Chapter 3
Symmetric Key Crypto
Sep 1st

Security of DES

- Security of DES depends a lot on S-boxes
  - Everything else in DES is linear
- Thirty years of intense analysis has revealed no "back door"
- Attacks today use exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were ahead of their time

Block Cipher Notation

- P = plaintext block
- C = ciphertext block
- Encrypt P with key K to get ciphertext C
  - $C = E(P, K)$
- Decrypt C with key K to get plaintext P
  - $P = D(C, K)$
- Note that
  - $P = D(E(P, K), K)$ and $C = E(D(C, K), K)$

Triple DES

- Today, 56 bit DES key is too small
- But DES is everywhere: What to do?
- **Triple DES** or **3DES** (112 bit key)
  - $C = E(D(E(P,K_1),K_2),K_1)$
  - $P = D(E(D(C,K_1),K_2),K_1)$
- Why use Encrypt-Decrypt-Encrypt (EDE) with 2 keys?
  - Backward compatible: $E(D(E(P,K),K),K) = E(P,K)$
  - And 112 bits is enough

3DES

- Why not $C = E(E(P,K),K)$?
  - Still just 56 bit key
- Why not $C = E(E(P,K_1),K_2)$?
- A (semi-practical) known plaintext attack
  - Precompute table of $E(P,K_1)$ for every possible key K_1 (resulting table has 2^{56} entries)
  - Then for each possible K_2 compute $D(C,K_2)$ until a match in table is found
  - When match is found, have $E(P,K_1) = D(C,K_2)$
  - Result is keys: $C = E(E(P,K_1),K_2)$

