Stream Ciphers

Stream Ciphers

Unconditionally Secure Cipher

Regardless of known ciphertext length, there is not enough information in the ciphertext to determine the plaintext uniquely.

Stream Ciphers

Unconditionally Secure Cipher

Regardless of known ciphertext length, there is not enough information in the ciphertext to determine the plaintext uniquely.

Entropy of a Symbol (Shannon – 1949)

Given alphabet $S=\{s_1, s_2, \ldots , s_n\}$ with probs $\{p_1, p_2, \ldots , p_n\}$ of occuring in message M of length m. Define entropy

$H(S) = -\sum_i p_i \log_2(p_i)$ for all non-zero p_i

Stream Ciphers

Unconditionally Secure Cipher

Regardless of known ciphertext length, there is not enough information in the ciphertext to determine the plaintext uniquely.

Entropy of a Symbol (Shannon – 1949)

Given alphabet $S=\{s_1, s_2, \ldots, s_n\}$ with probs $\{p_1, p_2, \ldots, p_n\}$ of occuring in message M of length m. Define entropy

$H(S) = -\sum_i p_i \log_2(p_i)$ for all non-zero p_i

Observe: if all p_i are equal, $H(S) = \log_2(n)$

if $p_1=1$, all other $p_i=0$, $H(S) = 0$.

if $p_1= ½$ and $p_2= ½$, other $p_i=0$, $H(S)=1$.

Stream Ciphers

Unconditionally Secure Cipher

Regardless of known ciphertext length, there is not enough information in the ciphertext to determine the plaintext uniquely.

Entropy of a Symbol (Shannon – 1949)

Given alphabet $S=\{s_1, s_2, \ldots, s_n\}$ with probs $\{p_1, p_2, \ldots, p_n\}$ of occuring in message M of length m. Define entropy

$H(S) = -\sum_i p_i \log_2(p_i)$ for all non-zero p_i

Observe: if all p_i are equal, $H(S) = \log_2(n)$

if $p_1=1$, all other $p_i=0$, $H(S) = 0$.

if $p_1= ½$ and $p_2= ½$, other $p_i=0$, $H(S)=1$.

For random bits, Pr (guess the next bit) = ½.

Entropy measures the minimum number of bits needed to encode a sequence of symbols.

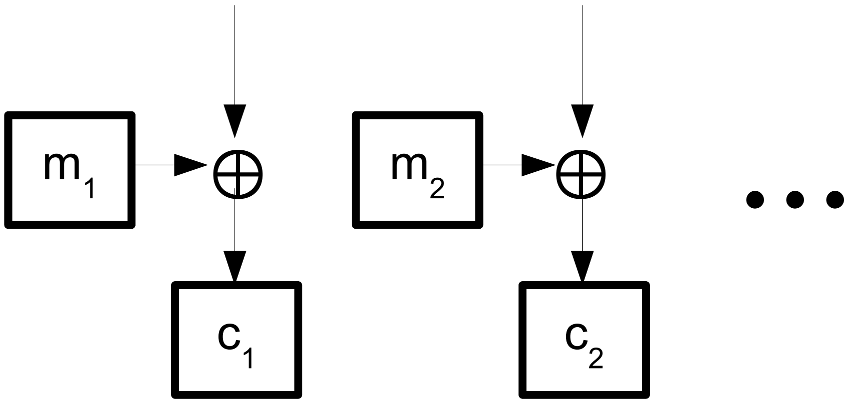
½ bit of entropy if p1 = 0.11002786.., p2 = 0.88997213...

Stream Ciphers

Consider one-time pad (Vernam – 1917)

Let p_1 be the probability that 1 is the next message bit m_i and p_0 is the probability that 0 is the next m_i.

