Cryptography CS 555

Lecture 4: Block Ciphers. 2DES, 3DES, IDEA, Blowfish, RC5

Department of Computer Sciences

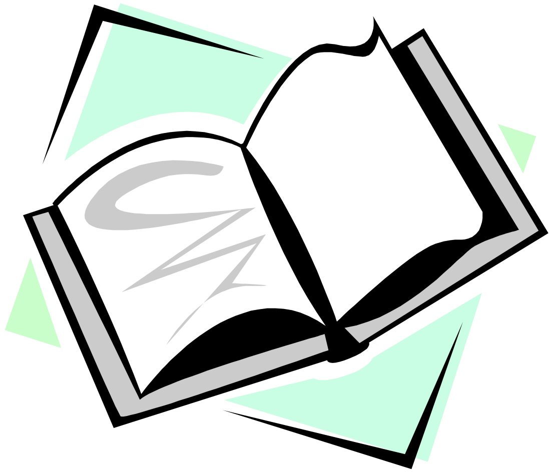
Purdue University

Lecture Outline

- Double DES
- Triple DES
- IDEA
- Blowfish
- RC5

Recommended Reading

- Chapter 3.6, 3.7 from Stinson
- Chapter 3 and 6 from Stallings (3rd edition)



Double DES

- DES uses a 56-bit key, this raised concerns about brute force attacks.
- One proposed solution: double DES.

- Apply DES twice using two keys, K1 and K2.

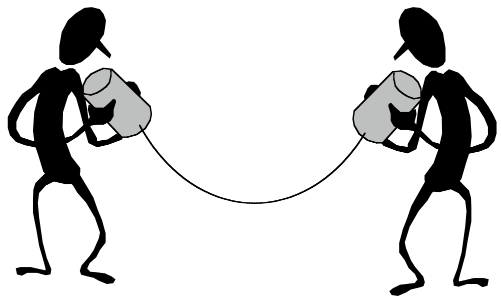
$$C = E_{K_2} [E_{K_1} [P]]$$
$$P = D_{K_1} [D_{K_2} [C]]$$

- This leads to a 2x56=112 bit key, so it is more secure than DES. **Is it?**

Meet-in-the-Middle Attack

- Goal: given the pair (P, C) find keys K_1 and K_2.
- Based on the observation:

$$C = E_{K_2}[\ E_{K_1}[\ P\ ]\ ]$$
$$E_{K_1}[\ P\ ] = D_{K_2}[\ C\ ]$$



- The attacks has higher chance of succeeding if another pair (P’, C’) is available to the cryptanalysis.

Meet-in-the-Middle Attack (cont.)

$C = E_{K_2}[\, E_{K_1}[\, P \,]\,]$
$E_{K_1}[\, P \,] = D_{K_2}[\, C \,]$

Attack, assumes the attacker knows two pairs (P,C) and (P'C'):

- Encrypt P with all 2^{56} possible keys K_1
- Store all pairs (K_1, $E_{K_1}[P]$), sorted by $E_{K_1}[P]$.
- Decrypt C using all 2^{56} possible keys K_2
- For each decrypted result, check to see if there is a match $D_{K_2}(C) = E_{K_1}(P)$.
- If yes, try another pair (P', C')
- If a match is found on the new pair, accept the keys K_1 and K_2.

Why Two Pairs (P, C)?

- DES encrypts 64-bit blocks, so for a given plaintext P, there are 2^{64} potential ciphertexts C.
- Key space: two 56-bit key, so there are 2^{112} potential double keys that can map P to C.
- Given a pair (P, C), the number of double keys (K_1, K_2) that produce $C = E_{K_2} [E_{K_1} [P]]$ is at most $2^{112}/2^{64} = 2^{48}$
- Therefore, for a pair (P, C), 2^{48} false alarms are expected.

Why Two Pairs (P, C)? (cont.)

- With one more pair (P', C'), extra 64-bit of known text, the alarm rate is $2^{48}/ 2^{64} = 1/2^{16}$
- If meet-in-the-middle is performed on two pairs (P, C) and (P', C'), the correct keys K_1 and K_2 can be determined with probability $1 - 1/2^{16.}$
- Known plaintext attack against double DES succeeds in 2^{56} as opposed to 2^{55} for DES (average).
- **The 112-bit key provides a security level similar to the 56-bit key.**

Triple DES

- With two different keys:

  Encrypt: $C = E_{K_1} [\, D_{K_2} [\, E_{K_1} [P] \,] \,]$
  Decrypt: $P = D_{K_1} [\, E_{K_2} [\, D_{K_1} [C] \,] \,]$

- Key space is 56 x 2 = 112 bits
- Some attacks: Merkle & Hellman; requires 2^{56} chosen plaintext-ciphertext pairs, effort is 2^{56}
- There is **no practical known attack** against 3DES with 2 keys

Triple DES (cont.)