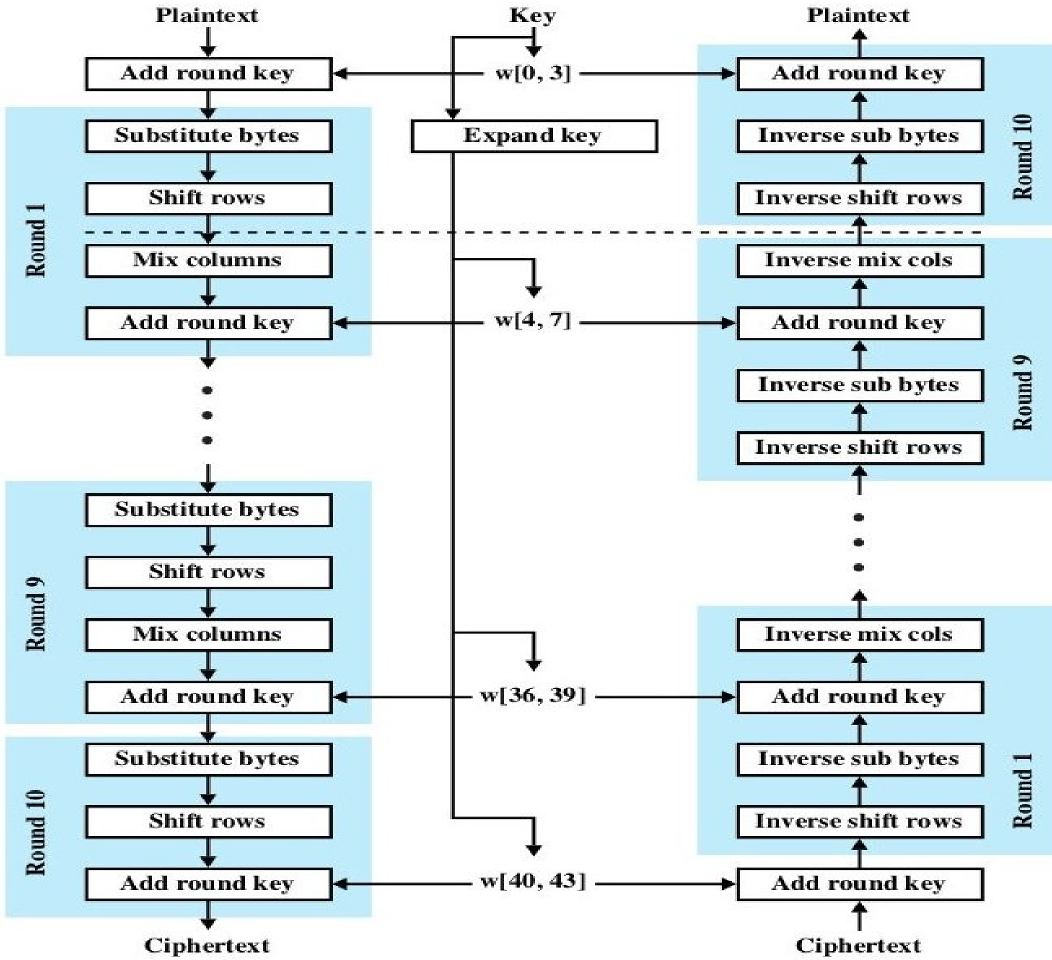
Advanced Encryption Standard

- Replacement for DES
- AES competition (late 90's)
  - NSA openly involved
  - Transparent process
  - Many strong algorithms proposed
  - Rijndael Algorithm ultimately selected
    - Pronounced like "Rain Doll" or "Rhine Doll"
- Iterated block cipher (like DES)
- Not a Feistel cipher (unlike DES)

AES Overview

- **Block size:** 128, 192 or 256 bits
- **Key length:** 128, 192 or 256 bits (independent of block size)
- 10 to 14 rounds (depends on key length)
- Each round uses 4 functions (in 3 "layers")
  - ByteSub (nonlinear layer)
  - ShiftRow (linear mixing layer)
  - MixColumn (nonlinear layer)
  - AddRoundKey (key addition layer)

