Symmetric Cryptography

CS461/ECE422

Fall 2009

Outline

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

Reading

- Chapter 9 from *Computer Science: Art and Science*
  - Sections 3 and 4
- *AES Standard issued as FIPS PUB 197*
  - http://csrc.nist.gov/publications/fips/fips197/fips
- *Handbook of Applied Cryptography*, Menezes, van Oorschot, Vanstone
  - Chapter 7
  - http://www.cacr.math.uwaterloo.ca/hac/

Stream, Block Ciphers

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

Examples

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

Confusion and Diffusion

- Confusion
  - Interceptor should not be able to predict how ciphertext will change by changing one character
- Diffusion
  - Cipher should spread information from plaintext over cipher text
  - See avalanche effect

Avalanche Effect

- Key desirable property of an encryption algorithm
- Where a change of **one** input or key bit results in changing approx **half of the** output bits
- 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

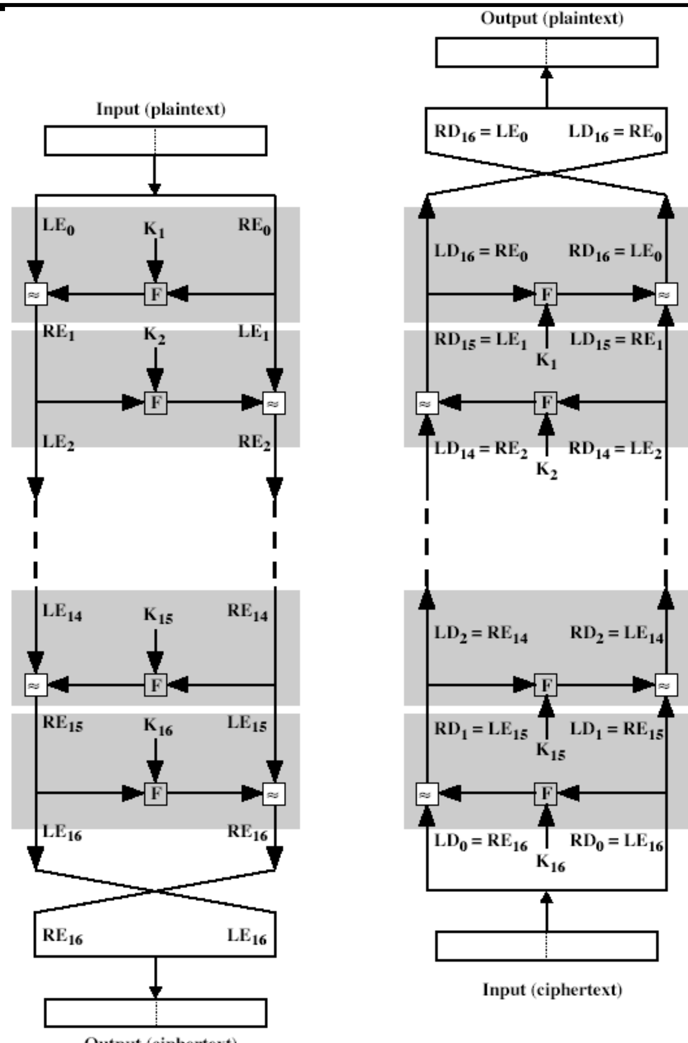
Overview of the DES

- A block cipher:
  - encrypts blocks of 64 bits using a 56 bit key
  - outputs 64 bits of ciphertext
- A product cipher
  - basic unit is the bit
  - performs both substitution (S-box) and transposition (permutation) (P-box) on the bits
- Cipher consists of 16 rounds (iterations) each with a round key generated from the user-supplied key

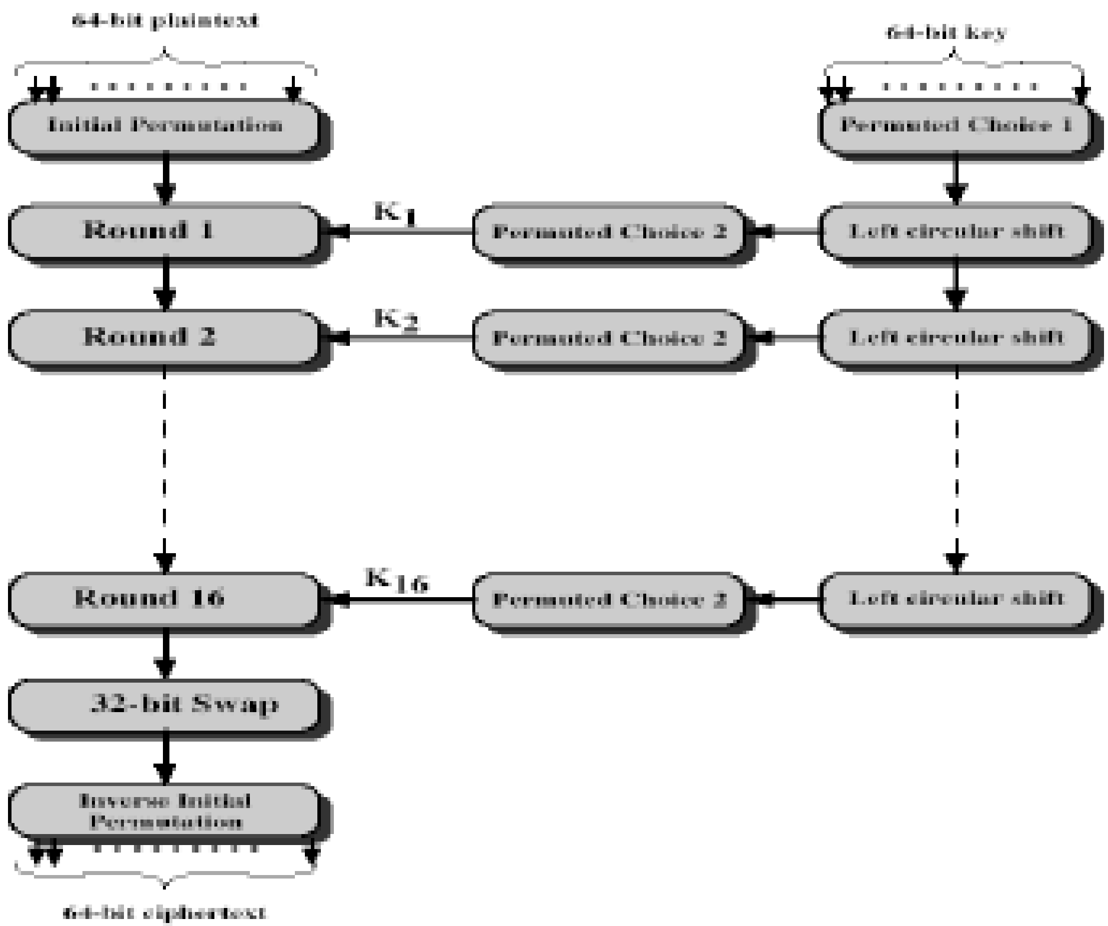
Feistel Network

- Structured to enable use of same S-box and P-box for encryption and decryption
  - Change only key schedule
- Major feature is key division and swapping
  - L(i) = R(i-1)
  - R(i) = L(i-1) xor f(K(i), R(i-1))