- Use three different keys

  Encrypt: $C = E_{K_3}[\,D_{K_2}[\,E_{K_1}[P]\,]\,]$
  Decrypt: $P = D_{K_1}[\,E_{K_2}[\,D_{K_3}[C]\,]\,]$

- Key space is 56 x 3 = 168 bits
- No known practical attack against it.
- Many protocols/applications use 3DES (example PGP)

WHY encryption and decryption are defined like that?

International Data Encryption Algorithm (IDEA)

- Originally designed by Massey and Lai at ETH (Zurich), 1990.
- Based on mixing operations from different algebraic groups (XOR, addition mod 2^{16} , multiplication mod 2^{16} +1).
- All operations are on 16-bit sub-blocks, with no permutations used.
- Speed: faster than DES in software.

IDEA

- Design goals:
  - Block Length: deter statistical analysis
  - Key Length: deter exhaustive search
- Features:
  - **128-bit key**
  - **64 bit blocks**
  - 8 rounds,
  - operates on 16-bit numbers

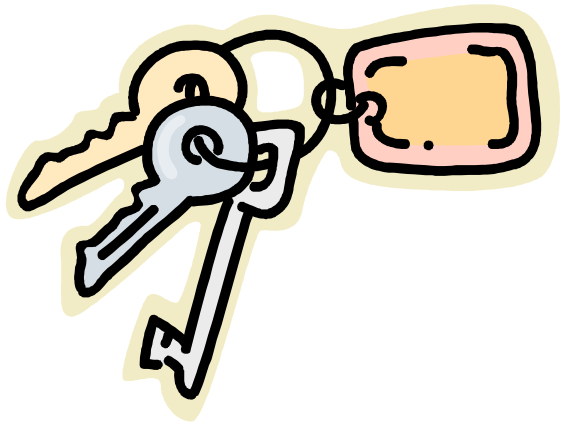
IDEA: Encryption

- 64-bit data block is divided in 4 parts:

  X_1 X_2 X_3 X_4
- In each of eight rounds with 14 steps the sub-blocks are XORd, added, multiplied with one another and with six 16-bit sub-blocks of key material, and the second and third sub-blocks are swapped.
- Finally some more key material is combined with the sub-blocks.

IDEA Key Schedule

- Total of 52 subkeys
- Subkey is generated by dividing the 128 bits key in 8 x 16 bits keys. Every time more subkeys are needed, rotate left the key 25 bits and divide again in 8 subkeys.
- The decryption keys are a little more difficult to generate.



IDEA Cryptanalysis

- **Currently there is no known practical attack against IDEA.**
- Appears secure against differential cryptanalysis.
- Key length protects against exhaustive search.
- IDEA has weak keys, avoided at key generation.

Blowfish

- A symmetric block cipher designed by Bruce Schneier in 1993/94.
- Fast implementation on 32-bit CPUs.
- Compact: runs in less than 5K of memory.
- Simple to implement and analyze its strength.
- Variable security: supports different key lengths

Blowfish Key Generation

- **Block size is 64**
- **Uses a key of variable size, from 32 to 448 bits.**
- Number of rounds is 16
- The key is used to generate:
  - 18   32-bit subkeys stored in P-arrays
  - 4   8x32 S-boxes stored in S-arrays
- Requires 521 encryptions, so it has a slow rekeying.

Blowfish Cryptanalysis

- **Key dependent S-boxes and subkeys**, generated using cipher itself, makes analysis very difficult.
- **Changing both halves** in each round increases security.
- Provided key is large enough, brute-force key search is not practical.

