



THE UNIVERSITY OF BRITISH COLUMBIA

Symmetric Crypto Systems

EECE 412

Copyright © 2004-2007 Konstantin Beznosov

9/25/07

Module Outline

- Stream ciphers “under the hood”
- Block ciphers “under the hood”
- Modes of operation for block ciphers



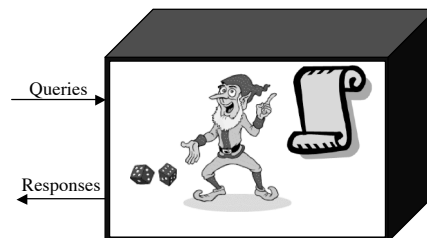
Stream Ciphers



3

Random Generator (Stream Cipher) as Random Oracle

- In:
 - short string (key)
 - length of the output



- Out: long random stream of bits (keystream)
- Applications:
 - Communications encryption
 - Storage encryption

Properties

- Should not reuse
 - Use *seed*



4

Stream Ciphers

- Not as popular today as block ciphers
- A5/1
 - Designed for hardware implementations
 - Based on shift registers
 - Used in GSM mobile phone system
- RC4
 - Designed for software implementations
 - Based on a changing lookup table
 - Used many places



5

A5/1

- A5/1 consists of 3 *shift registers*
 - X: 19 bits ($x_0, x_1, x_2, \dots, x_{18}$)
 - Y: 22 bits ($y_0, y_1, y_2, \dots, y_{21}$)
 - Z: 23 bits ($z_0, z_1, z_2, \dots, z_{22}$)



6

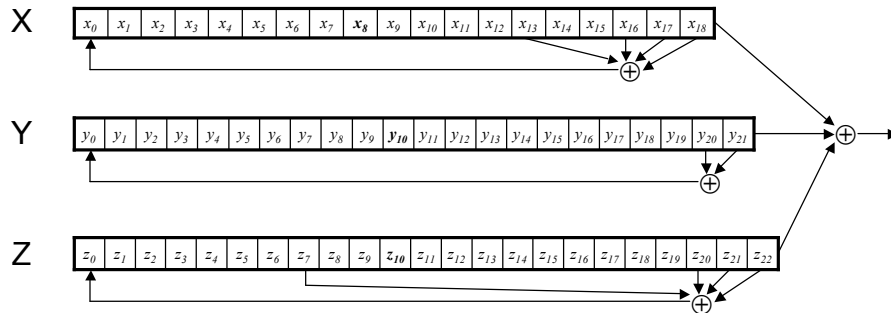
A5/1

- At each step: $m = \text{maj}(x_8, y_{10}, z_{10})$
 - Examples: $\text{maj}(0,1,0) = 0$ and $\text{maj}(1,1,0) = 1$
- If $x_8 = m$ then X steps
 - $t = x_{13} \oplus x_{16} \oplus x_{17} \oplus x_{18}$
 - $x_i = x_{i-1}$ for $i = 18, 17, \dots, 1$ and $x_0 = t$
- If $y_{10} = m$ then Y steps
 - $t = y_{20} \oplus y_{21}$
 - $y_i = y_{i-1}$ for $i = 21, 20, \dots, 1$ and $y_0 = t$
- If $z_{10} = m$ then Z steps
 - $t = z_7 \oplus z_{20} \oplus z_{21} \oplus z_{22}$
 - $z_i = z_{i-1}$ for $i = 22, 21, \dots, 1$ and $z_0 = t$
- Keystream bit is $x_{18} \oplus y_{21} \oplus z_{22}$