Random sequence of 0s and 1s



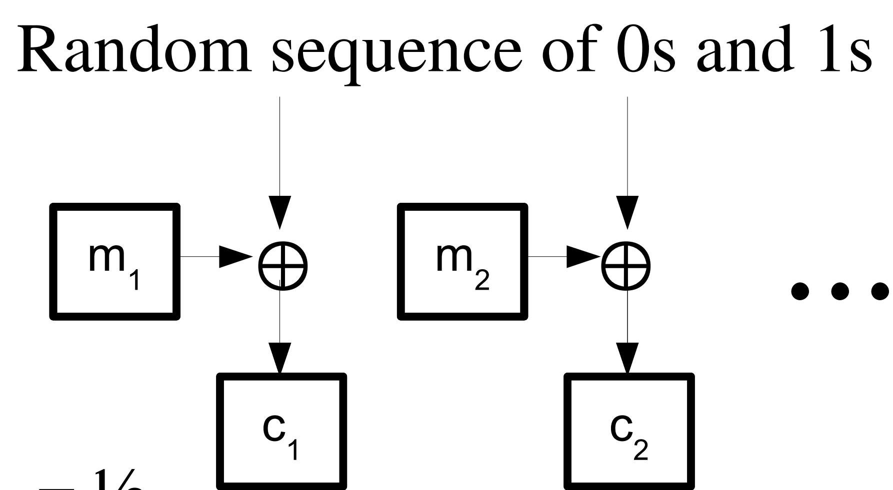
Then $\Pr(m_1 \oplus r_1 \text{ is } 0) = (½)p_1 + (½)p_0 = ½$

$\Pr(m_1 \oplus r_1 \text{ is } 1) = (½)p_0 + (½)p_1 = ½$

$H(c_i) = 1$

Stream Ciphers

Consider one-time pad (Vernam – 1917)



Let p_1 be the probability that 1 is the next message bit m_i and p_0 is the probability that 0 is the next m_i.

Then

$$Pr\ (m_1 \oplus r_1 \text{ is } 0) = (½)p_1 + (½)p_0 = ½$$

$$Pr\ (m_1 \oplus r_1 \text{ is } 1) = (½)p_0 + (½)p_1 = ½$$

$H(c_i) = 1$

Unconditionally secure: $H(M \mid C) = H(M)$

Given a ciphertext c_i, the probability that it was the encryption of some plaintext m_i is equal to the probability that it was the encryption of another plaintext m

This motivates consideration of Stream Ciphers for real apps.

Stream Ciphers

Important:

A necessary condition for a symmetric key encryption scheme to be unconditionally secure is $H(K) \geq H(M)$. (K is the key)

So, the uncertainty of the secret key must be at least as great as the uncertainty of the plaintext.

If the key is random and its length is k then $H(K) = k$.
In that case, we need $k \geq H(M)$.

But: then the key must be pretty long – which is impractical.
In practice some means of generating a pseudo-random sequence Q of bits is used. Unfortunately, it is generally true that $H(Q) \ll H(M)$.

Stream Ciphers

Block Ciphers:

Operates on fixed-length groups of bits called blocks.
Same operation for each block controlled by a secret key.
Encryption and decryption use “symmetric” algorithms.

Stream Ciphers

Block Ciphers:

Operates on fixed-length groups of bits called blocks.

Same operation for each block controlled by a secret key.

Encryption and decryption use “symmetric” algorithms.

Examples:

AES, Akelarre, Anubis, ARIA, BaseKing, Blowfish, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES, 3DES, DES-X, DFC, E2, FEAL, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Iraqi, Intel Cascade Cipher, Karn, KASUMI, KHAZAD, Khufu and Khafre, KN-Cipher, Libelle, LOKI89/91, LOKI97, Lucifer, M6, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, NewDES, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SC2000, SEED, Serpent, SHACAL, SHARK, Skipjack, SMS4, Square, TEA, Treyfer, Twofish, UES, Xenon, xmx, XTEA, XXTEA, Zodiac

Stream Ciphers

Block Ciphers:

Operates on fixed-length groups of bits called blocks.

Same operation for each block controlled by a secret key.

Encryption and decryption use “symmetric” algorithms.