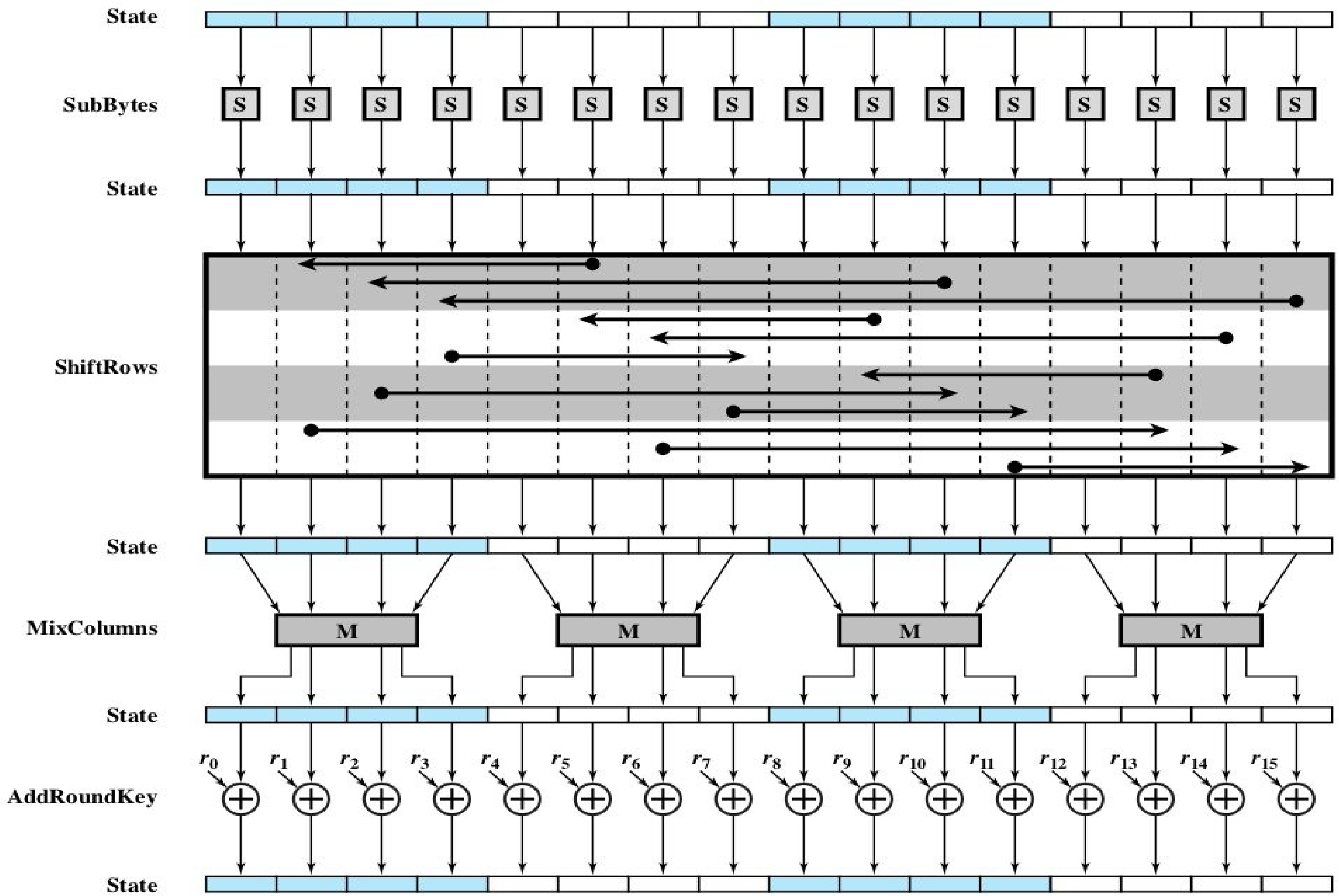
AES Overview



(a) Encryption (b) Decryption

From Computer Security, Priciples and Practice by William Stallings

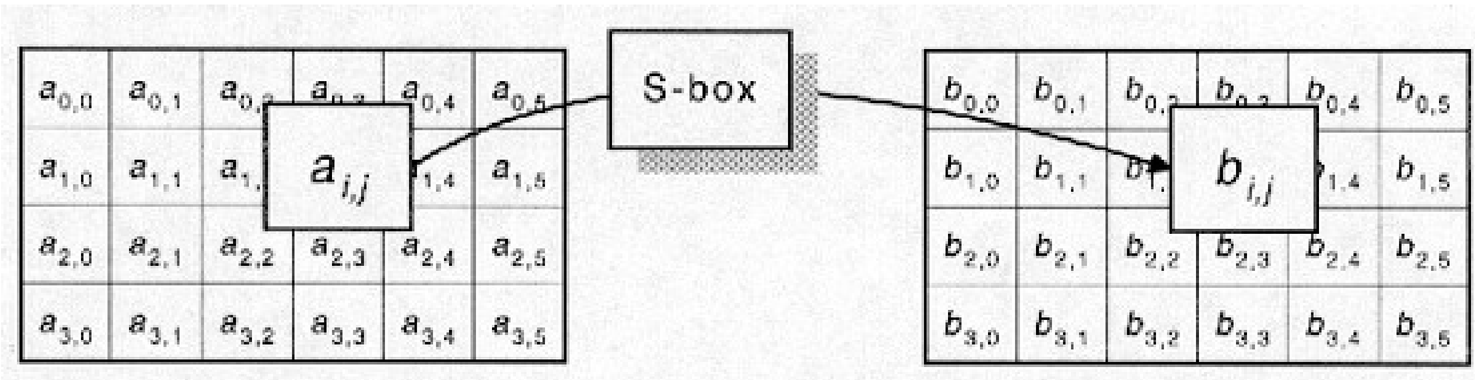
AES Encryption Round



From Computer Security, Priciples and Practice by William Stallings

AES ByteSub

- Assume 192 bit block, 4x6 bytes



- ByteSub is AES's "S-box"
- Can be viewed as nonlinear (but invertible) composition of two math operations
- Resilient to known attacks (low correlation between input and output bits)

AES "S-box"