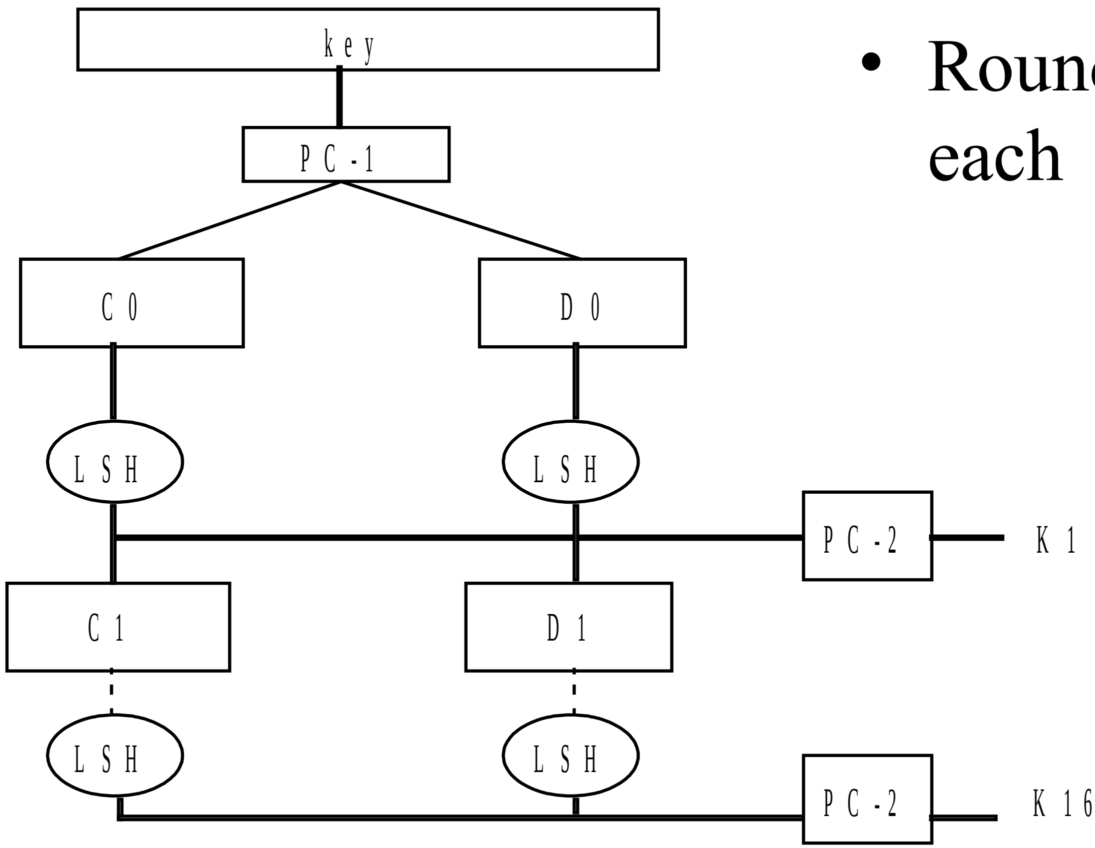
Feistel Structure Decryption



The Big Picture

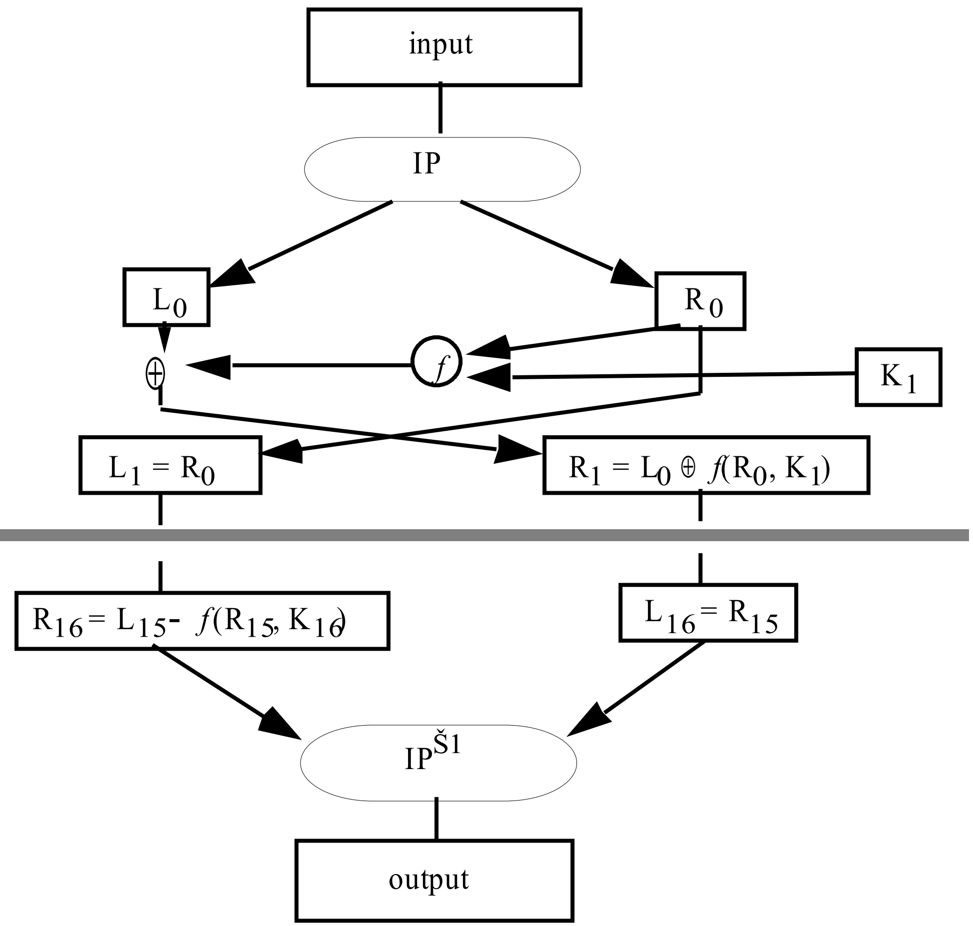


Generation of Round Keys



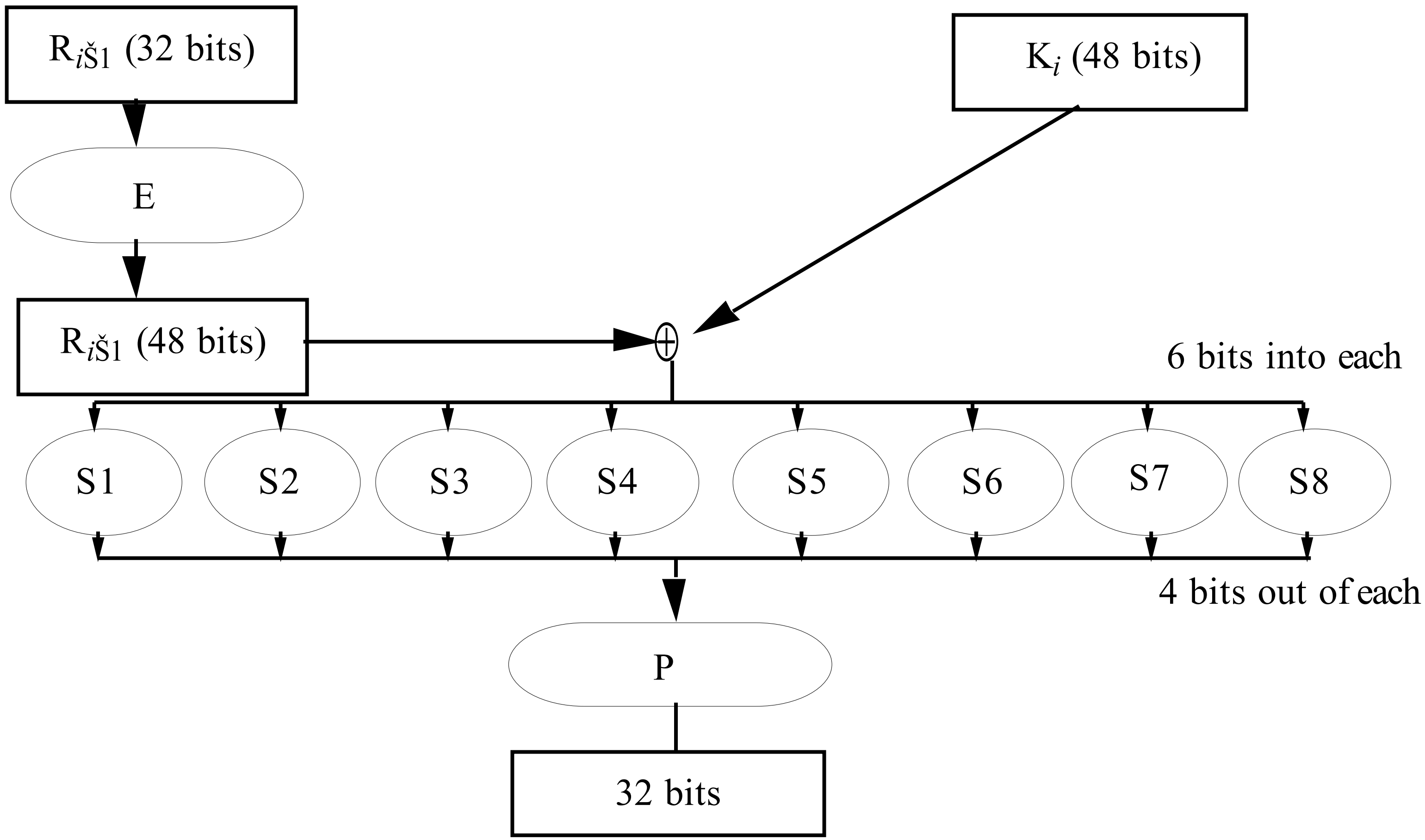
- Round keys are 48 bits each

Encryption



input
IP
L0
R0
f
K1
L1 = R0
R1 = L0 ⊕ f(R0, K1)
R16 = L15- f(R15, K16)
L16 = R15
IPŠ1
output

The *f* Function



Substitution boxes

- Key non-linear element to DES security
- have eight S-boxes which map 6 to 4 bits
  - outer bits 1 & 6 (**row**bits) select one rows
  - inner bits 2-5 (**col**bits) select column
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)
- example:
  - S(18 09 12 3d 11 17 38 39) = 5fd25e03

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
  - 1st round with SK16 undoes 16th encrypt round
  - ….
  - 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP thus recovering original data value

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

Undesirable Properties

- 4 weak keys
  - They are their own inverses
  - *i.e.* $\text{DES}_k(m) = c \Rightarrow \text{DES}_k(c) = m$
  - All 0's. All 1's. First half 1's second half 0's. Visa versa.
- 12 semi-weak keys
  - Each has another semi-weak key as inverse
  - *i.e.* $\text{DES}_{k1}(m) = c \Rightarrow \text{DES}_{k2}(c) = m$
- Possibly weak keys
  - Result in same subkeys being used in multiple rounds
- Complementation property
  - $\text{DES}_k(m) = c \Rightarrow \text{DES}_{k'}(m') = c'$

Brute Force Attack

- What do you need?
- How many steps should it take?
- How can you do better?

Differential Cryptoanalysis

- Was not reported in open literature until 1990
  - Tracks probabilities of differences inputs matching differences in outputs
- Chosen ciphertext attack

Differential Cryptoanalysis

- Build table of probabilities of inputs and outputs per round
  - $\Delta m_{i+1} = m_{i+1}$ xor m'_{i+1}
  - $\Delta m_{i+1} = [m_{i-1}$ xor $f(m_i,K_i)]$ xor $[\ m'_{i-1}$ xor $f(m'_i, K_i)]$
  - $\Delta m_{i+1} = \Delta m_{i-1}$xor $[f(m_i,K_i)$ xor $f(m'_i, K_i)]$
- Compose probabilities per round

Differential Cryptoanalysis

- Revealed several properties
  - Small changes in S-boxes reduces the number of pairs needed
  - The method was known to designer team as early as 1974
- Not so useful to break DES