Blowfish Speed

From *www.counterpane.com*

Algorithm	Clock cycles per round	# rounds	# of clock cycles per byte encrypted	
Blowfish	9	16	18	free
RC5	12	16	23	RSA security
DES	18	16	45	56-bit key
IDEA	50	8	50	Ascom-Systec
Triple-DES	18	48	108	

RC5

- Proprietary cipher owned by RSA Data Security (designed by Ron Rivest).
- Very fast, operates on words.
- Variable key size, block size and number of rounds.
- Clean and simple design.
- Low memory requirement.
- Data-dependent rotations that strengthen the algorithm against cryptanalysis.

RC5 Features

- RC5 is a family of ciphers rc5-w/r/b
  - W = word size in bits (16/32/64) nb data=2w
  - R = number of rounds (0..255)
  - B = number of bytes in the key (0..255)
- Nominal version is RC5-32/12/16
  - **32-bit words so encrypts 64-bit data blocks**
  - **16 bytes (128-bit) secret key**
  - Using 12 rounds

RC5 Key Schedule

- RC5 uses 2r+2 subkey words (w-bits)
- subkeys are stored in array `S[i]`, i=0..T-1
- Initialize S to a fixed pseudorandom value
- The byte key is copied (little-endian) into a c-word array L
- A mixing operation then combines L and S to form the final S array

RC5 Encryption

```
$L_0 = A + S[0]$
$R_0 = B + S[1]$
*for* i = 1 *to* r *do*
    $L_i = ((L_{i-1} \oplus R_{i-1}) <<< R_{i-1}) + S[2 * i]$
    $R_i = ((R_{i-1} \oplus L_i) <<< L_i) + S[2 * i + 1]$
```

- Rotation is main source of non-linearity
- x <<< y cyclic rotation of word x left by y bits
- x >>> y cyclic rotation of word x right by y bits

RC5 Decryption

```
for i = 1 down to r do
    $R_{i-1} = ((R_{i-1} - S[2 * i +1]) >>> L_i) \oplus L_i$
    $L_{i-1} = ((L_i – S[2 * i]) <<< R_{i-1}) \oplus R_{i-1}$
$B = R_0 – S[1]$
$A = L_0 – S[0]$
```

- x <<< y cyclic rotation of word x left by y bits
- x >>> y cyclic rotation of word x right by y bits

RC5 Encryption Modes

- RC5 Block Cipher, is ECB mode.
- RC5-CBC, is CBC mode.
- RC5-CBC-PAD, is CBC with padding by bytes with value being the number of padding bytes.
- RC5-CTS, a variant of CBC which is the same size as the original message, uses ciphertext stealing to keep size same as original, handles plaintext of any size and produces ciphertext of equal size.

Advanced Encryption Standard

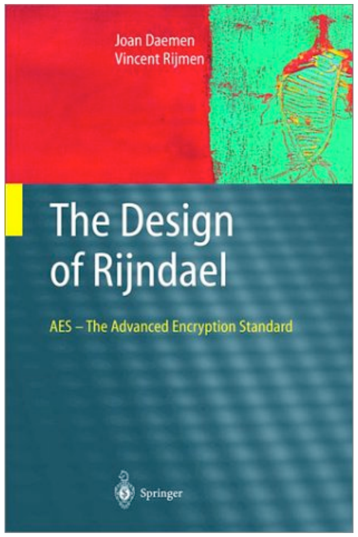
- In 1997, NIST made a formal call for algorithms stipulating that the AES would specify an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide.
- Goal: replace DES for both government and private-sector encryption.
- The algorithm must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.
- In 1998, NIST announced a group of 15 AES candidate algorithms.

AES Selection Process

- In 1999, out of 15, the selction was narrowed to 5 candidates: MARS, RC6, Rijndael, Serpent, and Twofish.
- All the five protocols were thought to be secure
- Criteria for selecting AES: security, robustness, speed
- On October 2, 2000, NIST announced that it has selected **Rijndael** (invented by Joan Daemen and Vincent Rijmen) to propose for the AES.
- February 2001, FIPS 197 (AES) was published for public review and comments.

