Symmetric Cryptography

CS 458: Information Security

Kevin Jin

Outline

- Commercial Symmetric systems
  - DES
  - AES
- Modes of block and stream ciphers

Some materials borrowed from Mark Stamp at San Jose State University

Reading

- Chapters 2 and 20 from text.
- *AES Standard issued as FIPS PUB 197*
  - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- *Handbook of Applied Cryptography,* Menezes, van Oorschot, Vanstone
  - Chapter 7
  - http://www.cacr.math.uwaterloo.ca/hac/

Administrivia

- Mr. Xiaoliang Wu, TA office Hour
  Wed 3:15 to 4:15 pm or by appointment

Stream, Block Ciphers

- E encipherment function
  - $E_k(b)$ encipherment of message b with key k
  - In what follows, $m = b_1 b_2$ ..., each b_i of fixed length
- Block cipher
  - $E_k(m) = E_k(b_1)E_k(b_2)$ ...
- Stream cipher
  - $k = k_1 k_2$ ...
  - $E_k(m) = E_{k1}(b_1)E_{k2}(b_2)$ ...
  - If $k_1 k_2$ ... repeats itself, cipher is *periodic* and the length of its period is one cycle of $k_1 k_2$ ...

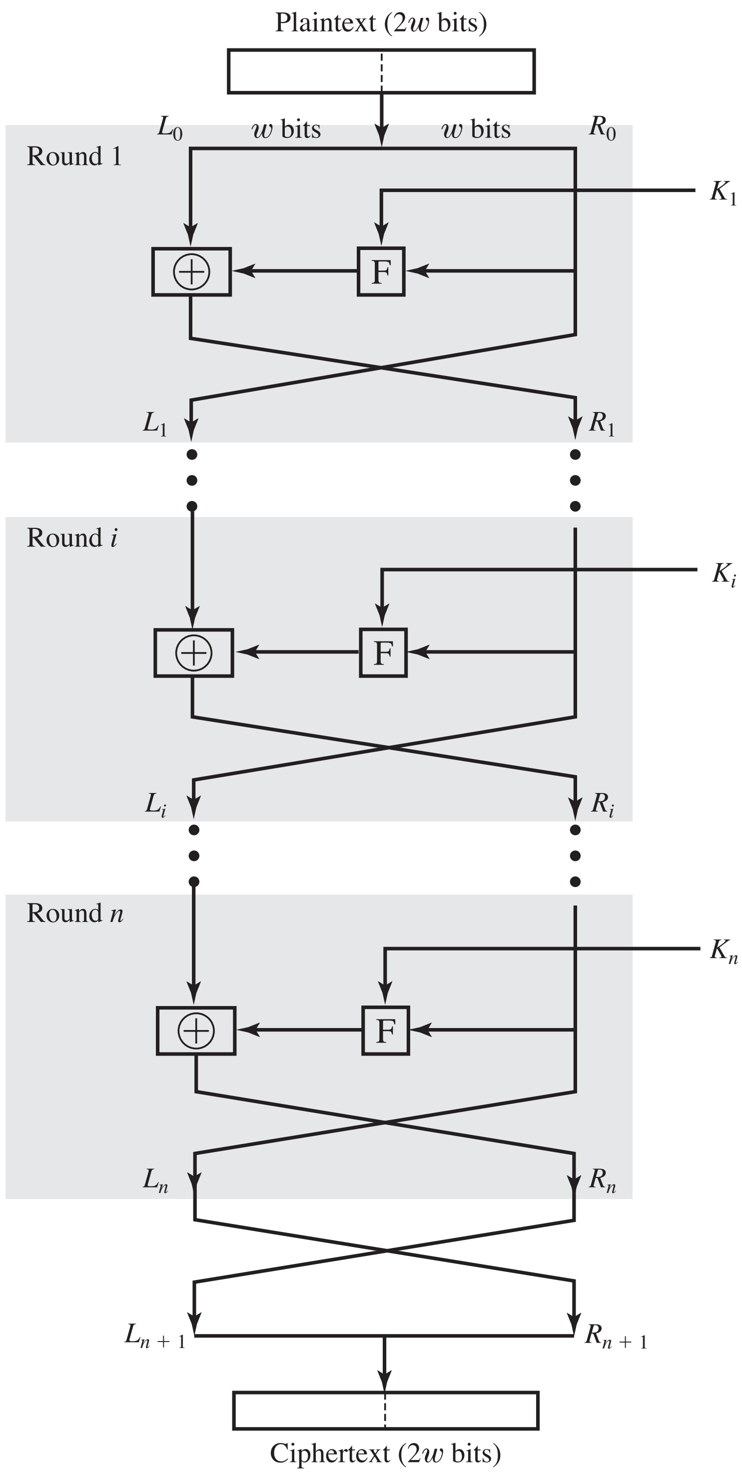
Examples

- Vigenère cipher
  - $|b_i|$ = 1 character, $k = k_1k_2 \ldots$ where $|k_i|$ = 1 character
  - Each b_i enciphered using $k_{i \bmod \mathrm{length}(k)}$
  - Stream cipher
- DES (Data Encryption Standard)
  - $|b_i|$ = 64 bits, $|k|$ = 56 bits
  - Each b_i enciphered separately using k
  - Block cipher

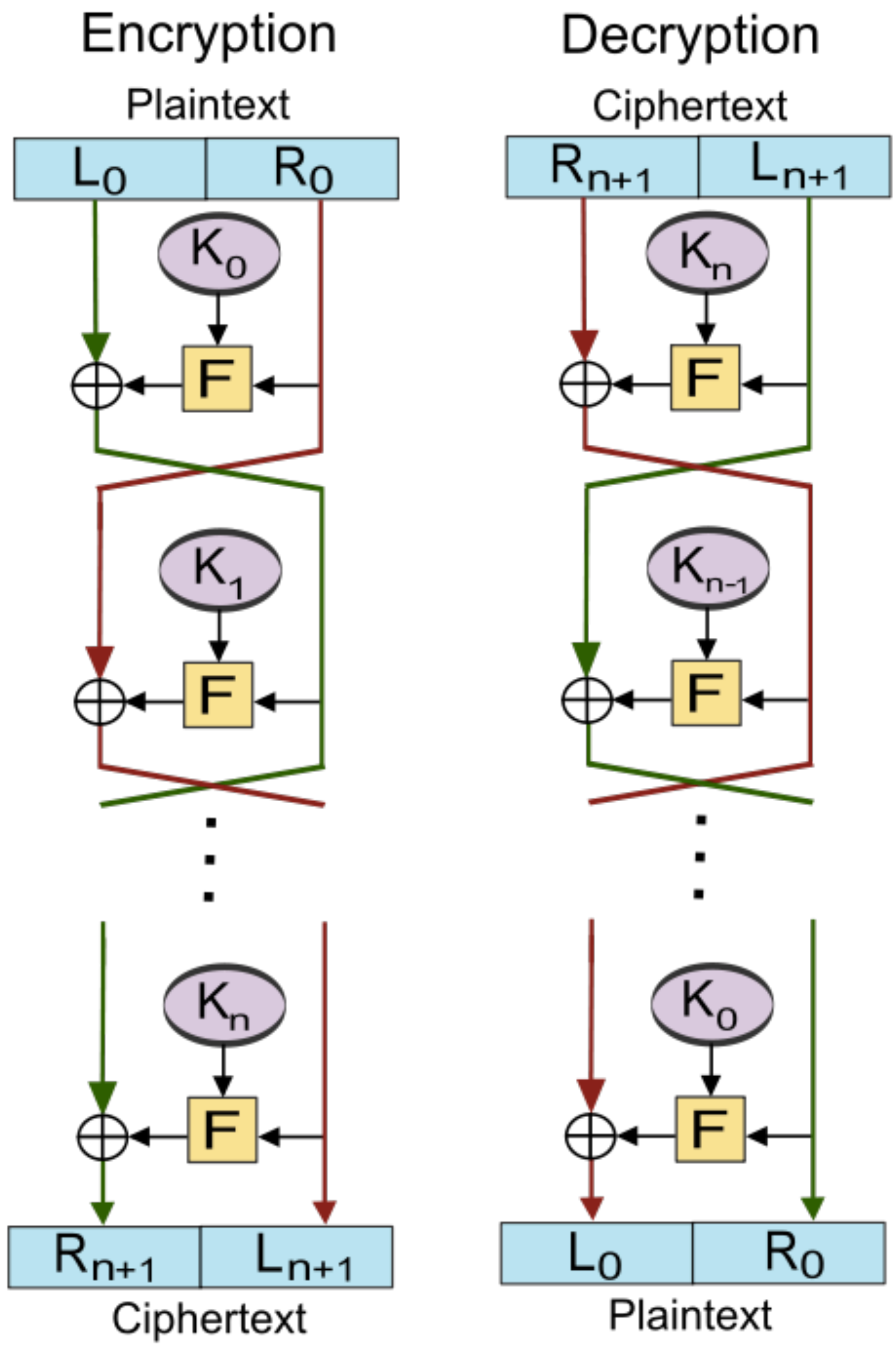
Avalanche Effect

- A key desirable property of an encryption algorithm
  - a change of **one** input or key bit results in changing approximately **half of the** output bits
- Why is this a good property?
- If the change were small, this might provide a way to reduce the size of the key space to be searched
- DES exhibits strong avalanche effect