  - But very useful to analyze the security of Feistel Network systems

Differential Cryptoanalysis

- Lucifer – IBM precursor to DES
  - Broken in 30 pairs
- FEAL-N
  - DES with different numbers of iterations
  - FEAL-4 broken in 20 pairs
  - FEAL-8 broken in 10,000 pairs
- DES with 15 rounds broken in 2^{52} tests
- DES with 16 rounds broken in 2^{58} tests

Current Status of DES

- A design for computer system and an associated software that could break any DES-enciphered message in a few days was published in 1998
- Several challenges to break DES messages solved using distributed computing
- National Institute of Standards and Technology (NIST) selected Rijndael as Advanced Encryption Standard (AES), successor to DES
  - Designed to withstand attacks that were successful on DES
  - It can use keys of varying length (128, 196, or 256)

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-

AES Requirements

- Private key symmetric block cipher
  - 128-bit data, 128/192/256-bit keys
- Stronger & faster than Triple-DES
- Active life of 20-30 years (+ archival use)
- Provide full specification & design details
- Both C & Java implementations
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- Initial criteria:
  - security –effort to practically cryptanalyse
  - cost –computational
  - algorithm & implementation characteristics
- Final criteria
  - general security
  - software & hardware implementation ease
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

AES Shortlist

- Shortlist August-99:
  - MARS (IBM) -complex, fast, high security margin
  - RC6 (USA) -v. simple, v. fast, low security margin
  - Rijndael(Belgium) -clean, fast, good security margin
  - Serpent (Euro) -slow, clean, v. high security margin
  - Twofish(USA) -complex, v. fast, high security margin
- Subject to further analysis & comment
- Saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - which refined existing ciphers verses new proposals

The AES Cipher - Rijndael

- Designed by Rijmen-Daemenin Belgium
  - Has 128/192/256 bit keys, 128 bit data
- An **iterative** rather than **feistel** cipher
  - treats data in 4 groups of 4 bytes
  - 4x4 matrix in column major order
  - operates an entire block in every round
- Designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - Simple design