7

A5/1

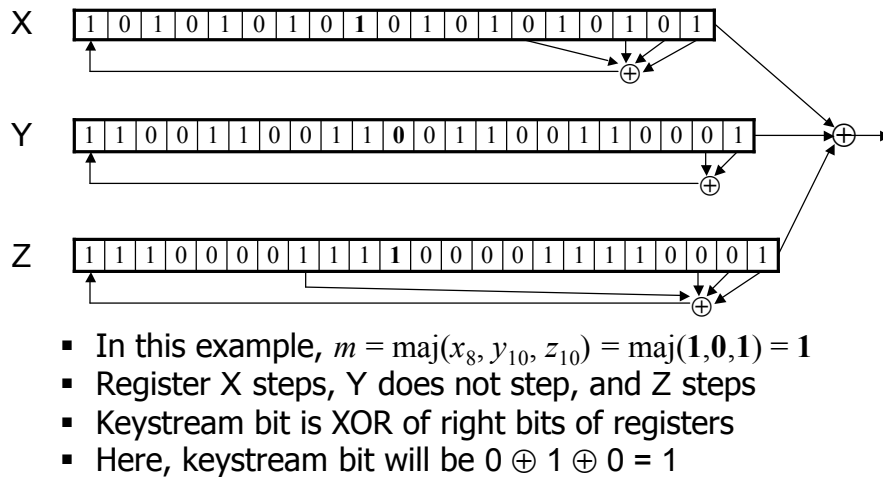


- Each value is a single bit
- Key is used as **initial fill** of registers
- Each register steps or not, based on (x_8, y_{10}, z_{10})
- Keystream bit is XOR of right bits of registers



8

A5/1: example



9



Shift Register Crypto

- Shift register-based crypto is efficient in hardware
- Harder to implement in software
- In the past, very popular
- Today, more is done in software due to faster processors
- Shift register crypto still used some

10



Use of Stream Ciphers

- Stream ciphers were big in the past
 - Efficient in hardware
 - Speed needed to keep up with voice, etc.
 - Today, processors are fast, so software-based crypto is fast enough

11

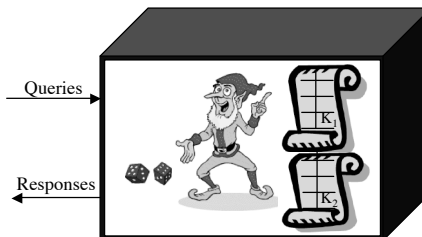


THE UNIVERSITY OF BRITISH COLUMBIA

Block Ciphers “Under the Hood”

Random Permutation (Block Cipher) as Random Oracle

- In
 - fixed size short string (plaintext) M ,
 - DES -- 64 bits
 - Key K



- Out
 - same fixed size short string (ciphertext) C

Notation

- $C = \{ M \}_K$
- $M = \{ C \}_K$



13

Related Notes

- Main properties of block ciphers
 - invertible
 - confusing
 - diffusing
- Main block ciphers
 - Data Encryption Standard (DES)
 - Advanced Encryption Standard (AES) a.k.a., Rijndael



14

(Iterated) Block Cipher

- Plaintext and ciphertext consists of fixed sized blocks
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of key and the output of previous round
- Usually implemented in software

15



Feistel Cipher

- type of block cipher design, not a specific cipher
- Split plaintext block into left and right halves:
Plaintext = (L_0, R_0)
- For each round $i=1,2,\dots,n$, compute
$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$
where F is **round function** and K_i is **subkey**
- Ciphertext = (L_n, R_n)

16



Feistel Cipher

- Decryption: Ciphertext = (L_n, R_n)
- For each round $i=n, n-1, \dots, 1$, compute
$$R_{i-1} = L_i$$
$$L_{i-1} = R_i \oplus F(R_{i-1}, K_i)$$
where F is round function and K_i is subkey
- Plaintext = (L_0, R_0)
- Formula “works” for any function F
- But only secure for certain functions F

17



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)

18



AES Overview

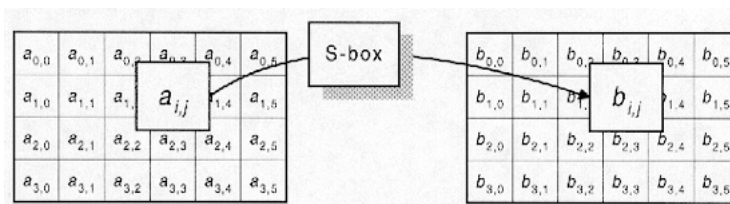
- **Block size:** 128, 192 or 256 bits
- **Key length:** 128, 192 or 256 bits (independent of block size)
- 10 to 14 rounds (depends on key length)
- Each round uses 4 functions (in 3 "layers")
 - ByteSub (nonlinear layer)
 - ShiftRow (linear mixing layer)
 - MixColumn (nonlinear layer)
 - AddRoundKey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

19



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

20



AES “S-box”