Feistel Cipher



$$L(i) = R(i-1)$$
$$R(i) = L(i-1) \text{ xor } f(K(i), R(i-1))$$



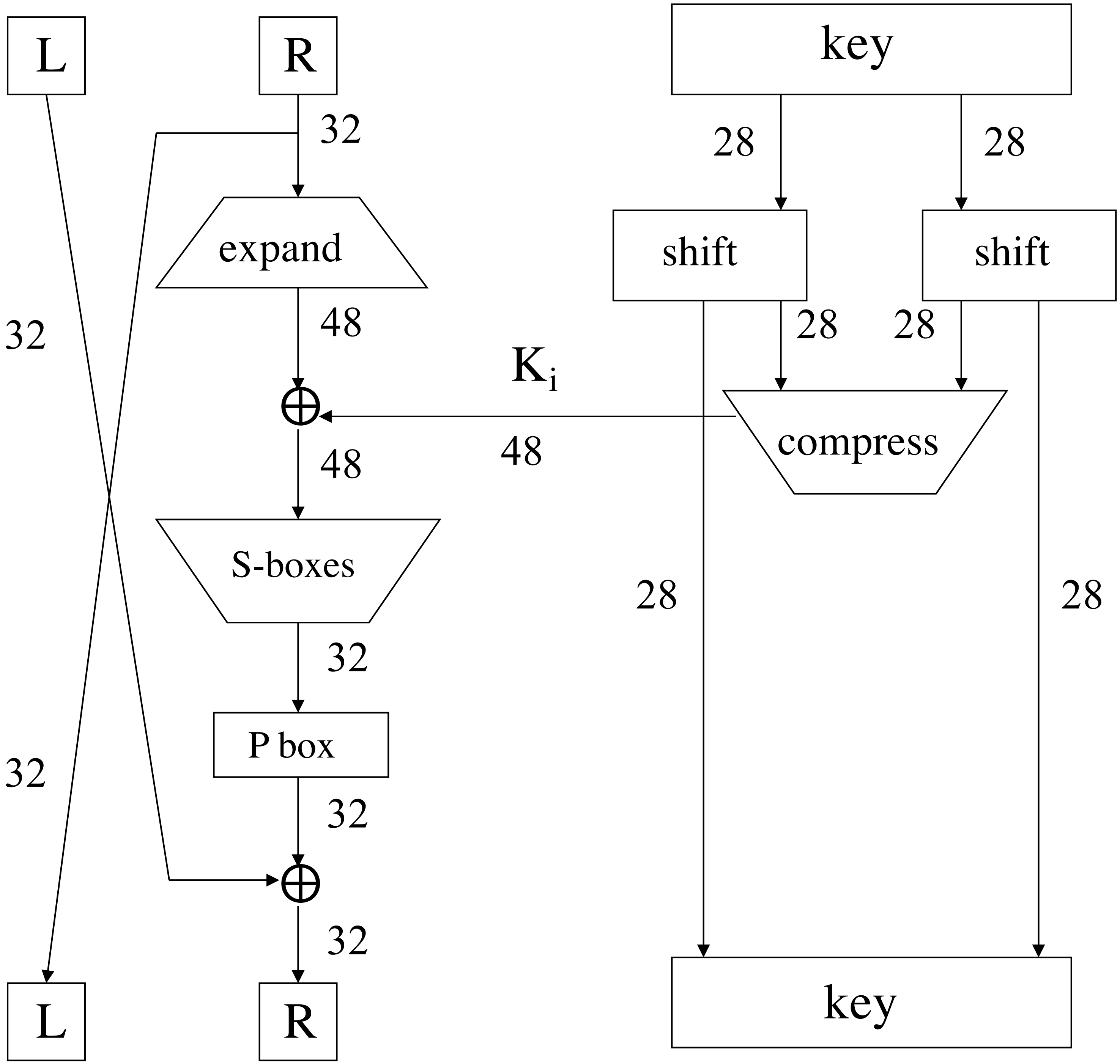
Why is this nice?

Data Encryption Standard

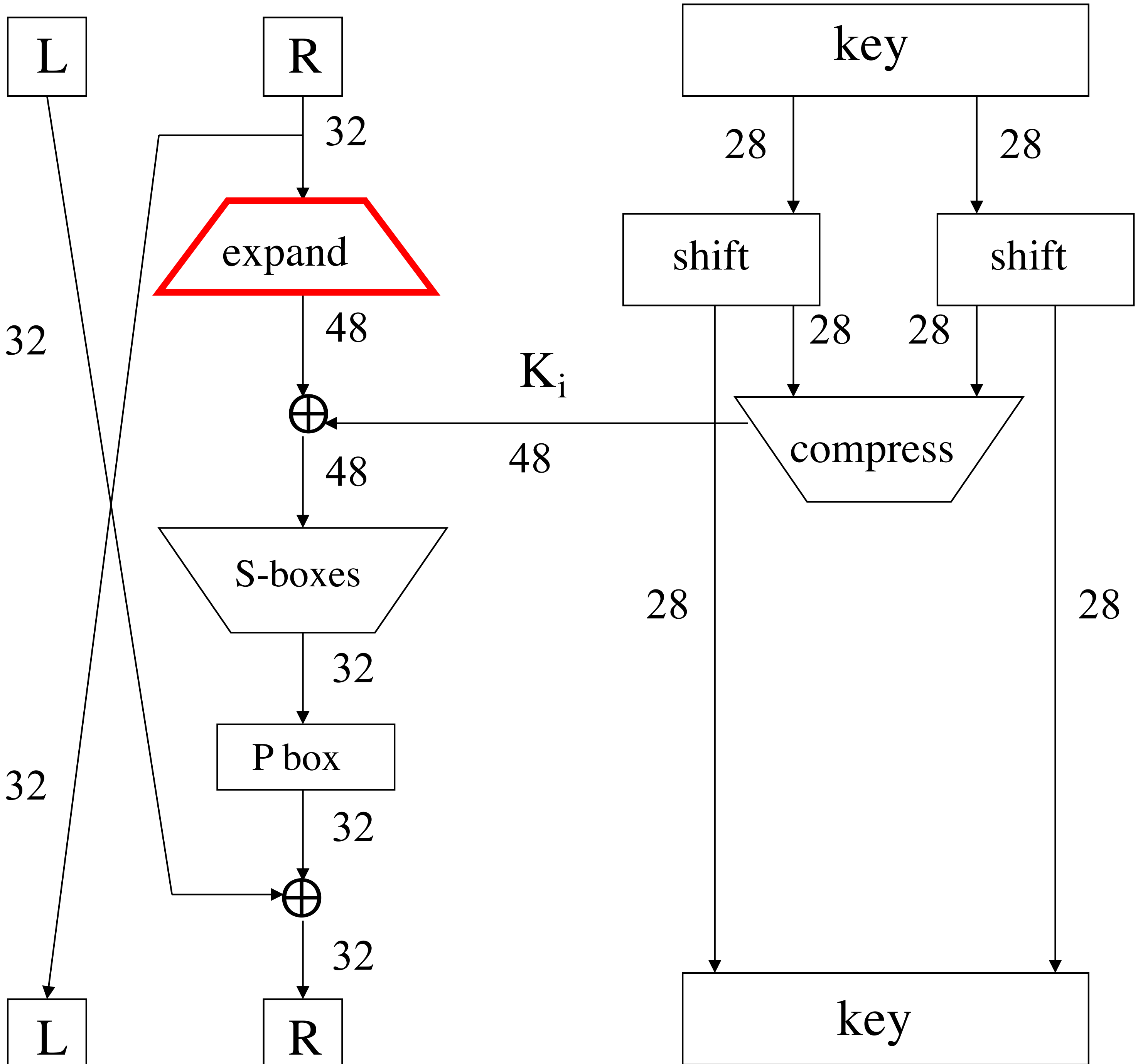
- **DES** developed in 1970's
- Based on IBM's Lucifer cipher
- DES was U.S. government standard
- DES development was controversial
  - NSA secretly involved
  - Design process was secret
  - Key length reduced from 128 to 56 bits
  - Subtle changes to Lucifer algorithm