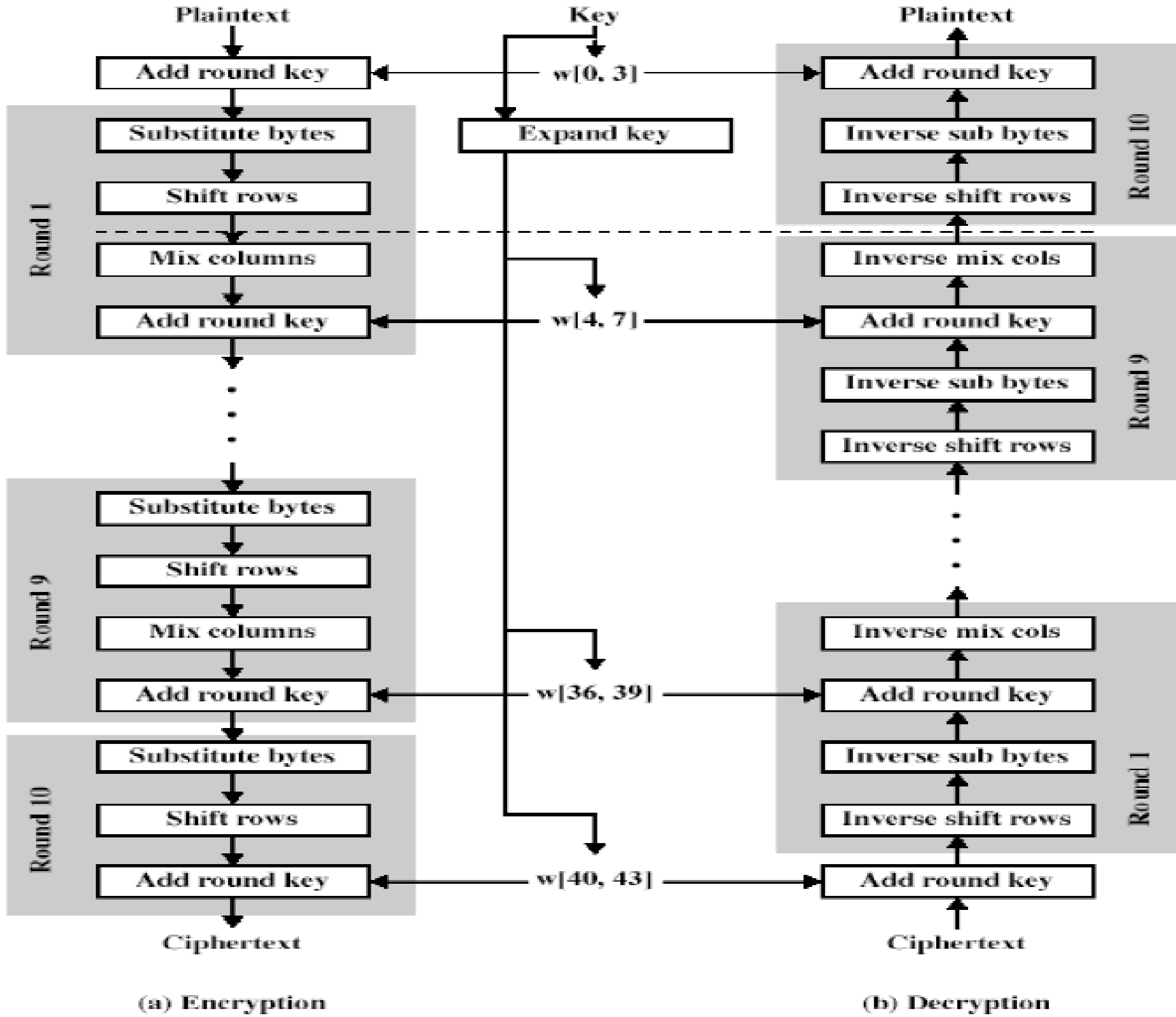
AES Block Matrix

In0	In4	In8	In12
In1	In5	In9	In13
In2	In6	In10	In14
In3	In7	In11	In15

Algorithm Overview

- Processes data as 4 groups of 4 bytes (state)
- Has 9/11/13 rounds in which state undergoes:
  - Byte substitution (1 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

Rijndael



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

Byte Substitution

- A simple substitution of each byte
- Uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- Each byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
- S-box is constructed using a defined transformation of the values in $GF(2^8)$
- Designed to be resistant to all known attacks

Shift Rows

- A circular byte shift in each row
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- Decrypt does shifts to right
- Since state is stored by columns, this step permutes bytes between the columns