Last 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

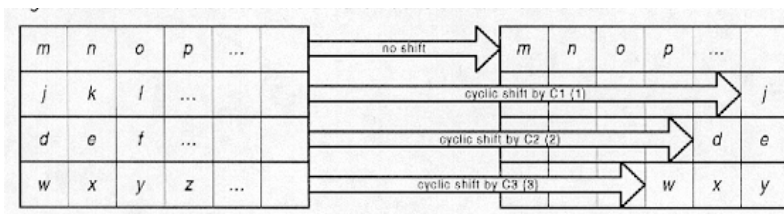
First 4 bits of input

21



AES ShiftRow

- Cyclic shift rows

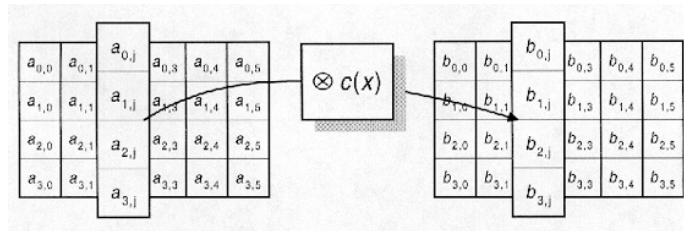


22



AES MixColumn

- Nonlinear, invertible operation applied to each column



- Implemented as a (big) lookup table

23



AES AddRoundKey

- XOR subkey with block

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

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

24



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)

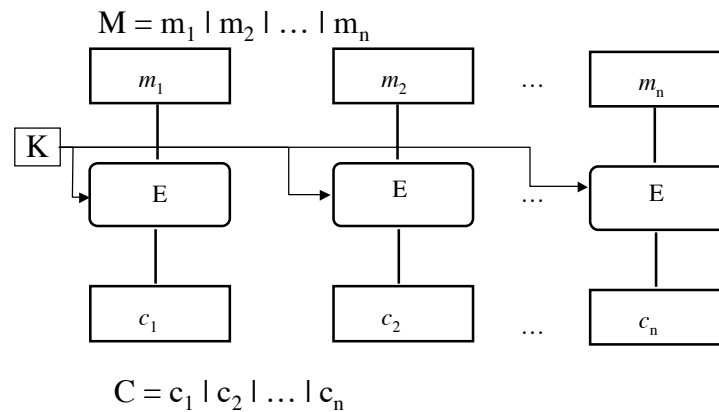
25



THE UNIVERSITY OF BRITISH COLUMBIA

Modes of Operation

Electronic Code Book (ECB)



Drawbacks

- Same message has same ciphertext
- Redundant/repetitive patterns will show through
- Subject to "cut-and-splice" attacks

27



Alice in ECB Mode

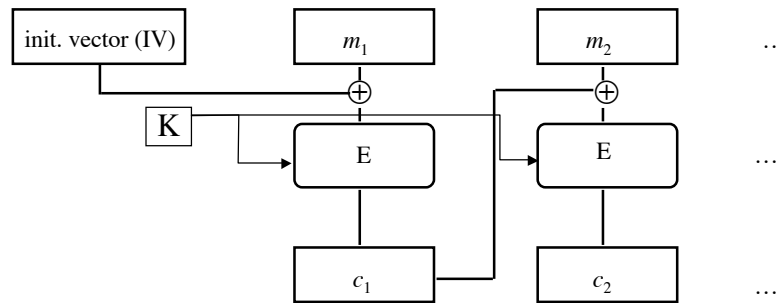


28



Cipher Block Chaining (CBC)