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



One Round of DES

DES Expansion Permutation

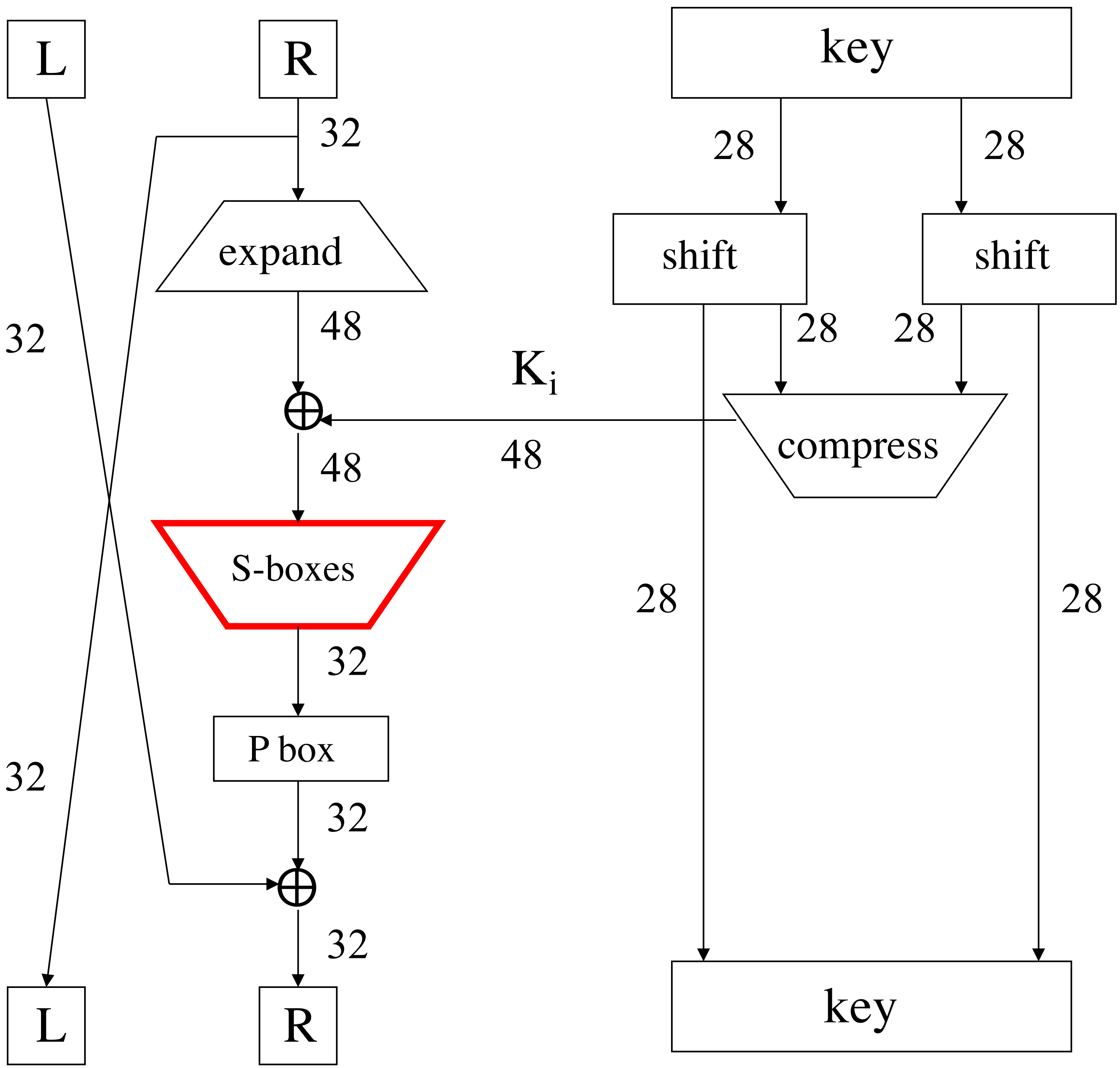
- Input 32 bits

```
 0  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
```

- Output 48 bits

```
31  0  1  2  3  4  3  4  5  6  7  8
 7  8  9 10 11 12 11 12 13 14 15 16
15 16 17 18 19 20 19 20 21 22 23 24
23 24 25 26 27 28 27 28 29 30 31  0
```

One Round of DES



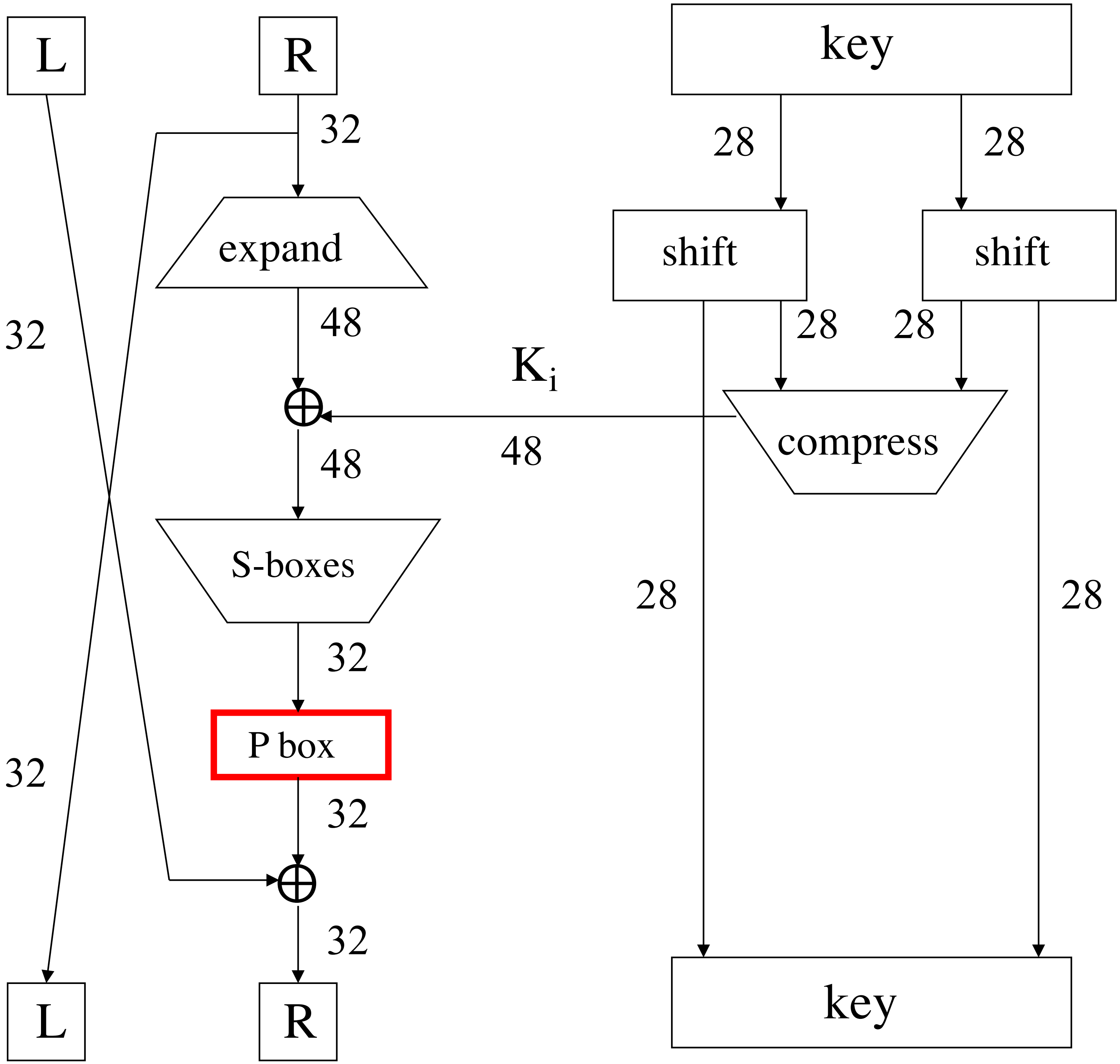
DES S-box

- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- S-box number 1

input bits (0,5)