Examples:

AES, Akelarre, Anubis, ARIA, BaseKing, Blowfish, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES, 3DES, DES-X, DFC, E2, FEAL, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Iraqi, Intel Cascade Cipher, Karn, KASUMI, KHAZAD, Khufu and Khafre, KN-Cipher, Libelle, LOKI89/91, LOKI97, Lucifer, M6, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, NewDES, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SC2000, SEED, Serpent, SHACAL, SHARK, Skipjack, SMS4, Square, TEA, Treyfer, Twofish, UES, Xenon, xmx, XTEA, XXTEA, Zodiac

Stream Ciphers:

Operates on individual digits (bits), one at a time.

Operation varies during the encryption.

Stream Ciphers

Block Ciphers:

Operates on fixed-length groups of bits called blocks.

Same operation for each block controlled by a secret key.

Encryption and decryption use “symmetric” algorithms.

Examples:

AES, Akelarre, Anubis, ARIA, BaseKing, Blowfish, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES, 3DES, DES-X, DFC, E2, FEAL, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Iraqi, Intel Cascade Cipher, Karn, KASUMI, KHAZAD, Khufu and Khafre, KN-Cipher, Libelle, LOKI89/91, LOKI97, Lucifer, M6, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, NewDES, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SC2000, SEED, Serpent, SHACAL, SHARK, Skipjack, SMS4, Square, TEA, Treyfer, Twofish, UES, Xenon, xmx, XTEA, XXTEA, Zodiac

Stream Ciphers:

Operates on individual digits (bits), one at a time.

Operation varies during the encryption.

Examples:

A5/1, A5/2, E0. FISH, Grain, HC-256, ISAAC, LILI-128, MUGI, Panama, Phelix, Pike, Py, Rabbit, RC4, Salsa20, Scream, SEAL, SOBER, SOBER-128, SOSEMANUK, Trivium, VEST, WAKE

Stream Ciphers

Usage: where plaintext comes in quantities of unknowable length.

Stream Ciphers

Usage: where plaintext comes in quantities of unknowable length.

Example: a secure wireless connection – cannot wait for a full block to be assembled before encrypting – either there is a delay until block is received or heavily padded blocks are output.

Stream Ciphers

Usage: where plaintext comes in quantities of unknowable length.

Example: a secure wireless connection – cannot wait for a full block to be assembled before encrypting – either there is a delay until block is received or heavily padded blocks are output.

Military applications: cipher stream can be generated in a separate box subject to strict security measures and fed to other devices which will perform the XOR operation as part of their function. The latter device can then be designed and used in less stringent environments.

Stream Ciphers

Synchronous stream cipher:

$$\begin{aligned}\sigma_{i+1} &= f(\sigma_i, k),\\ z_i &= g(\sigma_i, k),\\ c_i &= h(z_i, m_i),\end{aligned}$$