$$M = m_1 | m_2 | \dots | m_n$$

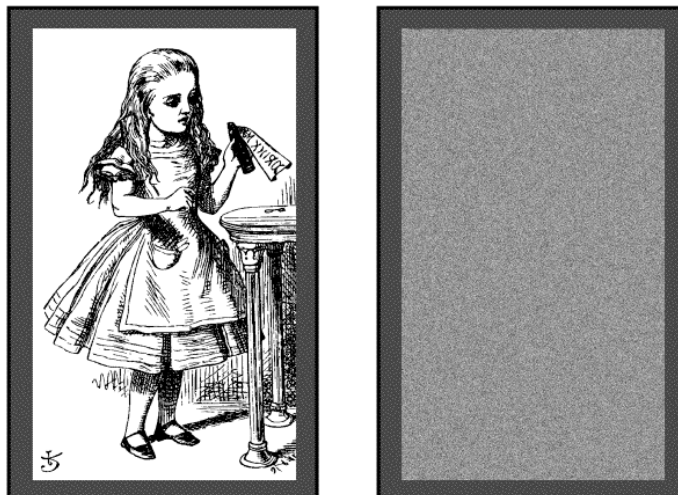


$$C = IV | c_1 | c_2 | \dots | c_n$$

29



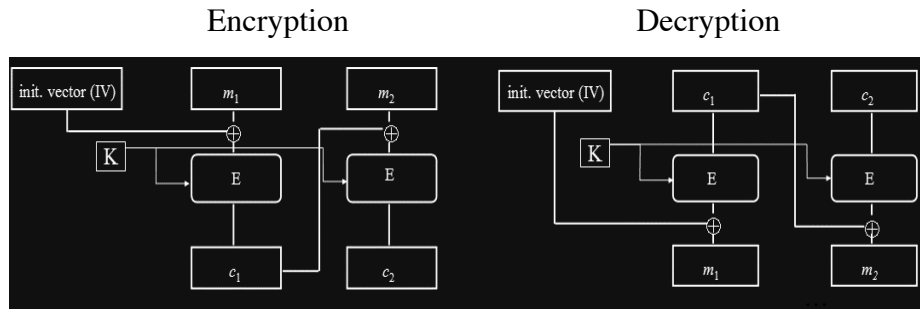
Alice in CBC Mode



30



Decryption with CBC



Encryption

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

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

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

Decryption

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

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

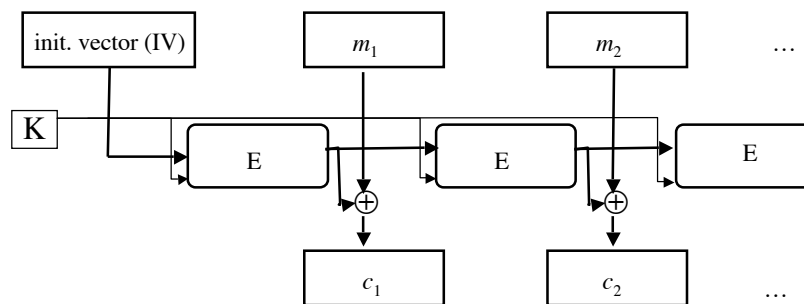
$$P_2 = C_1 \oplus D(C_2, K), \dots$$

31



Output Feedback (OFB) Mode

- $K_1 = \{IV\}_K, K_2 = \{K_1\}_K, \dots, K_i = \{K_{i-1}\}_K \dots$
- Purpose: use block cipher as a stream cipher
- $C_i = m_i \oplus \{K_{i-1}\}_K, K_0 = IV$