Rijndael Features

- Designed to be efficient in both hardware and software across a variety of platforms.
- Uses a variable block size, **128,192, 256-bits**, key size **of 128-, 192-, or 256-bits.**
- 128-bit round key used for each round (Can be pre-computed and cached for future encryptions).
- Note: AES uses a 128-bit block size.
- Variable number of rounds (10, 12, 14):
  - 10 if B = K = 128 bits
  - 12 if either B or K is 192 and the other is 192
  - 14 if either B or K is 256 bits



Rijndael Design

- Rijndael’s strength is in design simplicity, rich algebraic structure, and efficiency.
- Operations performed on State (4 rows of bytes).
- The 128 bit key is expanded as an array of 44 32bits words; 4 distinct words serve as a round key for each round.
- Algorithms composed of three layers
  - Linear diffusion
  - Non-linear diffusion
  - Key mixing

Rijandael: High-Level Description

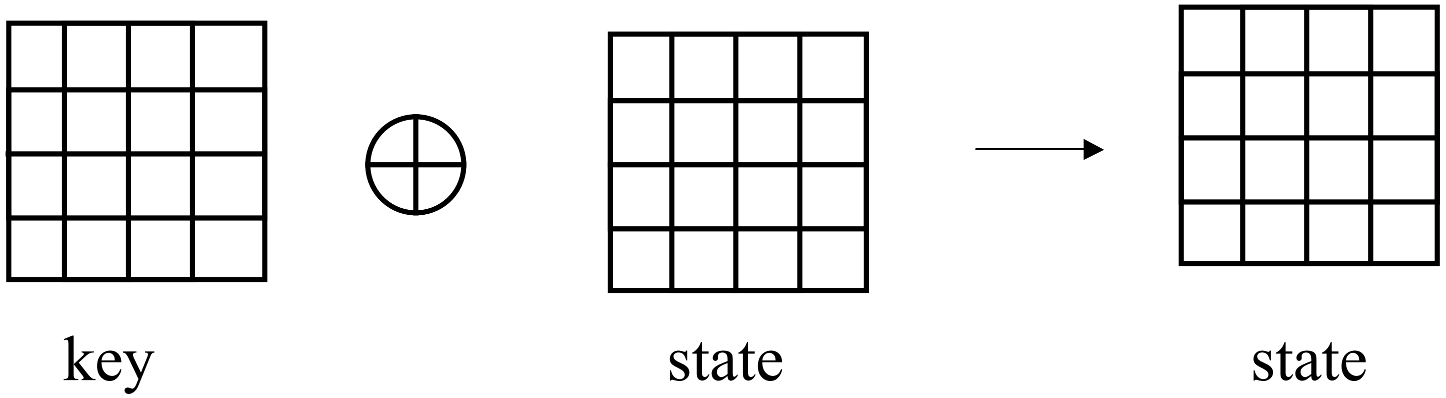
```
State = X
AddRoundKey(State, $Key_0$)
for r = 1 to Nr - 1
    SubBytes(State, S-box)
    ShiftRows(State)
    MixColumns(State)
    AddRoundKey(State, $Key_r$)
endfor
SubBytes(State, S-box)
ShiftRows(State)
AddRoundKey(State, $Key_{Nr}$)
Y = State
```

AddRound Key

State is represented as follows (16 bytes):

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

AddRoundKey(State, Key):