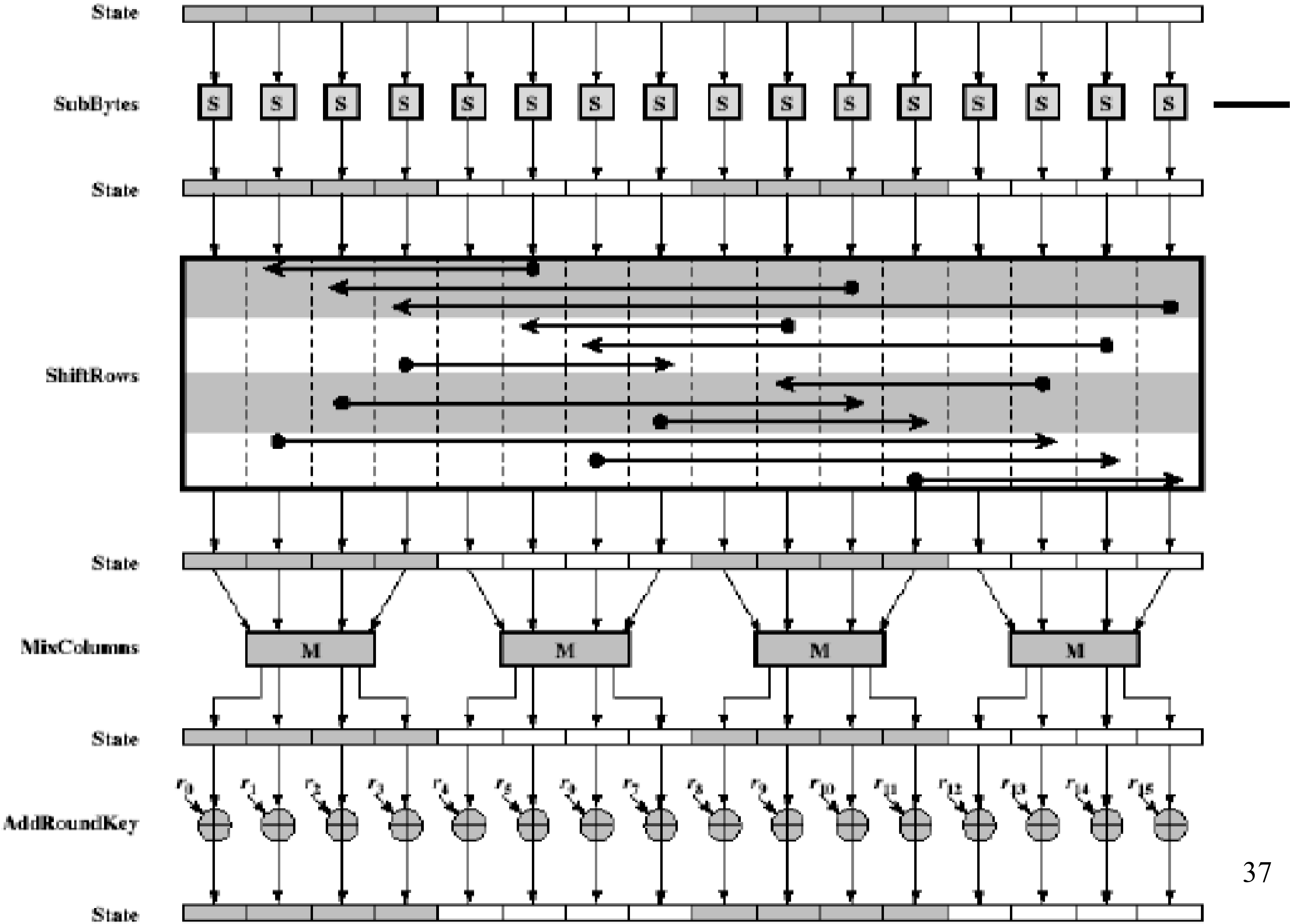
Mix Columns

- Each column is processed separately
- Each byte is replaced by a value dependent on all 4 bytes in the column
- Effectively a matrix multiplication in $GF(2^8)$ using prime poly $m(x) = x^8+x^4+x^3+x+1$

Add Round Key

- XOR state with 128-bits of the round key
- Again processed by column (though effectively a series of byte operations)
- Inverse for decryption is identical since XOR is own inverse, just with correct round key
- Designed to be as simple as possible

AES Round



State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
State

AES Key Expansion

- Takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- Start by copying key into first 4 words
- Then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - every 4th has S-box + rotate + XOR constant of previous before XOR together
- Designed to resist known attacks

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- But can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- Works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

Implementation Issues

- Can be efficiently implemented on 8-bit CPU
  - Byte substitution works on bytes using a table of 256 entries
  - Shift rows is simple byte shifting
  - Add round key works on byte XORs
  - Mix columns requires matrix multiply in $GF(2^8)$ on byte values, can be simplified to use a table lookup

Block Ciphers

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

ECB Problem

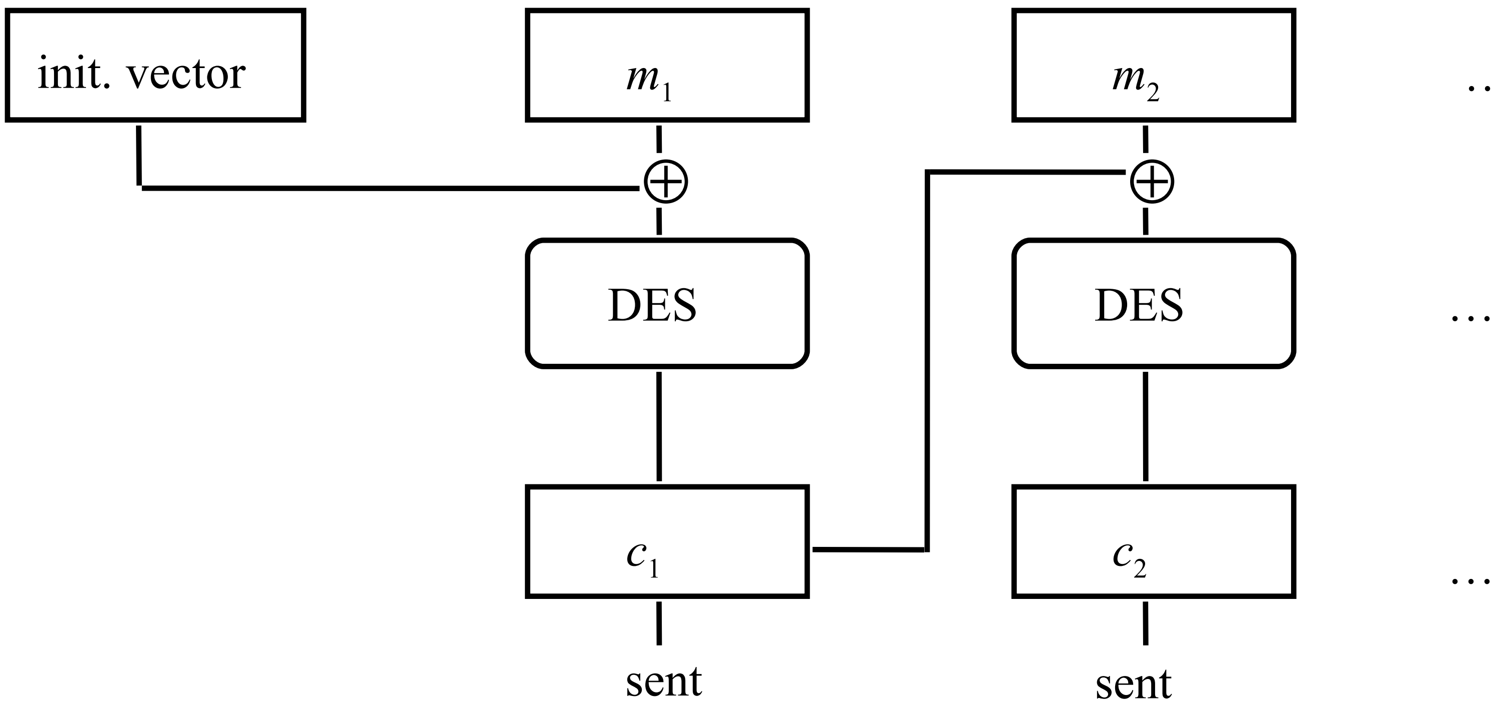
- Problem: identical plaintext blocks produce identical ciphertext blocks
  - Example: two database records
    - `MEMBER: HOLLY INCOME $100,000`
    - `MEMBER: HEIDI INCOME $100,000`
  - Encipherment:
    - `ABCQZRME GHQMRSIB CTXUVYSS RMGRPFQN`
    - `ABCQZRME ORMPABRZ CTXUVYSS RMGRPFQN`