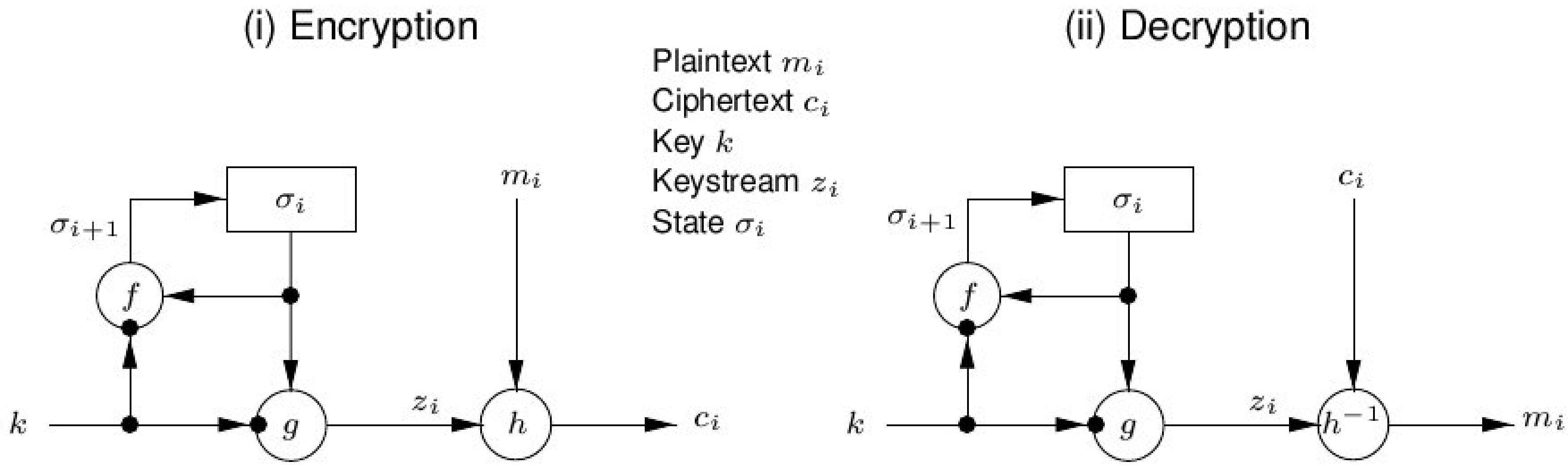
k is the key

σ_0 is the initial state, determined from the key

f is the next-state function

g is the function that produces the keystream

h is the output function



Memory vs. memoryless in the case of block ciphers

Stream Ciphers

Synchronous stream cipher:

A stream of "random" bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

Stream Ciphers

Synchronous stream cipher:

A stream of “random” bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

xor operation is the scrambler (binary additive stream cipher).

Stream Ciphers

Synchronous stream cipher:

A stream of "random" bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

xor operation is the scrambler (binary additive stream cipher).

Sender and receiver must be exactly in step – added or lost bits screw things up for ever after.

Stream Ciphers

Synchronous stream cipher:

A stream of "random" bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

xor operation is the scrambler (binary additive stream cipher).

Sender and receiver must be exactly in step – added or lost bits screw things up for ever after.

However synchronization may be restored by trying various offsets. Also the ciphertext may be "tagged" at regular points in the output.

Stream Ciphers

Synchronous stream cipher:

A stream of “random” bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

xor operation is the scrambler (binary additive stream cipher).

Sender and receiver must be exactly in step – added or lost bits screw things up for ever after.

However synchronization may be restored by trying various offsets. Also the ciphertext may be “tagged” at regular points in the output.

If bit is only corrupted in transmission the error does not propagate. This is great when the transmission error rate is high (if ever).

Stream Ciphers

Synchronous stream cipher:

A stream of "random" bits generated independently of the plaintext and ciphertext and combined with plaintext or the ciphertext to encrypt or decrypt.

xor operation is the scrambler (binary additive stream cipher).

Sender and receiver must be exactly in step – added or lost bits screw things up for ever after.

However synchronization may be restored by trying various offsets. Also the ciphertext may be "tagged" at regular points in the output.

If bit is only corrupted in transmission the error does not propagate. This is great when the transmission error rate is high (if ever).

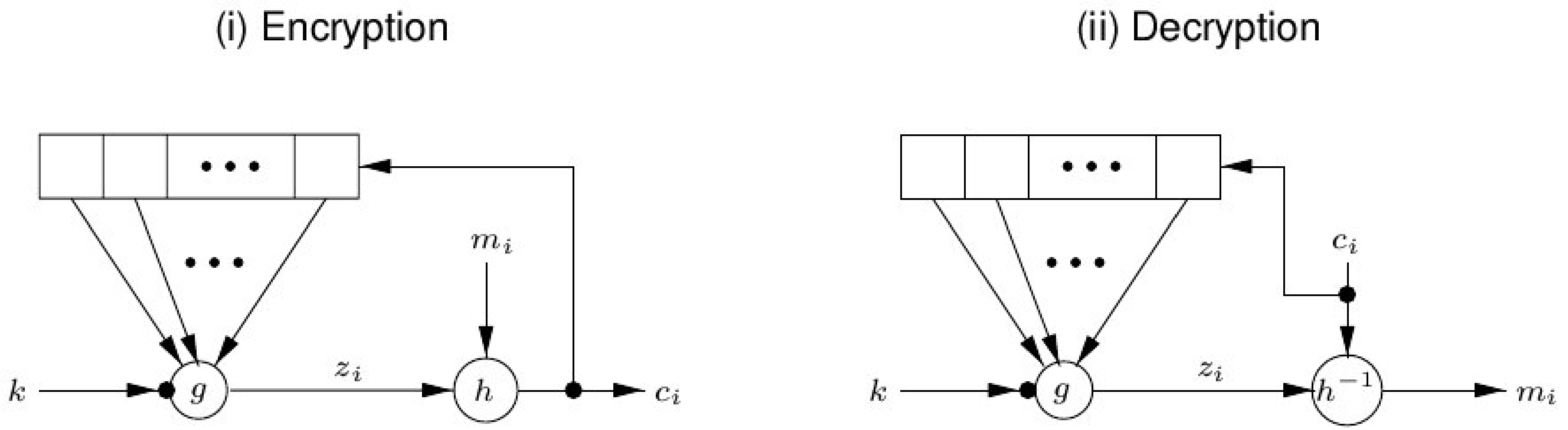
Susceptible to active attacks — if an attacker can change a bit in the ciphertext, it might be able to make predictable changes to the corresponding plaintext bit: flip a ciphertext bit to flip a plaintext bit.