SubBytes

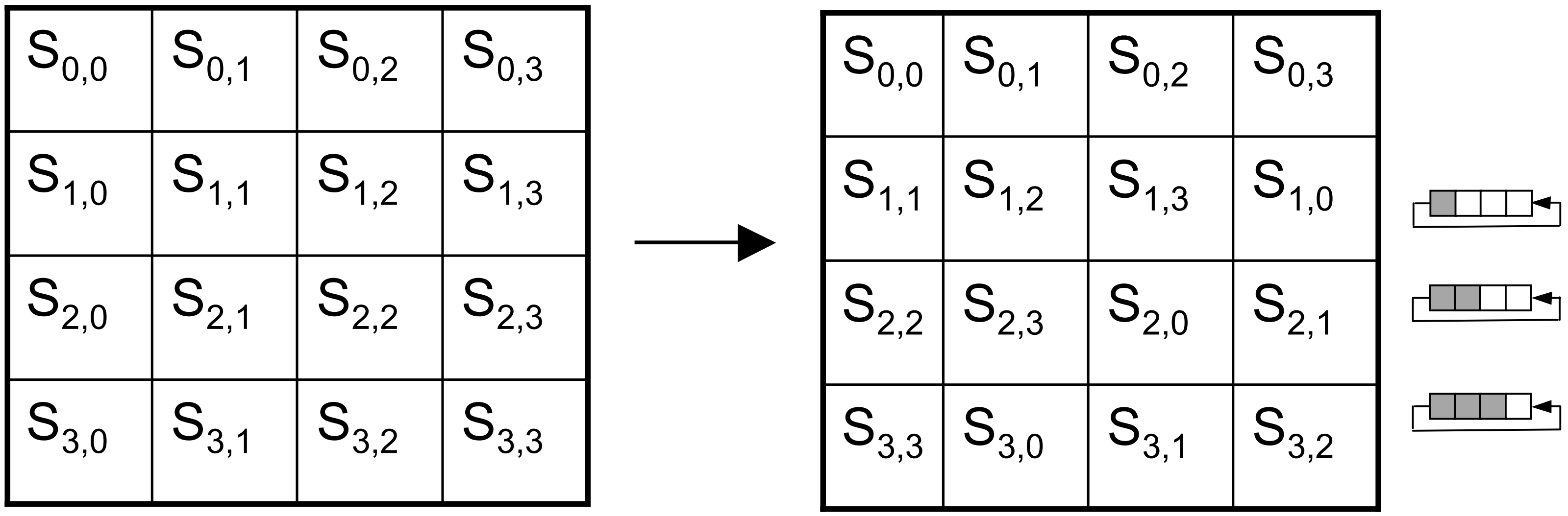
- Byte substitution using non-linear S-Box (independently on each byte).
- S-box is represented as a 16x16 array, rows and columns indexed by hexadecimal bits
- 8 bytes replaced as follows: 8 bytes defines a hexadecimal number rc, then $s_{r,c}$ = binary(S-box(r, c))
- How is AES S-box different from DES S-box?
  - Only one S-box
  - S-boxes based on modular arithmetic with polynomials, can be defined algebraically, not random
  - Easy to analyze, prove attacks fail

S-box Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	3	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

Example: hexa 53 is replaced with hexa ED

ShiftRows



$$\begin{array}{|c|c|c|c|}\hline S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ \hline S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ \hline S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ \hline S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|c|}\hline S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ \hline S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ \hline S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ \hline S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \\ \hline \end{array}$$

MixColumns

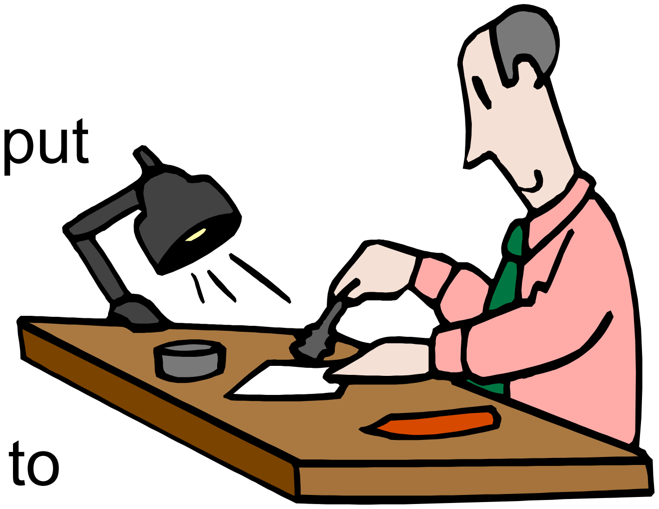
- Interpret each column as a vector of length 4.
- Each column of State is replaced by another column obtained by multiplying that column with a matrix in a particular field.

Decryption

- The decryption algorithm is not identical with the encryption algorithm, but uses the same key schedule.
- There is also a way of implementing the decryption with an algorithm that is equivalent to the encryption algorithm (each operation replaced with its inverse), however in this case, the key schedule must be changed.

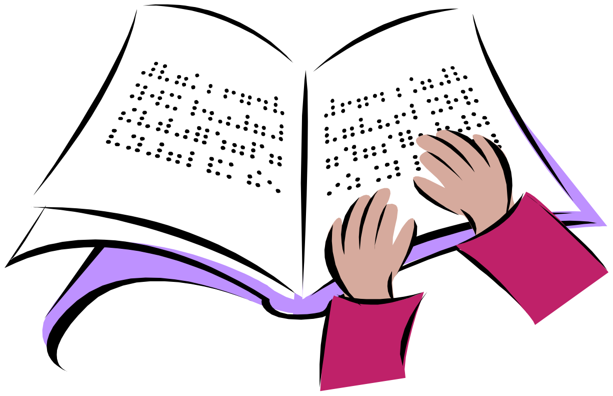
Rijandel Cryptanalysis

- Resistant to linear and differential cryptanalysis
- Differential trail
  - Probability that a given difference a’ pattern at input produces an output difference of b’
  - Choose S-box and multiplication polynomial to minimize maximum difference probability



Rijndael Cryptanalysis

- Academic break on weaker version of the cipher, 9 rounds
- Requires 2^{224} work and 2^{85} chosen *related-key* plaintexts.
- Attack not practical.