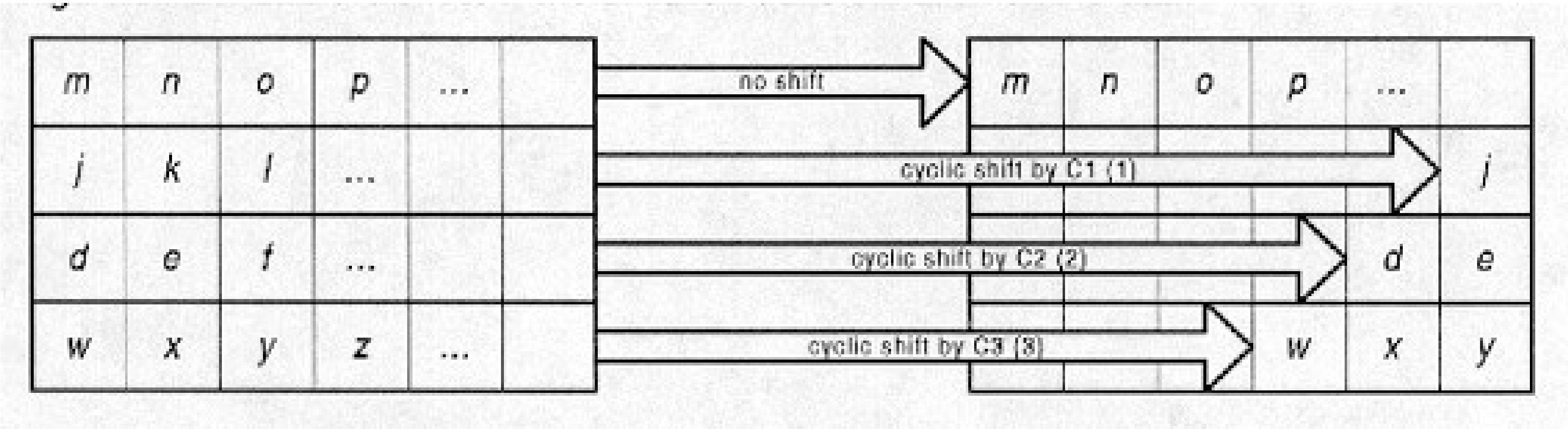
Last 4 bits of input

First 4 bits of input

	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	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

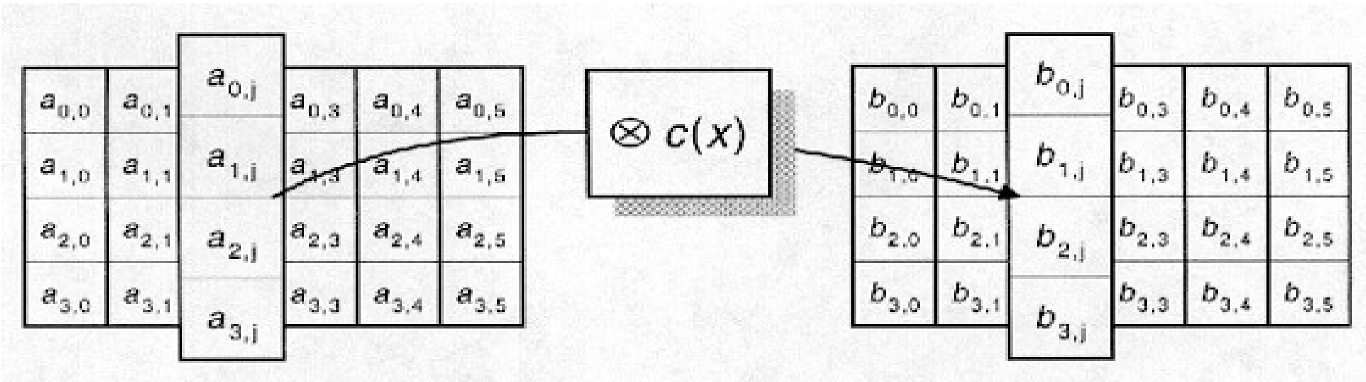
AES ShiftRow

- Cyclic shift rows



AES MixColumn

- Nonlinear, invertible operation applied to each column (just as S-box it is based on finite fields again)



- Implemented as a (big) lookup table
- Each byte of a column is function off all bytes in the column
- Together with shift row => after few rounds all output bits depend on all input bits

AES AddRoundKey

- XOR subkey with block

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} & k_{04} & k_{05} \\ k_{10} & k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\ k_{20} & k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \\ k_{30} & k_{31} & k_{32} & k_{33} & k_{34} & k_{35} \end{bmatrix} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} & b_{04} & b_{05} \\ b_{10} & b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{20} & b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{30} & b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \end{bmatrix}$$

Block Subkey

- RoundKey (subkey) determined by **key schedule** algorithm

AES Decryption

- To decrypt, process must be invertible
- Inverse of MixAddRoundKey is easy, since $\oplus$ is its own inverse
- MixColumn is invertible (inverse is also implemented as a lookup table)
- Inverse of ShiftRow is easy (cyclic shift the other direction)
- ByteSub is invertible (inverse is also implemented as a lookup table)