↓ input bits (1,2,3,4)

```
   | 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
------------------------------------------------------------------------------------
00 | 1110 0100 1101 0001 0010 1111 1011 1000 0011 1010 0110 1100 0101 1001 0000 0111
01 | 0000 1111 0111 0100 1110 0010 1101 0001 1010 0110 1100 1011 1001 0101 0011 1000
10 | 0100 0001 1110 1000 1101 0110 0010 1011 1111 1100 1001 0111 0011 1010 0101 0000
11 | 1111 1100 1000 0010 0100 1001 0001 0111 0101 1011 0011 1110 1010 0000 0110 1101
```



One Round of DES

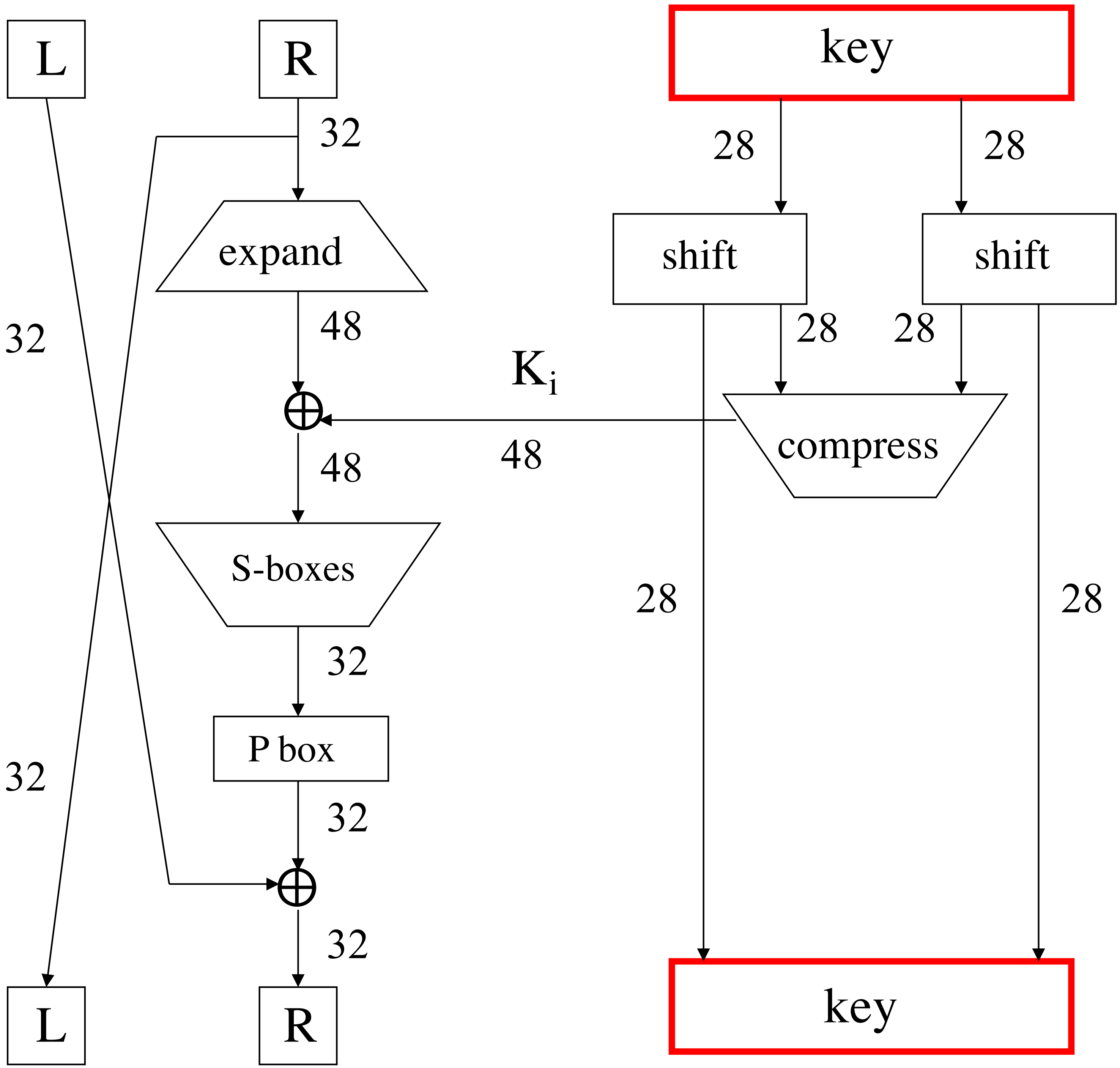
DES P-box

- Input 32 bits

0	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

- Output 32 bits

15	6	19	20	28	11	27	16	0	14	22	25	4	17	30	9
1	7	23	13	31	26	2	8	18	12	29	5	21	10	3	24



One Round of DES

DES Subkey

- 56 bit DES key, numbered 0,1,2,…,55
- Left half key bits, `LK`

```
49 42 35 28 21 14  7
 0 50 43 36 29 22 15
 8  1 51 44 37 30 23
16  9  2 52 45 38 31
```

- Right half key bits, `RK`

```
55 48 41 34 27 20 13
 6 54 47 40 33 26 19
12  5 53 46 39 32 25
18 11  4 24 17 10  3
```

DES Subkey

- For rounds `i=1,2,...,16`
  - Let LK = (LK circular shift left by r_i)
  - Let RK = (RK circular shift left by r_i)
  - Left half of subkey K_i is of LK bits

```
13 16 10 23  0  4  2 27 14  5 20  9
22 18 11  3 25  7 15  6 26 19 12  1
```

  - Right half of subkey K_i is RK bits

```
12 23  2  8 18 26  1 11 22 16  4 19
15 20 10 27  5 24 17 13 21  7  0  3
```

DES Subkey

- For rounds $1, 2, 9$ and 16 the shift r_i is 1, and in all other rounds r_i is 2
- Bits $8,17,21,24$ of LK omitted each round
- Bits $6,9,14,25$ of RK omitted each round
- **Compression permutation** yields 48 bit subkey K_i from 56 bits of LK and RK
- **Key schedule** generates subkey

DES Last Word (Almost)

- An initial permutation before round 1
- Halves are swapped after last round
- A final permutation (inverse of initial perm) applied to (R_{16}, L_{16})
- None of this serves security purpose

Security of DES

- Security depends heavily on S-boxes
  - Everything else in DES is linear
  - Have eight S-boxes which map 6 to 4 bits
  - Row selection depends on both data & key
    - feature known as autoclaving (auto keying)
- 30+ years of intense analysis has revealed no "back door"
- Attacks, essentially exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were way ahead of their time

