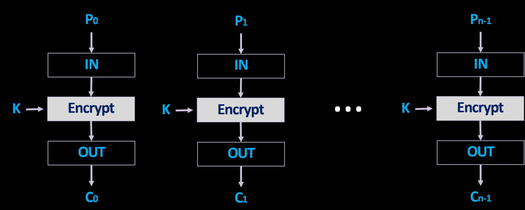
- Block Size (B)
  Not too small (exhaustive) and not too large (practical)
  DES: B=64b, K=56b. AES: B=128b, K=128b.
- Padding
  Use PKCS5 which always appends n = B – (P mod B) bytes to P and stores the number n in each of them (0 < n ≤ B). Note that it pads even if P divides B, and that Bob finds n in MSB.
- Encrypt each Block
  SPN combines substitution and permutation => confusion (the CT/KEY relation) + diffusion (the PT/CT relation). Avalanche effect: 1b flip in PT changes ~½ the CT bits.
- Across Blocks
  Use a so-called mode of operation: ECB, CBC, CTR, …

12

12

ECB Mode of Operation

ECB =Electronic Code Book



13

13

Example 1

```
byte[] pt = 8 bytes
byte[] ky = 8 bytes

Key secret = new SecretKeySpec(ky, "DES");
Cipher cipher = Cipher.getInstance("DES/ECB/NoPadding");
cipher.init(Cipher.ENCRYPT_MODE, secret);

byte[] ct = cipher.doFinal(pt);
```

With no padding, |PT| must be 8B

14

14

Example 2

```
byte[] pt = any number of bytes
byte[] ky = 8 bytes

Key secret = new SecretKeySpec(ky, "DES");
Cipher cipher = Cipher.getInstance("DES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secret);

byte[] ct = cipher.doFinal(pt);
```

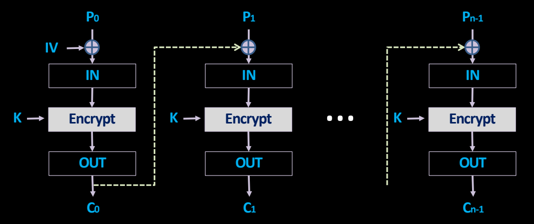
With padding, |PT| can be anything

15

15

CBC Mode of Operation

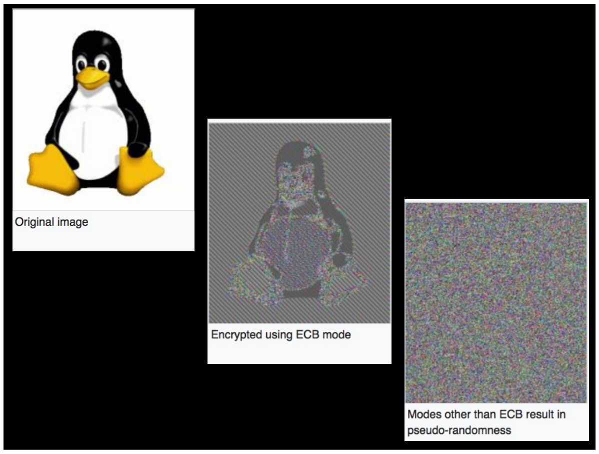
CBC = Cipher-Block Chaining



IV = Initialization Vector is sent in the clear.

16

16



Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness

17

Example 3

```
byte[] pt = any number of bytes
byte[] ky = 8 bytes
byte[] iv = CryptoTools.hexToBytes("0123456701234567");

Key secret = new SecretKeySpec(ky, "DES");
Cipher cipher = Cipher.getInstance("DES/CBC/PKCS5Padding");
AlgorithmParameterSpec aps = new IvParameterSpec(iv);
cipher.init(Cipher.ENCRYPT_MODE, secret, aps);
byte[] ct = cipher.doFinal(pt);
```

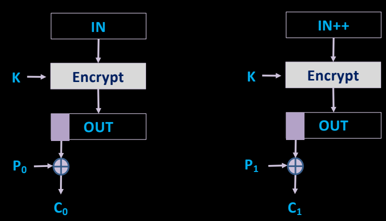
With pkcs5 and the CBC mode of op.
With CBC, an IV (8 bytes) is needed.