Solutions

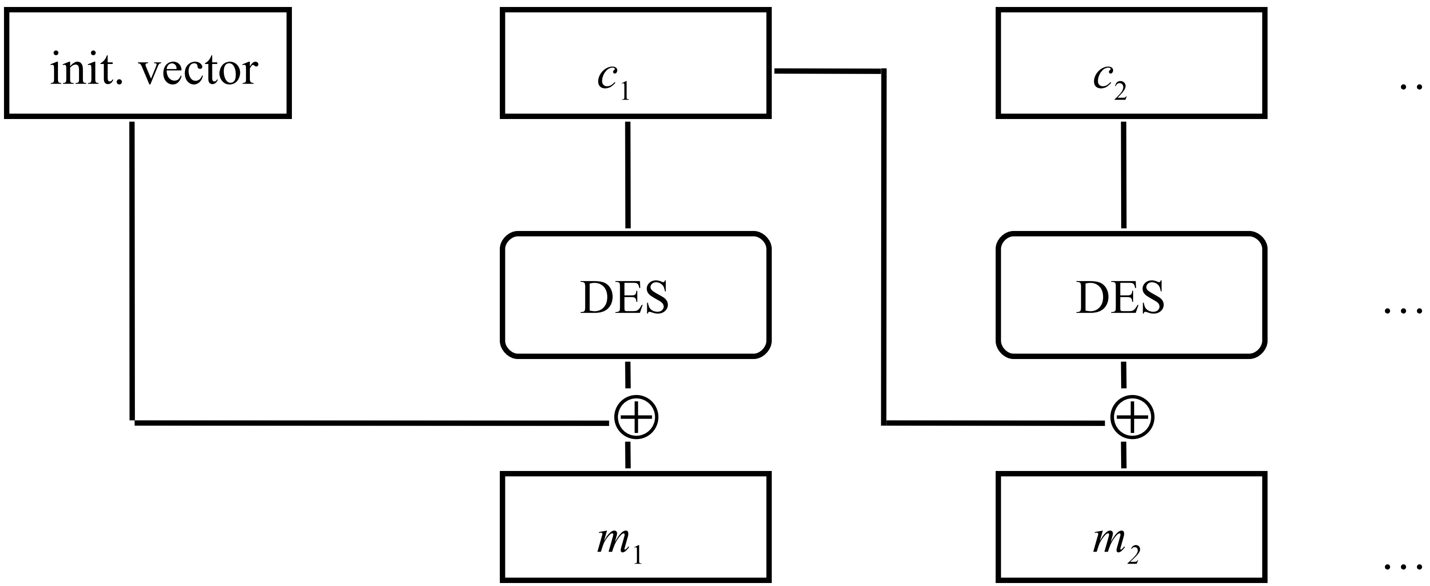
- Insert information about block's position into the plaintext block, then encipher
- *Cipher block chaining (CBC)*:
  - Exclusive-or current plaintext block with previous ciphertext block:
    - $c_0 = E_k(m_0 \oplus I)$
    - $c_i = E_k(m_i \oplus c_{i-1})$ for i > 0

  where I is the initialization vector

CBC Mode Encryption



CBC Mode Decryption



Self-Healing Property

- If one block of ciphertext is altered, the error propagates for at most two blocks
- Initial message
  - `3231343336353837 3231343336353837 3231343336353837 3231343336353837`
- Received as (underlined 4c should be 4b)
  - `ef7c4cb2b4ce6f3b f6266e3a97af0e2c 746ab9a6308f4256 33e60b451b09603d`
- Which decrypts to
  - `efca61e19f4836f1 3231333336353837 3231343336353837 3231343336353837`
  - Incorrect bytes underlined
  - Plaintext "heals" after 2 blocks

Multiple Encryptions

- Double encryption not generally used
  - Meet-in-the-middle attack
  - $C = E_{k2}(E_{k1}(P))$
  - Modifies brute force to require only 2^{n+1} steps instead of 2^{2n}
- Encrypt-Decrypt-Encrypt Mode (2 or 3 keys: k, k')
  - $c = DES_k(DES_{k'}^{-1}(DES_{k''}(m)))$
  - Also called Triple DES or 3DES when used with 3 keys
  - 168 bits of key, but effective key length of 112 due to meet-in-the middle
  - Not yet practical to break but AES much faster
- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
  - $c = DES_k(DES_{k'}(DES_{k''}(m)))$

Stream Ciphers

- Often (try to) implement one-time pad by xor'ing each bit of key with one bit of message
  - Example:

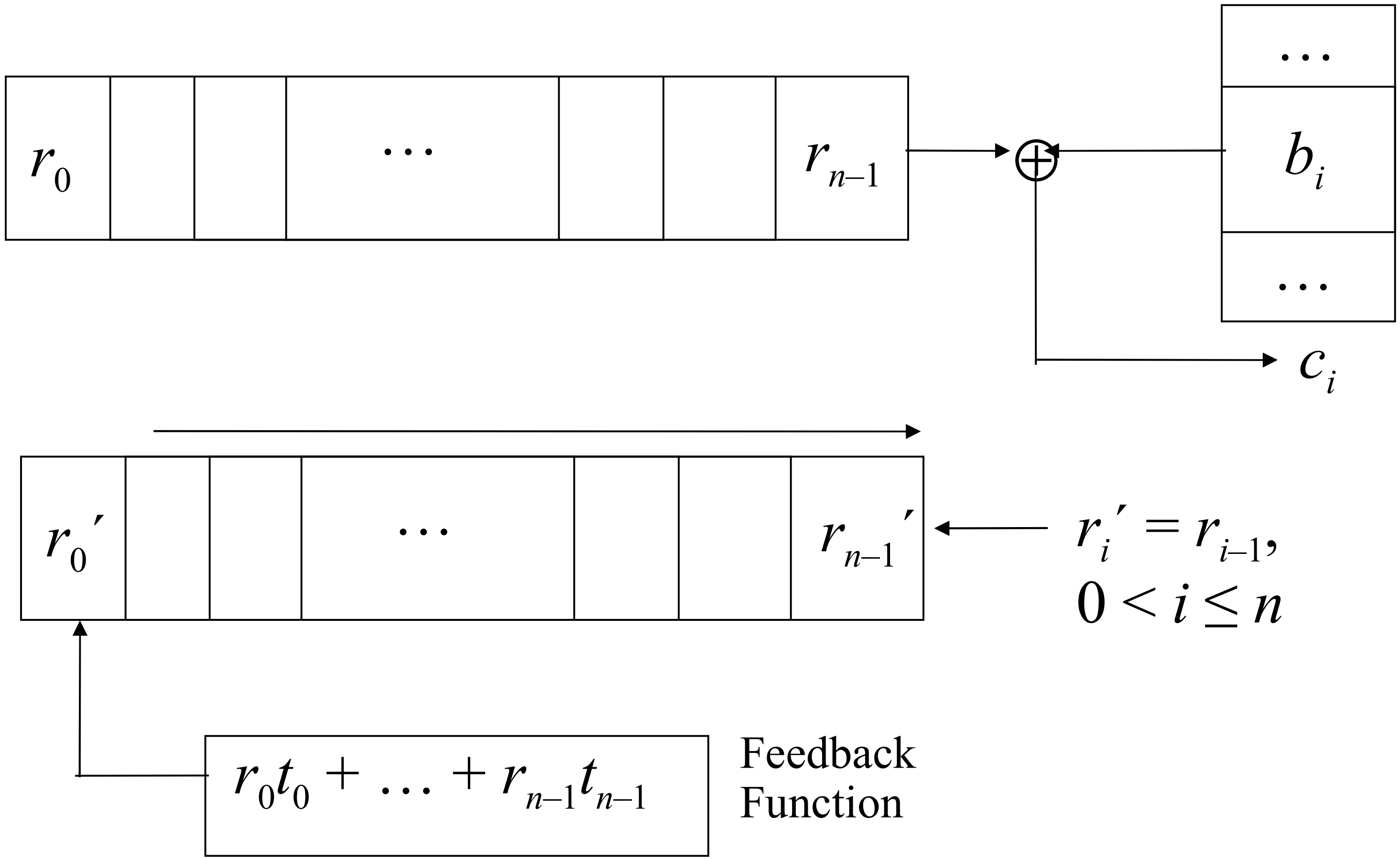
$$m = 00101$$
$$k = 10010$$
$$c = 10111$$

- But how to generate a good key?