Stream Ciphers

Self-synchronizing stream ciphers: uses several of the previous N ciphertext bits to compute a keystream.

$$\begin{aligned}\sigma_i &= (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}),\\ z_i &= g(\sigma_i, k),\\ c_i &= h(z_i, m_i),\end{aligned}$$

$$\sigma_0 = (c_{-t}, c_{-t+1}, \ldots, c_{-1})$$



Stream Ciphers

Self-synchronizing stream ciphers: uses several of the previous N ciphertext bits to compute a keystream.

The receiver will automatically synchronize with the keystream generator after receiving N ciphertext bits – hence recovery if bits are dropped or added.

Stream Ciphers

Self-synchronizing stream ciphers: uses several of the previous N ciphertext bits to compute a keystream.

The receiver will automatically synchronize with the keystream generator after receiving N ciphertext bits – hence recovery if bits are dropped or added.

It is somewhat more difficult to perform active attacks on self-synchronizing stream ciphers by comparison with their synchronous counterparts – modifying one cipher bit may affect several keystream bits.

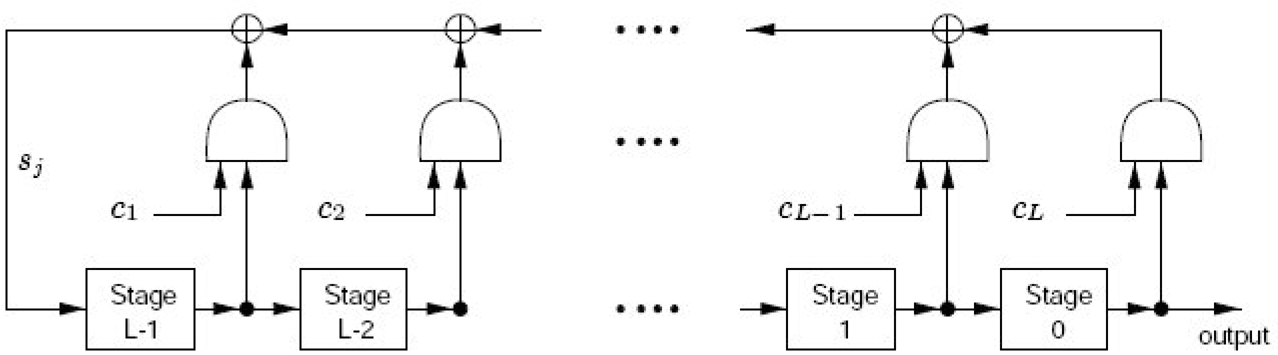
Stream Ciphers

Feedback Shift Register:

The basic component of many keystream generators.