Controversy

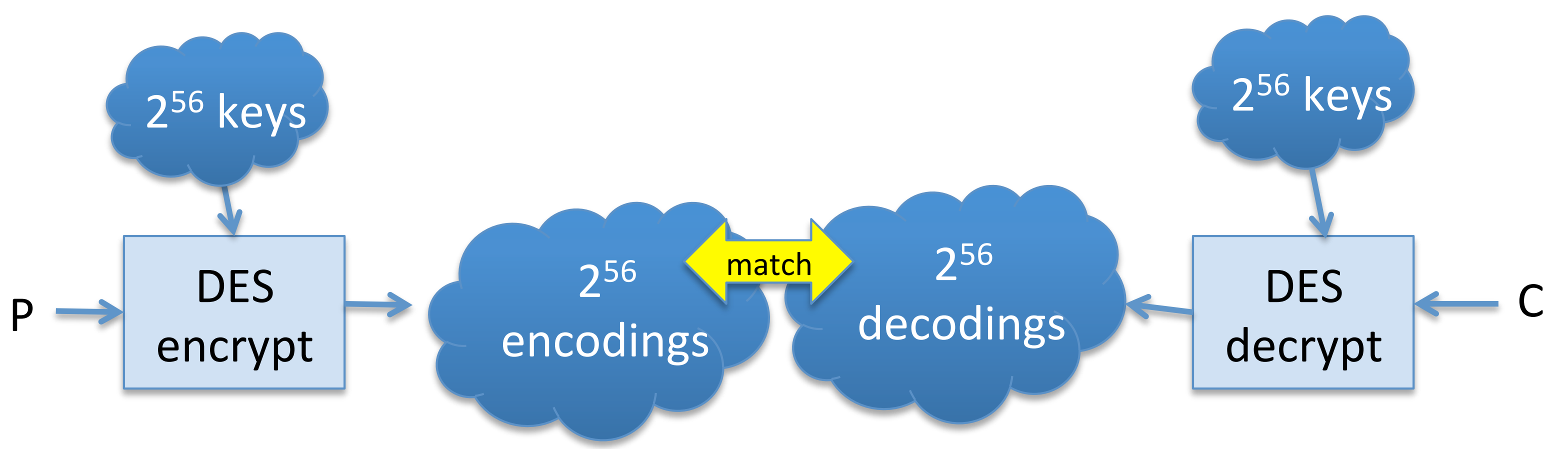
- Considered too weak
  - Diffie Hellman said in a few years technology would allow DES to be broken in days
    - Design using 1999 technology published
  - Design decisions not public
    - NSA controlled process
    - Some of the design decisions underlying the S-Boxes are unknown
    - S-boxes may have backdoors
    - Key size reduced from 112 bits in original Lucifer design to 56 bits

Brute Force Attack

- C= E(P, K), 56-bit keys, 2^{56} possibilities
- Why not $C = E(\ E(P, K), K)$?
  - Trick question --- it's still just 56 bit key
- Why not $C = E(\ E(P, K_1), K_2)$? (Double DES)

Double DES

- Double encryption not generally used
  - $C = E_{k2}(E_{k1}(P))$
  - Encode twice, using 2 different keys
  - Susceptible to **"Meet in the Middle (MTM) attack"**
  - Suppose you have plaintext P and corresponding ciphertext C



Modifies brute force to require only 2^{n+1} steps instead of 2^{2n}

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)
  - $C = E(D(E(P, K_1), K_2), K_1)$
  - $P = D(E(D(C, K_1), K_2), K_1)$
- Why Encrypt-Decrypt-Encrypt with 2 keys?
  - Backward compatible: $E(D(E(P,K),K),K) = E(P,K)$
  - And 112 bits is enough

AES Background

- Clear a replacement for DES was needed
  - Can use Triple-DES, but slow with small blocks
- US NIST issued call for ciphers in 1997
  - 15 candidates accepted in Jun 98
  - 5 were short-listed in Aug-99
- Rijndael was selected as AES in Oct-2000
  - issued as FIPS PUB 197 standard in Nov-2001
  - http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- Iterated block cipher (like DES)
- Not a Feistel cipher (unlike DES)

AES Overview

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

AES ByteSub

- Treat a 128-bit block as 4x4 byte array

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \texttt{ByteSub} \longrightarrow \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

- ByteSub is AES's "S-box"
- Can be viewed as nonlinear (but invertible) composition of two math operations

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

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \texttt{ShiftRow} \longrightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{bmatrix}$$

AES MixColumn

- Invertible, linear operation applied to each column

$$\begin{bmatrix} a_{0i} \\ a_{1i} \\ a_{2i} \\ a_{3i} \end{bmatrix} \longrightarrow \texttt{MixColumn} \longrightarrow \begin{bmatrix} b_{0i} \\ b_{1i} \\ b_{2i} \\ b_{3i} \end{bmatrix} \quad \text{for } i = 0, 1, 2, 3$$

- Implemented as a (big) lookup table

AES AddRoundKey

- XOR subkey with block

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \oplus \begin{bmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{bmatrix} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