AES Encryption Modes

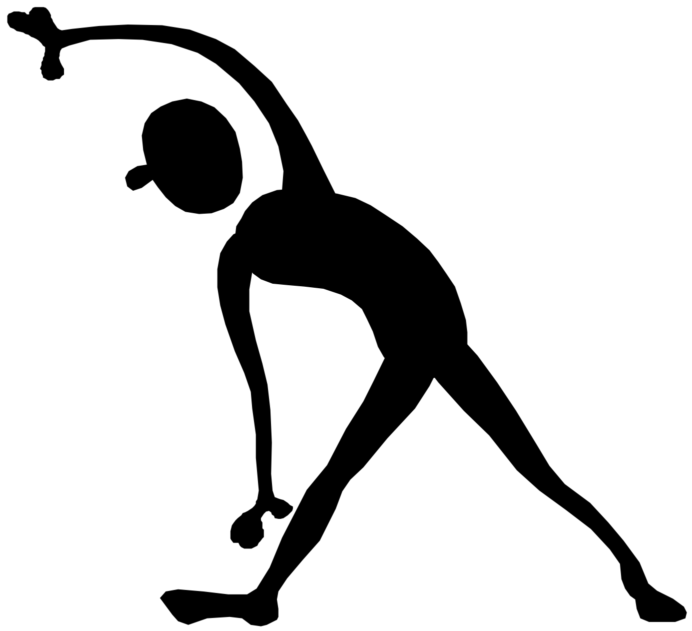
- Sp 800-38A, Recommendation for Block Cipher Modes of Operation
- ECB
- CBC
- CFB
- OFB
- CTR

Characteristics of Modern Block Ciphers

- Variable key length
- Mixed operators: use more than one arithmetic and/or Boolean; this can provide non-linearity
- Data dependent rotation
- Key-dependent S-boxes
- Lengthy key schedule algorithm
- Variable plaintext/ciphertext block length
- Variable number of rounds
- Operation on both data halves each round
- Variable F function (varies from round to round
- Key-dependent rotation

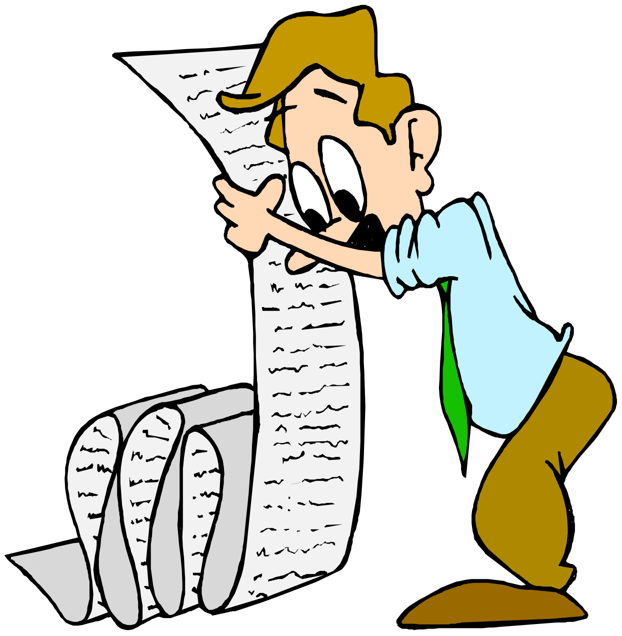
Exercise

- Homework 2 handed
- Due in class Feb. 8
- Everybody get a terminal and run openssl speed des
- Openssl speed aes
- Openssl speed idea
- What do you notice?



Summary

- Attacks against DES: brute force, differential cryptanalysis (not really effective), and linear cryptanalysis (not practical).
- The main critique to DES is the key length
- Other block cipher protocols: Blowfish (fast), IDEA, RC5 and AES (current standard). Not known practical attacks against them.



Coming Attractions…

- Stream ciphers
- Recommended reading:
- Stinson: 3.6
- Stallings: 5.1 and 5.2, 6.5 (for RC4)