Stream Ciphers

Linear Feedback Shift Register (LFSR)

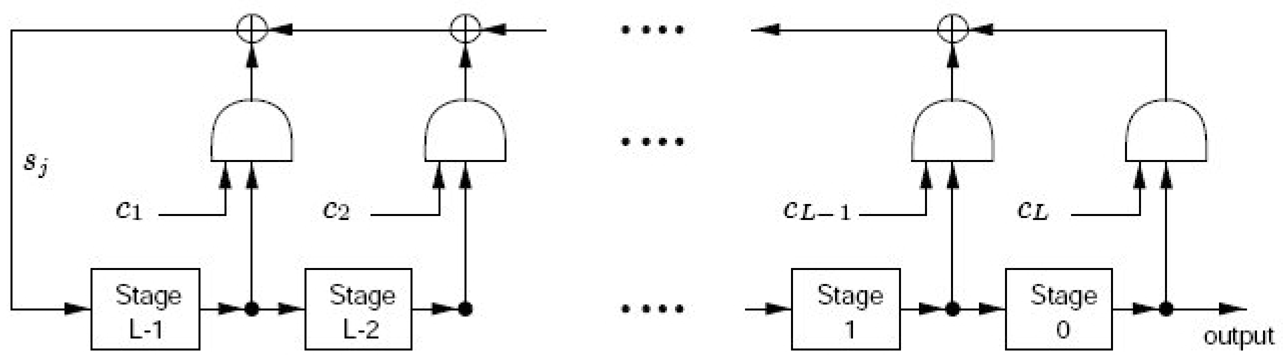


A **LFSR** of length L consists of L delay elements numbered 0, 1, ..., L–1, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data.

c_1, ... , c_L is, in this case, the non-secret initial state, obtained via runup.

Stream Ciphers

Linear Feedback Shift Register (LFSR)



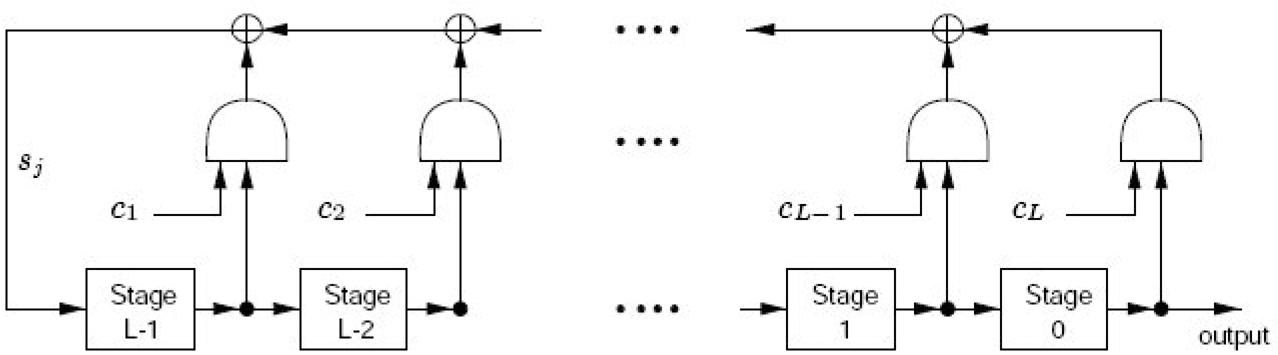
Properties:

They are well-suited for hardware implementations
They can produce sequences of long periods
They can produce sequences with good statistical properties
They can be readily analyzed using algebraic techniques

more later...

Stream Ciphers

Linear Feedback Shift Register (LFSR)



Operation:

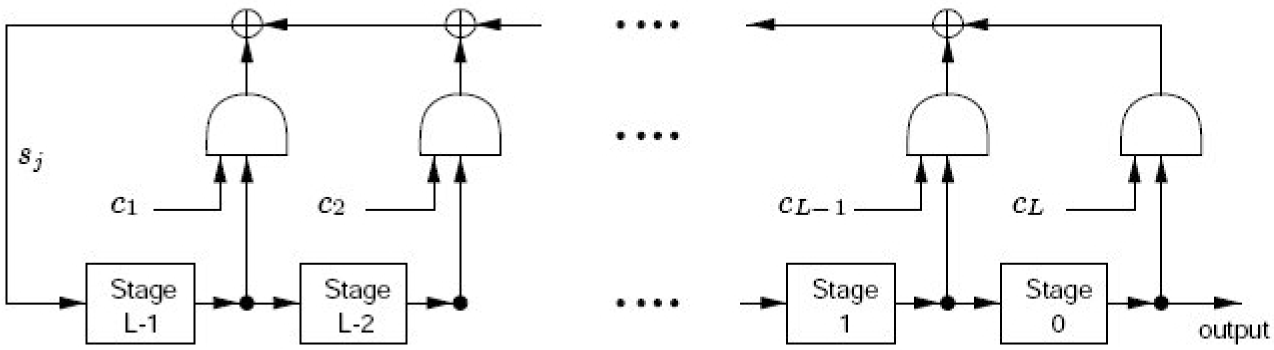
At each time step the following can happen:

(i) element 0 is output and forms part of the output sequence.

(ii) element *i* is moved to element *i-1*

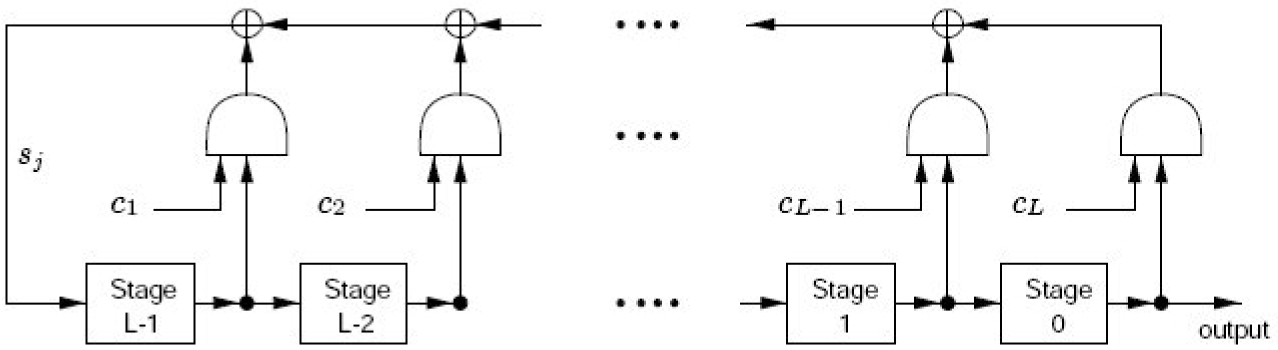
(iii) element L-1 determined from mod 2 addition of selected elements

Stream Ciphers



$s_j = D_j = (c_1 D_{j-1} \oplus c_2 D_{j-2} \oplus \ldots \oplus c_L D_{j-L})$ for $j > L$, j^{th} output bit

Stream Ciphers



$s_j = D_j = (c_1 D_{j-1} \oplus c_2 D_{j-2} \oplus \ldots \oplus c_L D_{j-L})$ for $j > \mathrm{L}$, j^{th} output bit

Example: L = 4, $c_1 = c_4 = 1$, $c_2 = c_3 = 0$, $D_0 = D_3 = 0$, $D_1 = D_2 = 1$

t	D_3	D_2	D_1	D_0
0	0	1	1	0
1	0	0	1	1
2	1	0	0	1
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0

t	D_3	D_2	D_1	D_0
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	0	1	1	0

Output sequence: 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1

Stream Ciphers

Connection Polynomial:

$$C(D) = 1 + c_1 D + c_2 D^2 + c_3 D^3 + \ldots + c_L D^L$$

An LFSR is said to generate a sequence s if there is some initial state for which the output sequence of the LFSR is s.

Similarly, an LFSR is said to generate a finite sequence s^N if there is some initial state for which the output sequence of the LFSR has s^N as its first N terms.

Linear complexity of s - $L(s)$:

1. if $s = 0,0,\ldots$ then $L(s) = 0$
2. if no LFSR generates s then $L(s) = \infty$
3. $L(s)$ is the length of the shortest LFSR that generates s

Stream Ciphers

Complexity Profile of s

Linear complexity of a subsequence s^N of s is denoted L_N

The sequence $L_1, L_2, \ldots L_N$ is called the linear complexity profile of s.

If $j > i$ then $L_j \geq L_i$.

$L_{N+1} > L_N$ is possible only if $L_N \leq N/2$.