18

18

CTR Mode of Operation

CTR = Counter



IV = initial value of the counter **IN**

This mode allows block ciphers to be "streamy"

21

21

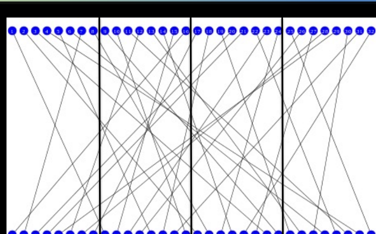
Block Cipher Design

1. S Box
   Substitution via xor with key → very quick (but group)
2. P Box
   Permutation via shift → very quick (but group)
3. Product
   SxP still very quick but not a group if they don't commute
4. Rounds (SP Network)
   No group => repeat with diff subkeys (derived from key)
5. Add "More Confusion" to S
   Non-injective/surjective (DES) or nonlinear function (AES)

22

22

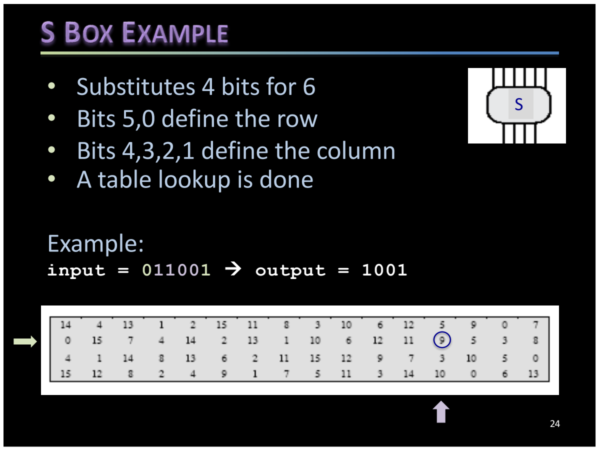
P Box Example



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

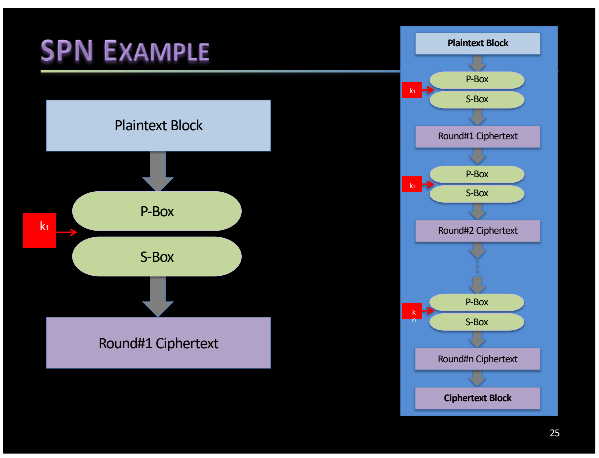
23

23

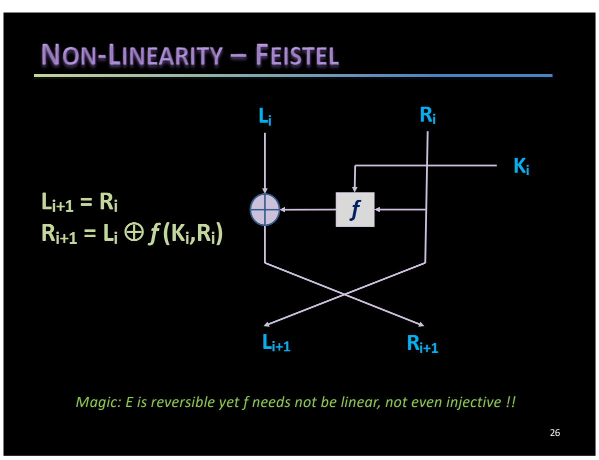


14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

24



25

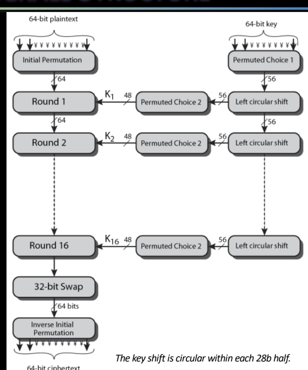


26

DES & AES

27

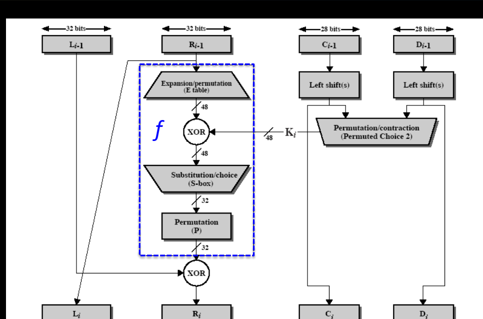
DES Overall Structure



28

28

One Round of DES



(Note the Feistel f Function)