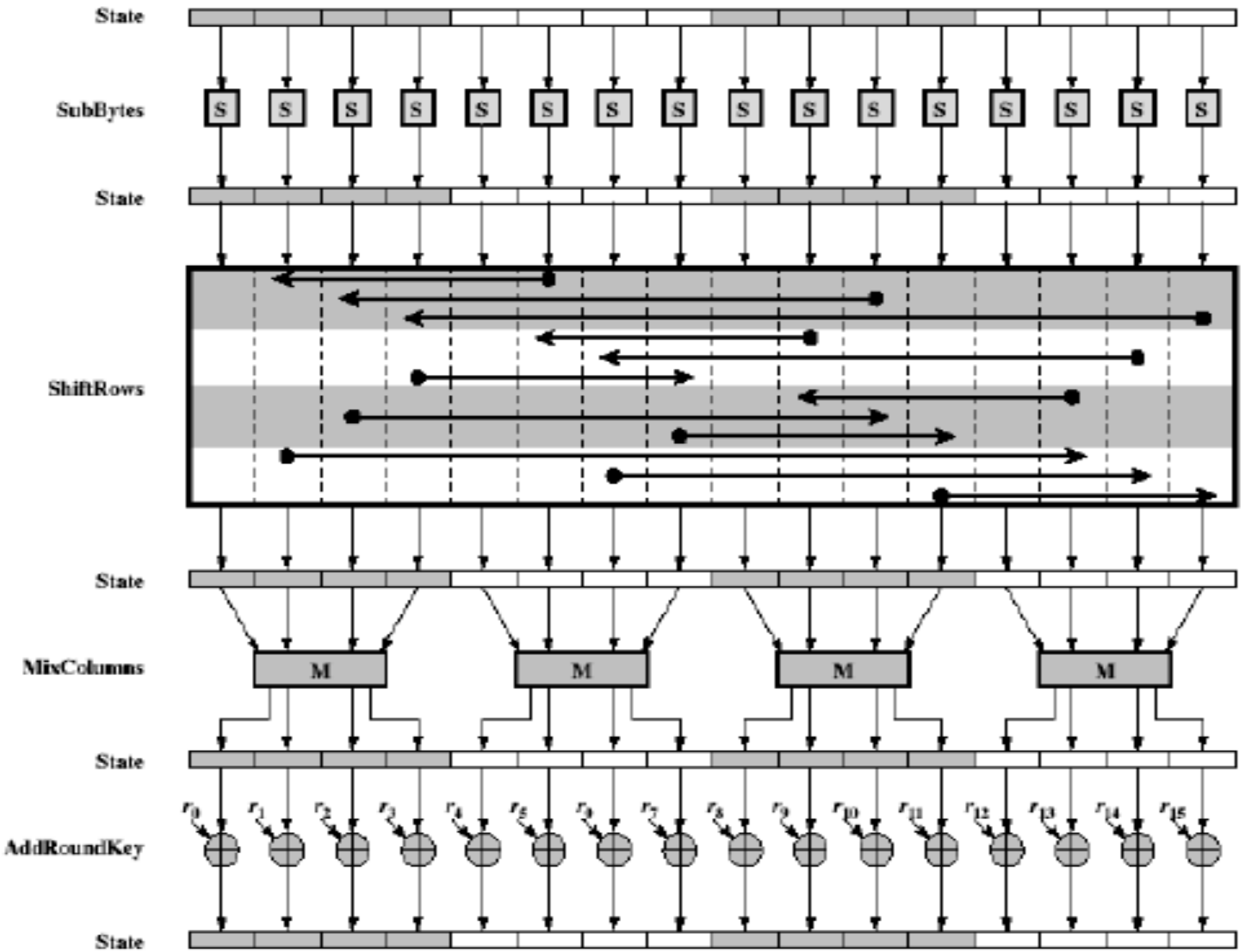
Block Subkey

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

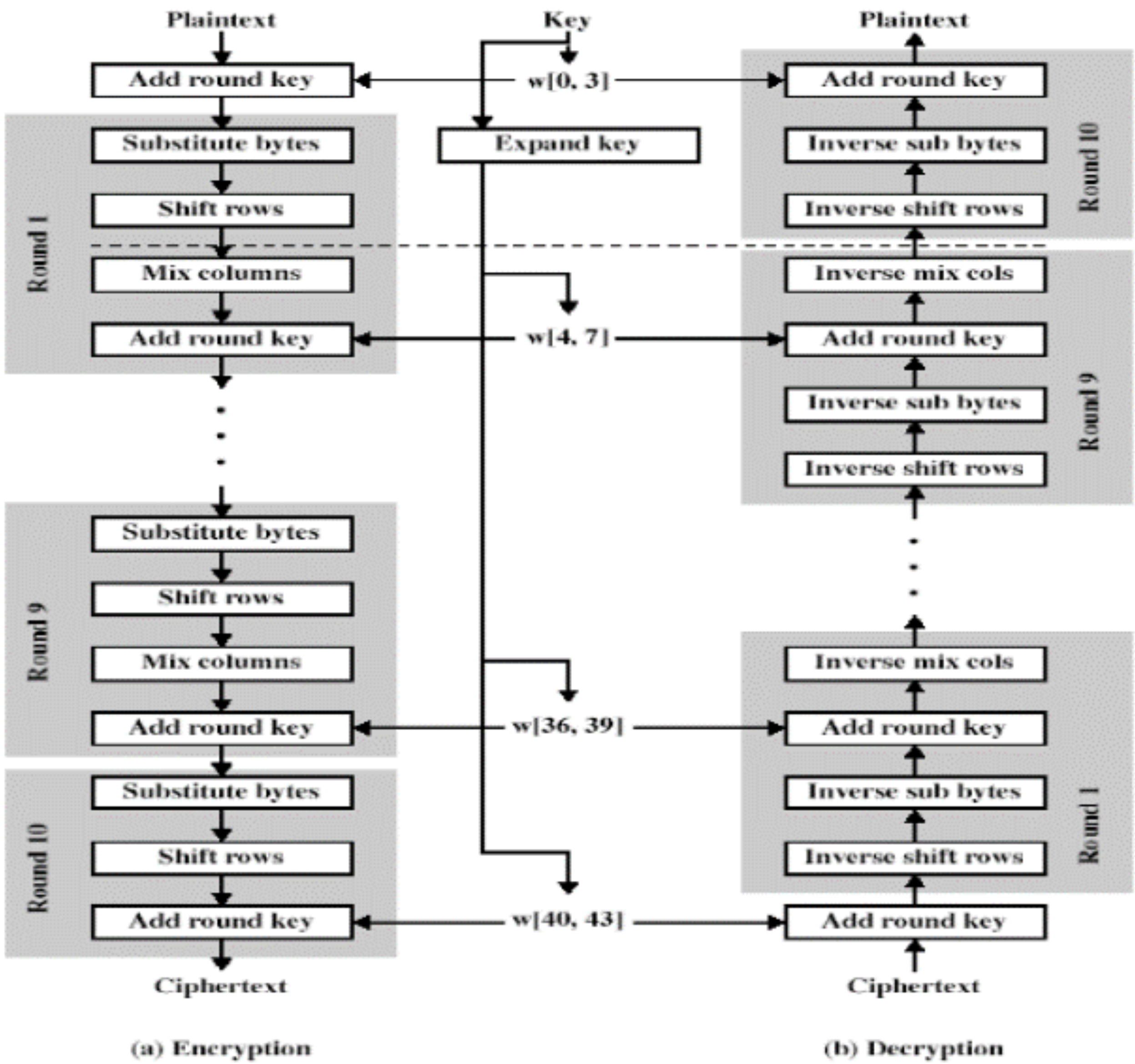
Algorithm Overview

- Processes data as 4 groups of 4 bytes (state)
- Has 9/11/13 rounds in which state undergoes:
  - Byte substitution (one S-box used on every byte)
  - Shift rows (permute bytes between groups/columns)
  - Mix columns (subs using matrix multiply of groups)
  - Add round key (XOR state with key material)
- All operations can be combined into XOR and table lookups, hence very fast & efficient

One AES Round



Rijndael



Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Shift rows
Inverse shift rows
Round 1
Round 10
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
Substitute bytes
Shift rows
Round 9
Mix columns
Inverse mix cols
Add round key
w[36, 39]
Add round key
Substitute bytes
Inverse sub bytes
Round 1
Shift rows
Inverse shift rows
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

AES Decryption

- To decrypt, process must be invertible
- Inverse of AddRoundKey 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)

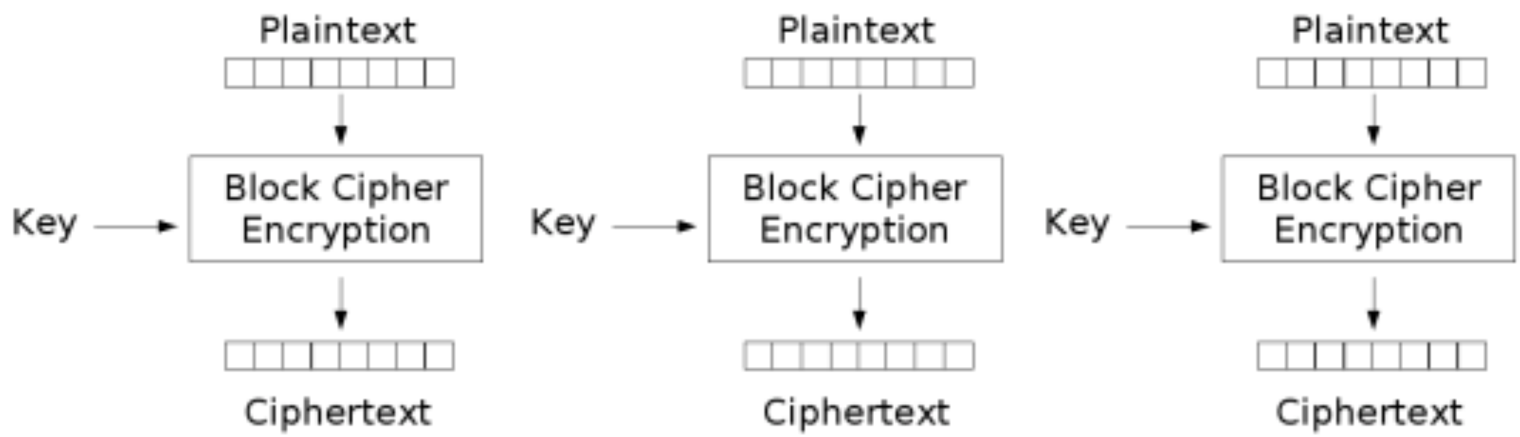
Attack on AES

- Only recently have some cryptoanalysis techniques been successful.
  - Biclique Cryptanalysis of the Full AES
    - http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf
    - But not yet a practical concern

Key recovery on AES-128 has complexity $2^{126.1}$
Key recovery on AES-192 has complexity $2^{189.7}$
Key recovery on AES-256 has complexity $2^{254.4}$

Block Ciphers

- Encipher, decipher multiple bits at once
- Each block enciphered independently
  - Electronic Code Book Mode (ECB)



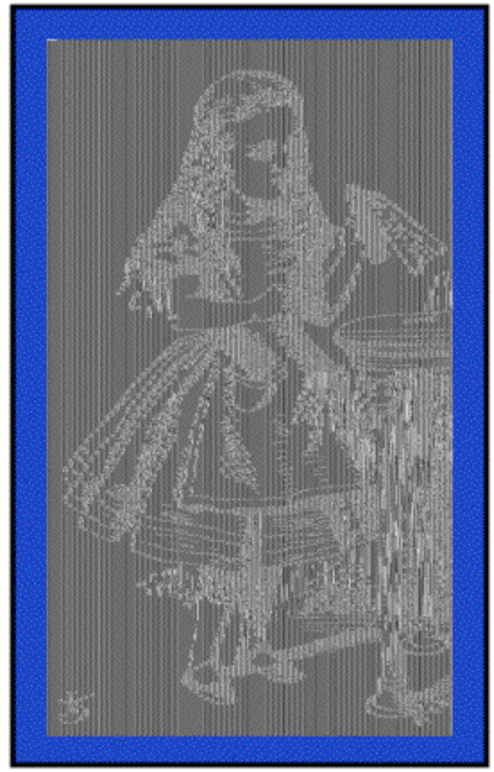
Electronic Codebook (ECB) mode encryption

ECB Problem

- Problem: identical plaintext blocks produce identical ciphertext blocks
  - Example: two database records
    - `MEMBER: HOLLY INCOME $100,000`
    - `MEMBER: HEIDI INCOME $100,000`
  - Encipherment:
    - `ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN`
    - `ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN`

Alice Hates ECB Mode

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



- ❑ Why does this happen?
- ❑ Same plaintext yields same ciphertext!