Synchronous Stream Ciphers

- n-stage Linear Feedback Shift Register: consists of
  - n bit register $r = r_0 \ldots r_{n-1}$
  - n bit tap sequence $t = t_0 \ldots t_{n-1}$
  - Use:
    - Use r_{n-1} as key bit
    - Compute $x = r_0 t_0 \oplus \ldots \oplus r_{n-1} t_{n-1}$
    - Shift r one bit to right, dropping r_{n-1}, x becomes r_0

Operation



r_0
…
r_{n-1}
$\oplus$
…
b_i
…
c_i
r_0'
…
r_{n-1}'
$r_i' = r_{i-1},$
$0 < i \leq n$
$r_0 t_0 + \ldots + r_{n-1} t_{n-1}$
Feedback
Function

Example

- 4-stage LFSR; t = 1001

r	k_i	*new bit computation*	*new r*
0010	0	01⊕00⊕10⊕01 = 0	0001
0001	1	01⊕00⊕00⊕11 = 1	1000
1000	0	11⊕00⊕00⊕01 = 1	1100
1100	0	11⊕10⊕00⊕01 = 1	1110
1110	0	11⊕10⊕10⊕01 = 1	1111
1111	1	11⊕10⊕10⊕11 = 0	0111

– 00 11⊕10⊕10⊕11 = 1 1011

– Key sequence has period of 15 (010001111010110)

LFSR Period

- For n bit register
  - Maximum possible period is 2^n-1
  - -1 because 0’s will only yield 0’s
- Not all tap sequences will yield this period
  - Large theory on computing maximal period feedback functions

NLFSR

- n-stage Non-Linear Feedback Shift Register: consists of
  - n bit register $r = r_0 \ldots r_{n-1}$
  - Use:
    - Use r_{n-1} as key bit
    - Compute $x = f(r_0, \ldots, r_{n-1})$; f is any function
    - Shift r one bit to right, dropping r_{n-1}, x becomes r_0

Note same operation as LFSR but more general bit replacement function

Example

- 4-stage NLFSR; $f(r_0, r_1, r_2, r_3) = (r_0 \,\&\, r_2) \mid r_3$

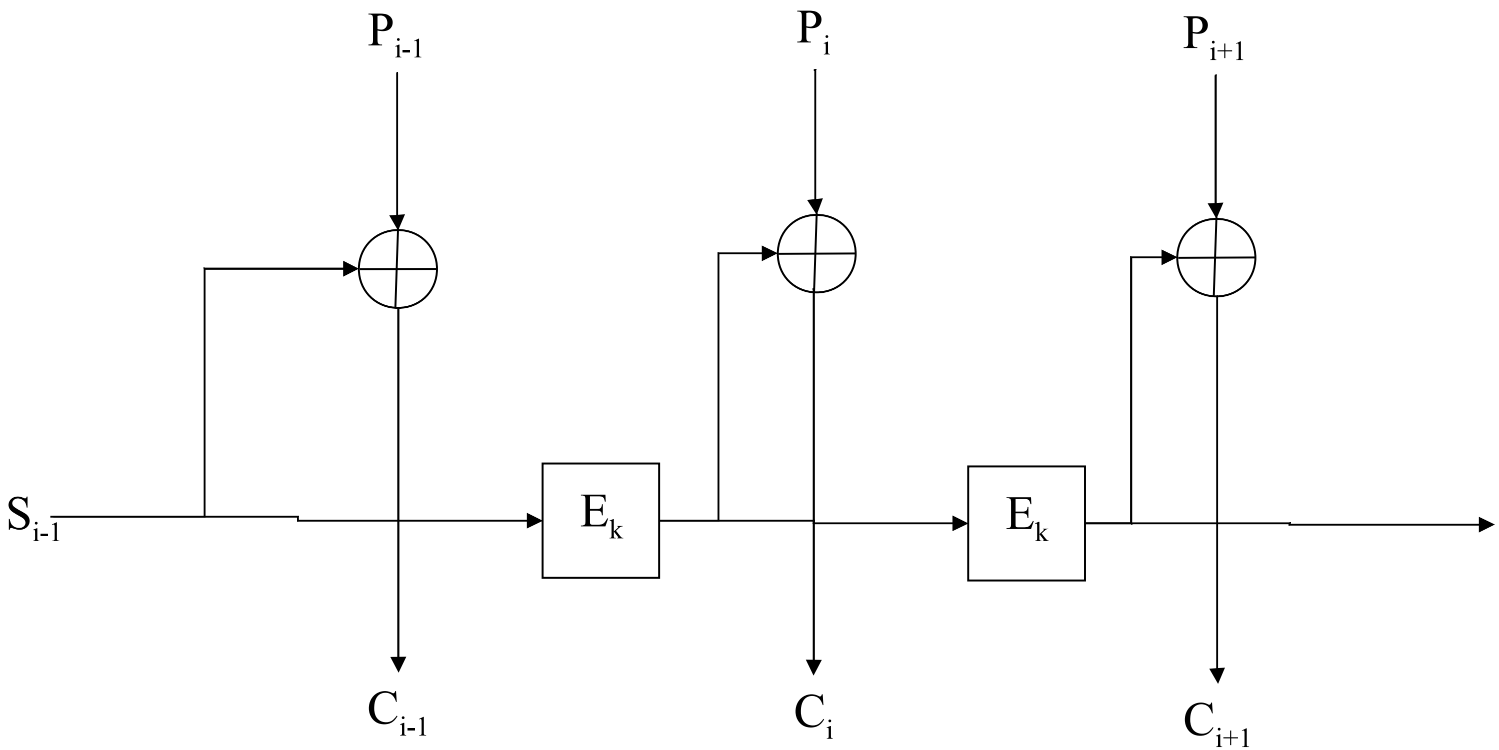
r	k_i	*new bit computation*	*new r*
`1100`	`0`	`(1 & 0)	0 = 0`	`0110`
`0110`	`0`	`(0 & 1)	0 = 0`	`0011`
`0011`	`1`	`(0 & 1)	1 = 1`	`1001`
`1001`	`1`	`(1 & 0)	1 = 1`	`1100`
`1100`	`0`	`(1 & 0)	0 = 0`	`0110`
`0110`	`0`	`(0 & 1)	0 = 0`	`0011`
`0011`	`1`	`(0 & 1)	1 = 1`	`1001`

  – Key sequence has period of 4 (0011)