29

29

DES Sub-Key Generation

(a) Input Key

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

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	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

(c) Permuted Choice Two (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

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

30

30

DES IP, E, and P

(a) Initial 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

(b) Inverse Initial Permutation (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

(c) Expansion Permutation (E)

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

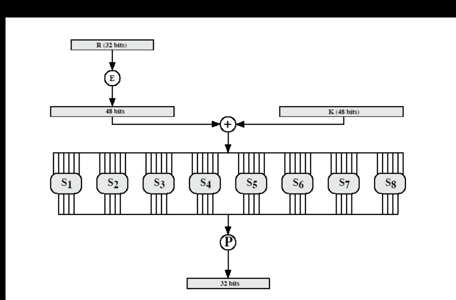
(d) Permutation Function (P)

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

31

31

The DES S Boxes



32

32

The DES S Tables

Example:
In: 011001
Out: 1001

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_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_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_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
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_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_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_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

33

33

Example

K = 0x133457799BBCDFF1
PT = 0x0123456789ABCDEF
Find CT after the first round.

```
      00000000 01111111 11122222 22222333 33333334 44444444 45555555 55566666
      12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234
K   = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001
PC1 = 11110000 11001100 10101010 11110101 01010110 01100111 10001111
C   = 11100001 10011001 01010101 1111  Rotated C [28]
D   = 10101010 11001100 11110001 1110  Rotated D [28]
PC2 = 00011011 00000010 11101111 11111100 01110000 01110010 SUBKEY#1
PT  = 00000001 00100011 01000101 01100111 10001001 10101011 11001101 11101111
IP  = 11001100 00000000 11001100 11111111 11110000 10101010 11110000 10101010
R   = 01111010 00010101 01010101 01111010 00010101 01010101    AFTER E
XOR = 01100001 00010111 10111010 10000110 01100101 00100111
      011000 010001 011110 111010 100001 100110 010100 100111  SAME BUT [6*8]
       0101   1100   1000   0010   1011   0101   1001   0111   AFTER S BOXES
    = 01011100 10000010 10110101 10010111
P   = 00100011 01001010 10101001 10111011 THIS IS f(K, R)
XOR = 11101111 01001010 01100101 01000100 THE NEW R
CT  = 11110000 10101010 11110000 10101010 11101111 01001010 01100101 01000100
```

34

34

Exhaustive Attack

- **Crack Time Estimates**
  - $2^{56} \approx 7\text{x}10^{16}$ is not trivial on a single core (2000 years)
  - But it takes less than a day on today's clusters (17 hours)
- **Key Space Reduction**
  - Typical PT byte ~69 possibilities (52 letters, 10 digits, 1 space, …)
  - Key acceptance probability (69/256 is about ¼ = $(½)^2$ per byte)
  - Assuming byte independence, $(½)^{16}$ per block.
  - Key must also decrypt a second block → 2^{32} reduction
- **Parallel Operations**
  Different regions of the key space searched simultaneously

Due to the above, the key space must be enlarged.

35

35

A Cryptanalytic Attack

- **2DES**
  - DES is not a group –proved experimentally
  - 2DES requires 2^{112} key trials to crack
  - This means 10^{12} years on a cluster
  - But with Meet in the Middle, it reduces to 1DES
- **Meet in the Middle**
  - This is a KPA attack
  - Encrypt the known PT with all 2^{56} possible keys
  - Decrypt the known CT with all 2^{56} possible keys
  - There must be at least one match
  - To avoid false positives, repeat with a second KPA.

36

36

Symmetric Crypto Today

- 3DES
  - Uses three different keys
  - Effective cryptographic strength is 56x3 = 168 bits
  - Meet in the middle reduces this to 112. Still strong.
  - Adopted by PGP, S/MIME and other network protocols
- 3DES E-D-E
  - EDE with different keys is as strong as regular 3DES
  - EDE with K1=K2 provides backward compatibility with DES
- AES
  - Block size = 128 bits. Key sizes = 128, 192, or 256 bits
  - Relies on arithmetic operations over Galois Field GF(2^8)
  - Adopted as NIST standard

37

37