Solutions

- Insert information about block's position into the plaintext block, then encipher
  - Variety of ways one might encode "position"
  - $c_0 = E_k(m_0 \oplus I)$
  - $c_i = E_k(m_i \oplus i)$ for i>0, or
  - $c_i = E_k(m_i \oplus f(i))$ for i>0

Trick is to use something the receiver knows and so can apply XOR in reverse when decoding.

I is the initialization vector

Cipher block chaining (CBC) Mode

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

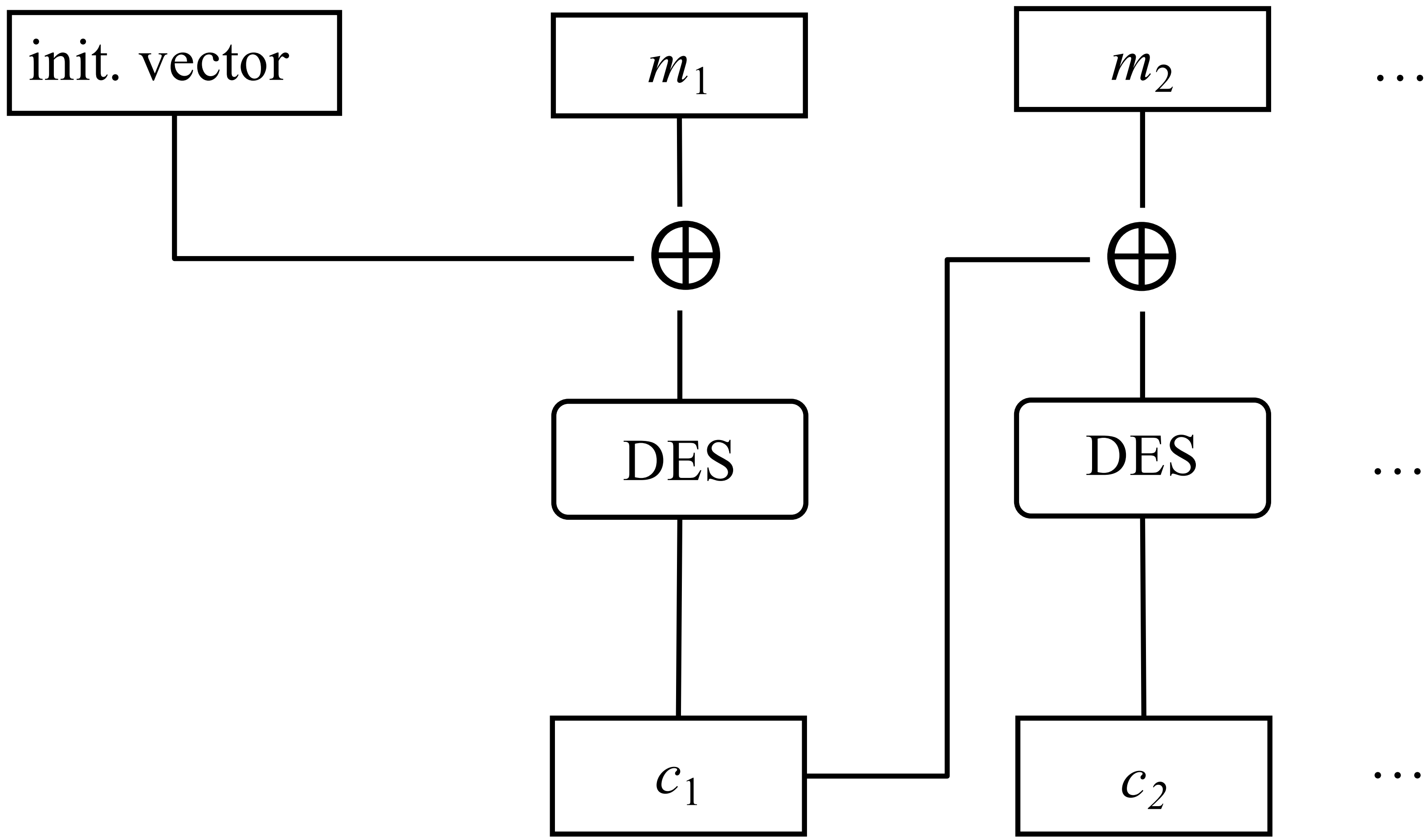
$P_0 = IV \oplus D(C_0, K),$

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

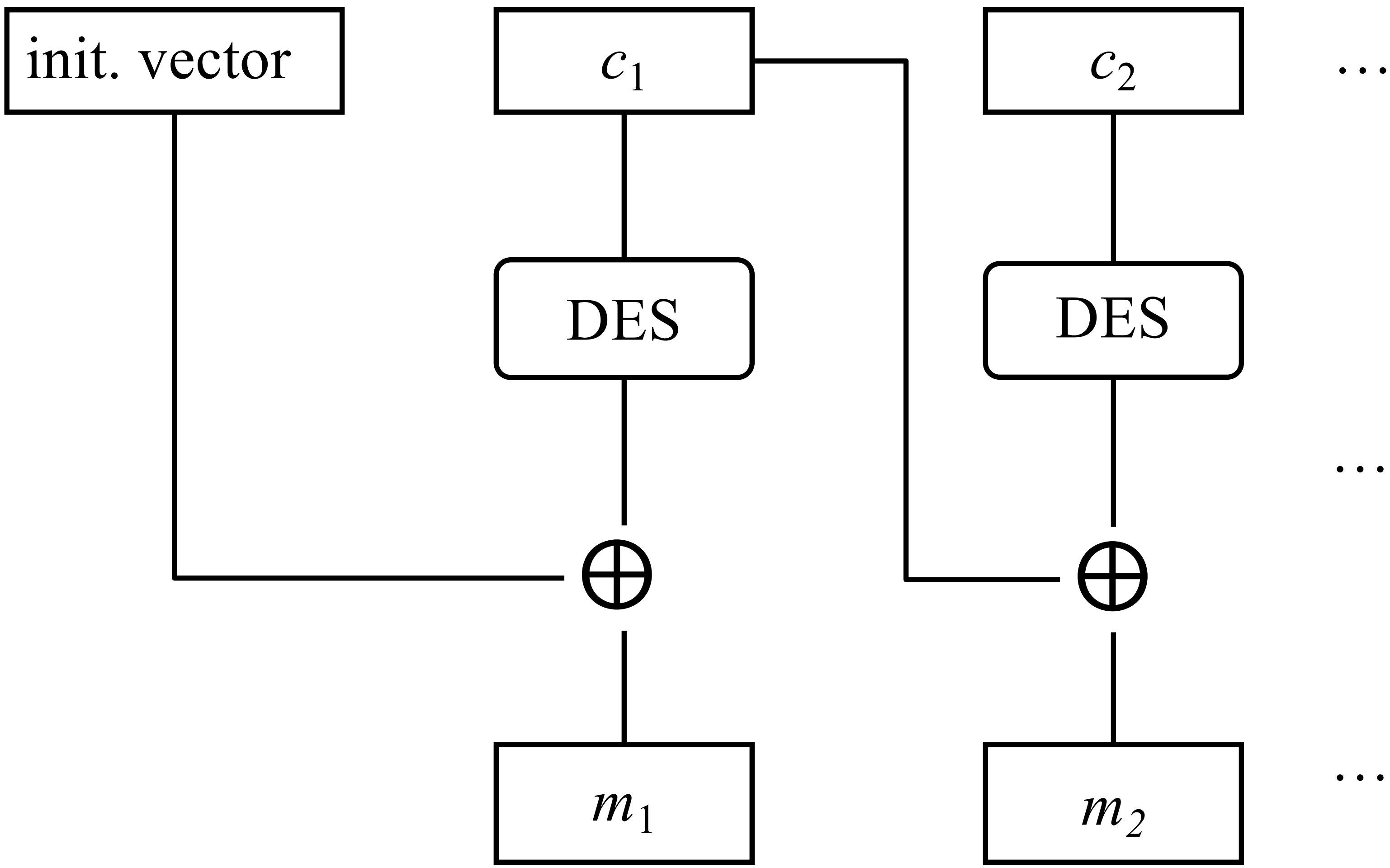
$P_2 = C_1 \oplus D(C_2, K),\ldots$

- Analogous to classic codebook *with additive*

CBC Mode Encryption



CBC Mode Decryption



CBC Mode

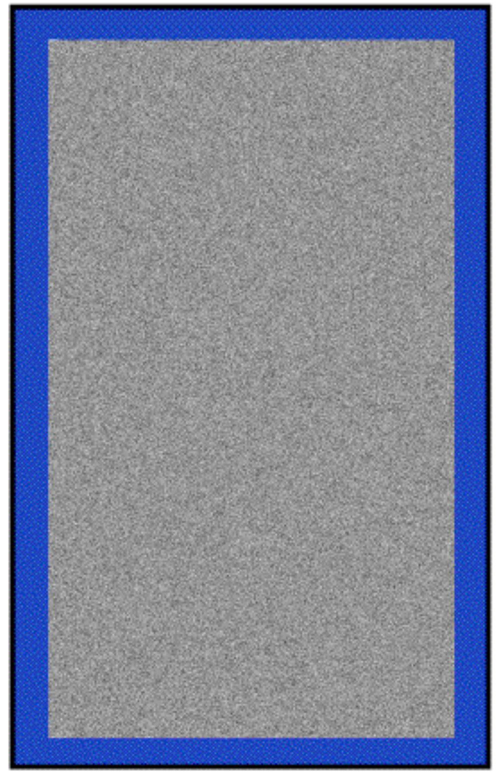
- Identical plaintext blocks yield different ciphertext blocks — this is good!