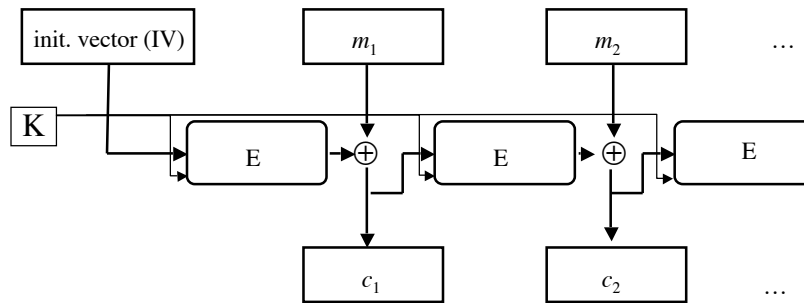


32



Cipher Feedback (CFB) Mode

$$M = m_1 | m_2 | \dots | m_n$$

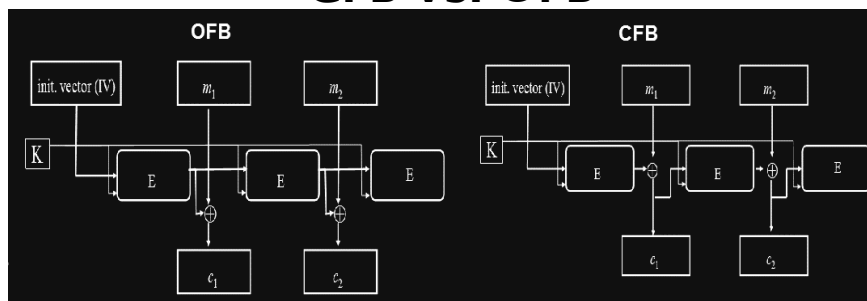


$$C = IV | c_1 | c_2 | \dots | c_n$$

33



CFB vs. OFB



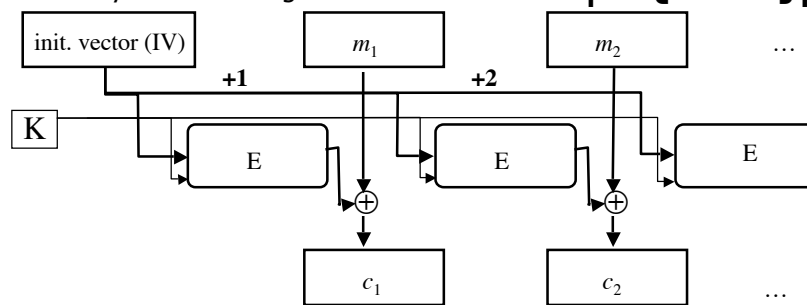
- Recovery from:
 - Erroneous Bit:
 - OFB – only that bit lost, CFB – that bit + next block lost
 - Lost Block
 - OFB – no need to retransmit, CFB – must retransmit
- Message Integrity
 - OFB – no protection, CFB – more protection, but still not secure

34



Counter Encryption

- Drawbacks of feedback modes
 - Hard to parallelize
 - CBC -- cannot precompute
 - OFB -- memory requirements
- Counter Encryption is easier to parallelize
 - Key for i th block generated from IV + i : $K_i = \{IV + i\}_K$

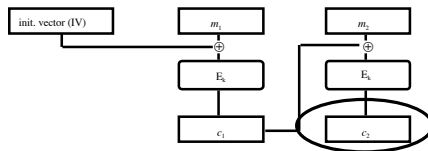


35



Message Authentication Code (MAC)

- Purpose
 - protect message integrity and authenticity
- How to do MAC with a block cipher?



In CBC mode, the last block of cipher text serves as the MAC for the entire message

- How to do MAC and encryption of a message?

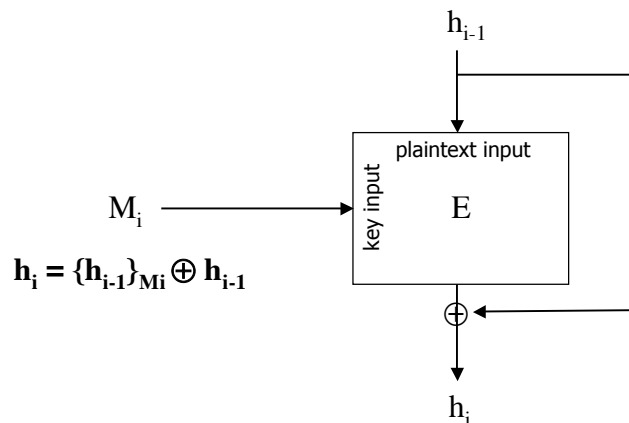
36



Hash Function from a Block Cipher

$$h = H(M)$$

1. Easy to compute h from M - efficient
2. Hard to compute M from h - one way
3. For **given** M , hard to find another M' s.t. $H(M) == H(M')$ - weak collision resistance
4. For **any** M & M' s.t. $H(M) == H(M')$ - strong collision resistance



37



Common Hash Functions and Applications

- Common hash functions
 - (Message Digest) MD5 value 128b
 - (Secure Hash Algorithm) SHA-1 180b value, SHA-256, SHA-512
- Applications
 - MACs
 - $MAC_K(M) = H(K, M)$
 - $HMAC_K(M) = H(K \oplus A, H(K \oplus B, M))$, A & $B = \text{magic}$ (pg. 94, Stamp)
 - Time stamping service
 - key updating
 - $K_i = H(K_{i-1})$
 - Backward security
 - Autokeying
 - $K_{i+1} = H(K_i, M_{i1}, M_{i2}, \dots)$
 - Forward security

38



Key Points

- Ciphers are either substitution, transposition (a.k.a., permutation), or product
- Any block cipher should confuse and defuse
- Block ciphers are implemented in SP-networks
- Stream ciphers and hash functions are commonly implemented with block ciphers
- Hash functions used for
 - fingerprinting data, MAC, key updating, autokeying,