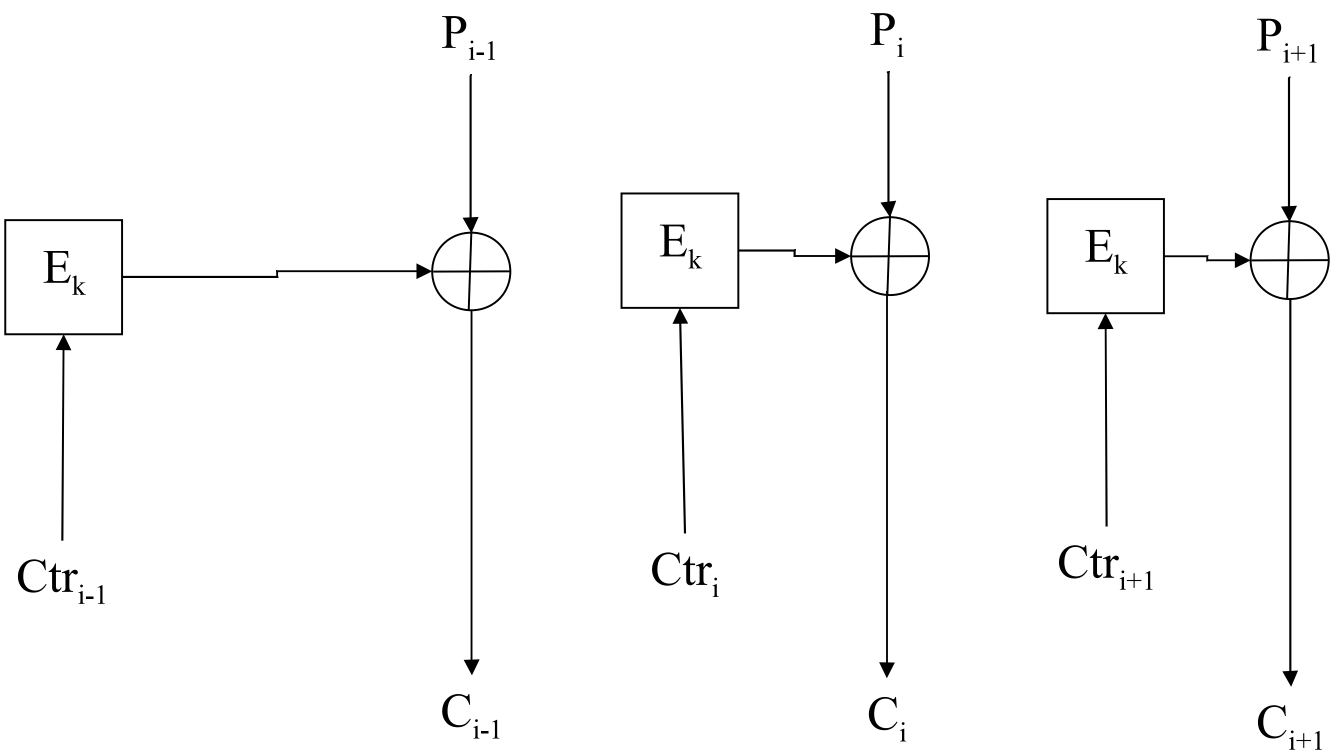
Eliminating Linearity

- NLFSRs not common
  - No body of theory about how to design them to have long period
- Alternate approach: *output feedback mode*
  - For E encipherment function, k key, r register:
    - Compute $r' = E_k(r)$; key bit is rightmost bit of r'
    - Set r to r' and iterate, repeatedly enciphering register and extracting key bits, until message enciphered
  - Variant: use a counter that is incremented for each encipherment rather than a register
    - Take rightmost bit of $E_k(i)$, where i is number of encipherment

OFB Mode



Counter Mode



Issues with OFB/Counter

- Additional standard modes for DES/AES
- Losing Synchronicity is fatal
  - All later decryptions will be garbled
- OFB needs an initialization vector
- Counter mode lets you generate a bit in the middle of the stream
- RC4 is a well-known stream cipher that uses OFB. Used in WEP

Self-Synchronous Stream Cipher

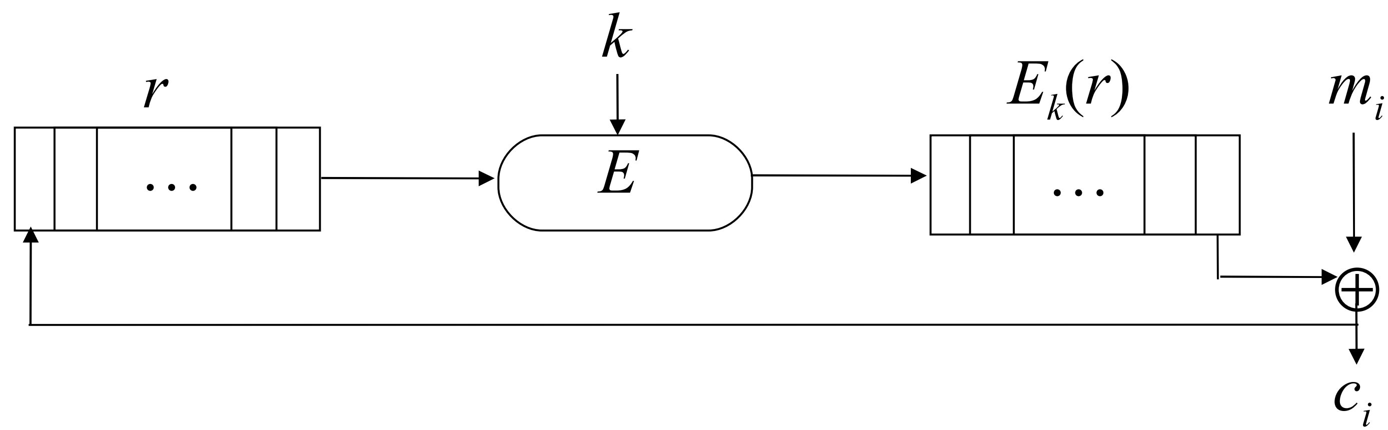
- Take key from message itself (*autokey*)
- Example: Vigenère, key drawn from plaintext
  - *key* `XTHEBOYHASTHEBA`
  - *plaintext* `THEBOYHASTHEBAG`
  - *ciphertext* `QALFPNFHSLALFCT`
- Problem:
  - Statistical regularities in plaintext show in key
  - Once you get any part of the message, you can decipher more

Another Example

- Take key from ciphertext (*autokey*)
- Example: Vigenère, key drawn from ciphertext
  - *key* `XQXBCQOVVNGNRTT`
  - *plaintext* `THEBOYHASTHEBAG`
  - *ciphertext* `QXBCQOVVNGNRTTM`
- Problem:
  - Attacker gets key along with ciphertext, so deciphering is trivial

Variant

- Cipher feedback mode: 1 bit of ciphertext fed into *n* bit register
  - Self-healing property: if ciphertext bit received incorrectly, it and next *n* bits decipher incorrectly; but after that, the ciphertext bits decipher correctly
  - Need to know *k*, *E* to decipher ciphertext



Key Points

- Historical Ciphers
  - Give examples of linguistic attacks
  - Substitution and transposition ciphers
- Symmetric key ciphers
  - AES and DES
  - Today's workhorse algorithms
  - Crypto analysis attacks on algorithms
  - Product ciphers