A Few Other Block Ciphers

- Briefly…
  - IDEA
  - Blowfish
  - RC6
- More detailed…
  - TEA

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 to produce the necessary nonlinearity
  - 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

$$R_i = L_{i-1} \oplus K_i$$

$$L_i = R_{i-1} \oplus F(L_{i-1} \oplus K_i)$$

- 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

RC6

- Invented by Ron Rivest (RC4, RSA, MD5)
- Variables
  - Block size
  - Key size
  - Number of rounds
- An AES finalist
- Uses **data dependent rotations**
  - Unusual to rely on data as part of algorithm

Tiny Encryption Algorithm

- 64 bit block, 128 bit key
- Assumes 32-bit arithmetic
- Number of rounds is variable (32 is considered secure)
- Uses "weak" round function, so large number rounds required

TEA Encryption

Assuming 32 rounds:

(K[0],K[1],K[2],K[3]) = 128 bit key

(L,R) = plaintext (64-bit block)

delta = 0x9e3779b9

sum = 0

for i = 1 to 32

sum += delta

L += ((R<<4)+K[0]) ⊕(R+sum)⊕((R>>5)+K[1])

R += ((L<<4)+K[2]) ⊕(L+sum)⊕((L>>5)+K[3])

next i

ciphertext = (L,R)

TEA Decryption

Assuming 32 rounds:

(K[0],K[1],K[2],K[3]) = 128 bit key

(L,R) = ciphertext (64-bit block)

delta = 0x9e3779b9

sum = delta << 5

for i = 1 to 32

$R \mathrel{-}= ((L\ll 4)+K[2]) \oplus (L+sum) \oplus ((L\gg 5)+K[3])$

$L \mathrel{-}= ((R\ll 4)+K[0]) \oplus (R+sum) \oplus ((R\gg 5)+K[1])$

sum −= delta

next i

plaintext = (L,R)

TEA comments

- **Almost** a Feistel cipher
  - Uses + and - instead of $\oplus$ (XOR)
- Simple, easy to implement, fast, low memory requirement, etc.
- Possibly a related key attack
- eXtended TEA (XTEA) eliminates related key attack (slightly more complex)
- Simplified TEA (STEA) — insecure version used as an example for cryptanalysis

Block Cipher Modes

Multiple Blocks

- How to encrypt multiple blocks?
- A new key for each block?
  - As bad as (or worse than) a one-time pad!
- Encrypt each block independently?
- Make encryption depend on previous block(s), i.e., "chain" the blocks together?
- How to handle partial blocks?

Modes of Operation

- Many modes of operation — we discuss three
- Electronic Codebook (**ECB**) mode
  - Obvious thing to do
  - Encrypt each block independently
  - There is a serious weakness
- Cipher Block Chaining (**CBC**) mode
  - Chain the blocks together
  - More secure than ECB, virtually no extra work
- Counter Mode (**CTR**) mode
  - Acts like a stream cipher
  - Popular for random access

ECB Mode

- Notation: $C=E(P,K)$
- Given plaintext $P_0,P_1,\ldots,P_m,\ldots$
- Obvious way to use a block cipher is

Encrypt	**Decrypt**
$C_0 = E(P_0, K)$,	$P_0 = D(C_0, K)$,
$C_1 = E(P_1, K)$,	$P_1 = D(C_1, K)$,
$C_2 = E(P_2, K),\ldots$	$P_2 = D(C_2, K),\ldots$

- For a fixed key K, this is an electronic version of a codebook cipher
- A new codebook for each key

ECB Cut and Paste Attack

- Suppose plaintext is

  `Alice digs Bob. Trudy digs Tom.`

- Assuming 64-bit blocks and 8-bit ASCII:

  P_0 = "`Alice di`", P_1 = "`gs Bob. `",

  P_2 = "`Trudy di`", P_3 = "`gs Tom. `"

- Ciphertext: C_0,C_1,C_2,C_3
- Trudy cuts and pastes: C_0,C_3,C_2,C_1
- Decrypts as