- If C_1 is garbled to, say, G then

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

- But $P_3 = C_2 \oplus D(C_3, K), P_4 = C_3 \oplus D(C_4, K), \ldots$
- Automatically recovers from errors! (self healing)
- Cut and paste is still possible, but more complex (and will cause garbles)

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 a stream cipher

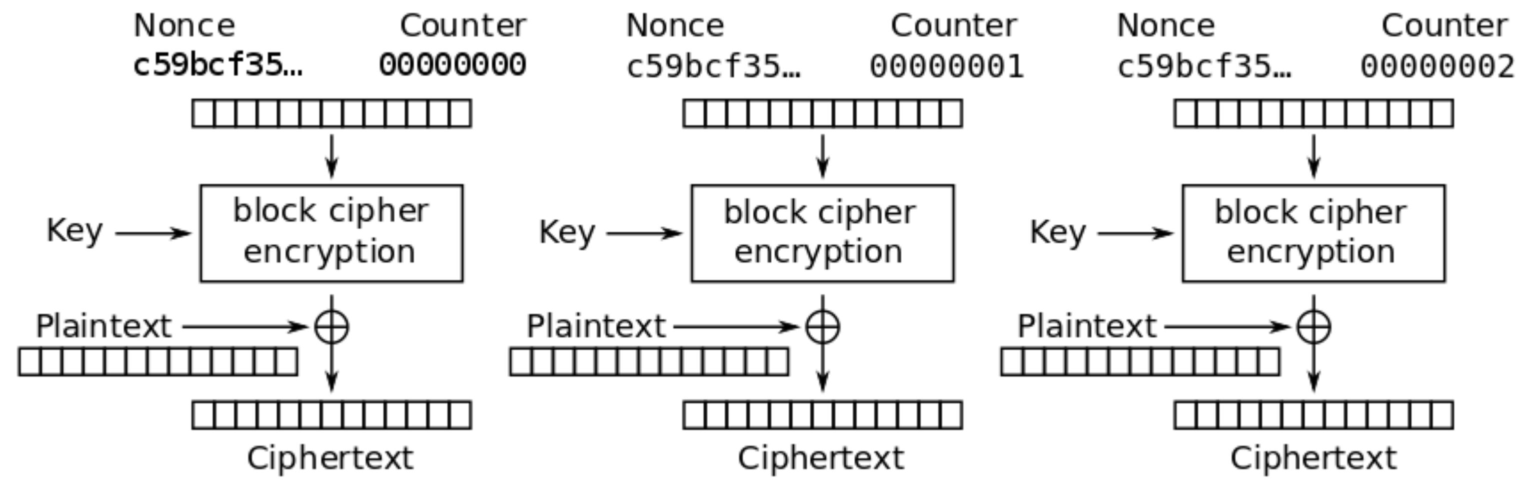
Encryption

$$C_0 = P_0 \oplus E(IV, K),$$
$$C_1 = P_1 \oplus E(IV+1, K),$$
$$C_2 = P_2 \oplus E(IV+2, K), \ldots$$

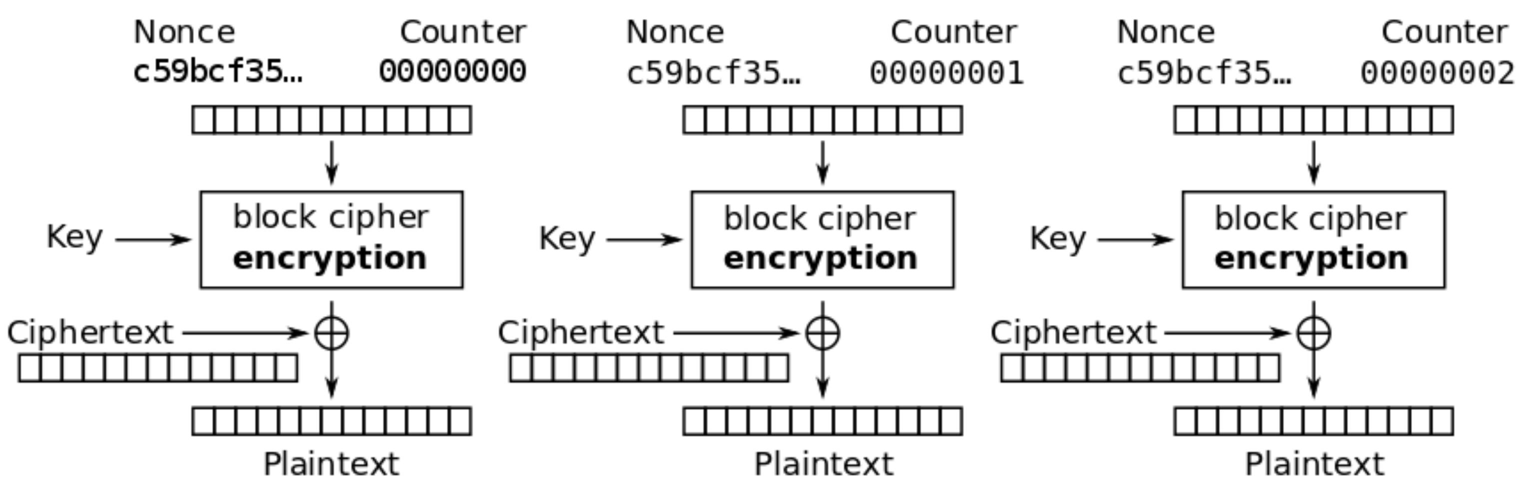
Decryption

$$P_0 = C_0 \oplus E(IV, K),$$
$$P_1 = C_1 \oplus E(IV+1, K),$$
$$P_2 = C_2 \oplus E(IV+2, K), \ldots$$

Counter Mode (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Key Points

- Symmetric key ciphers
  - AES and DES
  - Today's workhorse algorithms
  - Crypto analysis attacks on algorithms
  - Product ciphers
- Block Ciphers
- Stream ciphers