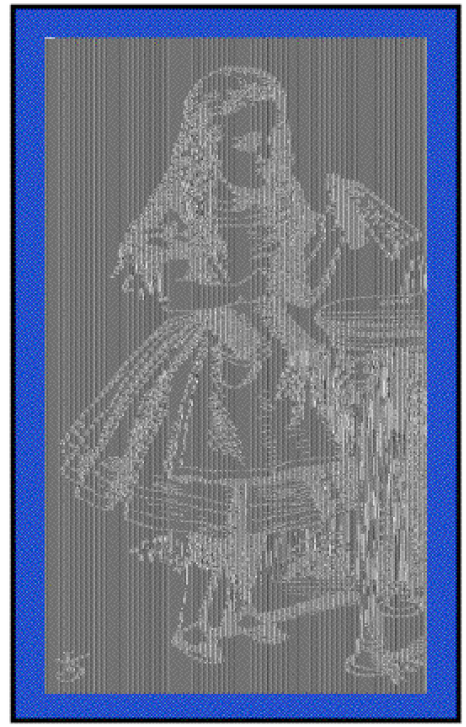
  `Alice digs Tom. Trudy digs Bob.`

ECB Weakness

- Suppose $P_i = P_j$
- Then $C_i = C_j$ and Trudy knows $P_i = P_j$
- This gives Trudy some information, even if she does not know P_i or P_j
- Trudy might know P_i
- Is this a serious issue?

Alice Hates ECB Mode

- Alice's uncompressed image, Alice ECB encrypted (TEA)

- Why does this happen?
- Same plaintext block $\Rightarrow$ same ciphertext!

CBC Mode

- Blocks are "chained" together
- A random initialization vector, or IV, is required to initialize CBC mode
- IV is random, but need not be secret

Encryption

$C_0 = E(IV \oplus P_0, K),$

$C_1 = E(C_0 \oplus P_1, K),$

$C_2 = E(C_1 \oplus P_2, K),\ldots$

Decryption

$P_0 = IV \oplus D(C_0, K),$

$P_1 = C_0 \oplus D(C_1, K),$

$P_2 = C_1 \oplus D(C_2, K),\ldots$

CBC Mode

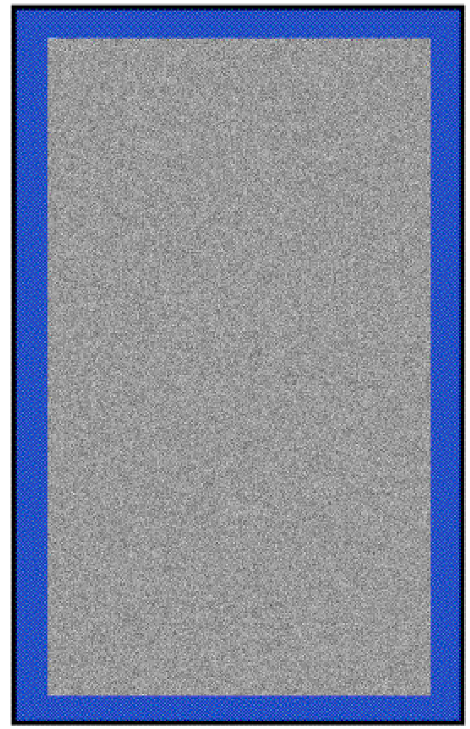
- Identical plaintext blocks yield different ciphertext blocks
- Cut and paste is still possible, but more complex (and will cause garbles)
- If C_1 is garbled to, say, G then

$$P_1 \neq C_0 \oplus D(G, K),\ P_2 \neq G \oplus D(C_2, K)$$

- But $P_3 = C_2 \oplus D(C_3, K),\ P_4 = C_3 \oplus D(C_4, K), \ldots$
- Automatically recovers from errors!

Alice Likes CBC Mode

- Alice's uncompressed image, Alice CBC encrypted (TEA)



- Why does this happen?
- Same plaintext yields different ciphertext!

Counter Mode (CTR)

- CTR is popular for random access
- Use block cipher like stream cipher

Encryption **Decryption**

$C_0 = P_0 \oplus E(IV, K),$ $P_0 = C_0 \oplus E(IV, K),$

$C_1 = P_1 \oplus E(IV+1, K),$ $P_1 = C_1 \oplus E(IV+1, K),$

$C_2 = P_2 \oplus E(IV+2, K),\ldots$ $P_2 = C_2 \oplus E(IV+2, K),\ldots$

Next …
Integrity
(& Public Key Cryptography)