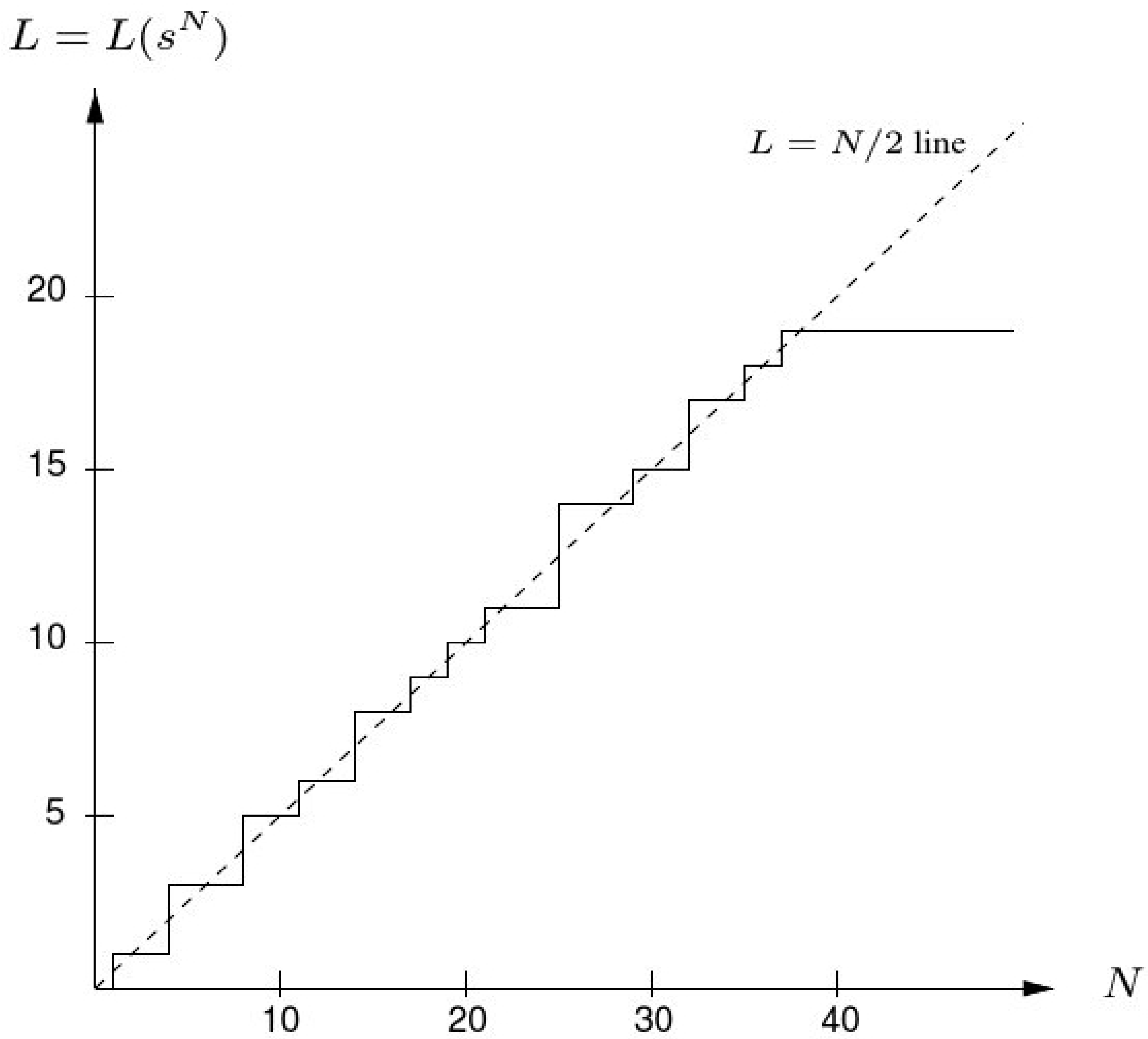
If $L_{N+1} > L_N$ then $L_{N+1} + L_N = N+1$

Example:

s = 1,0,0,1,0,0,1,1,1,1,0,0,0,1,0,0,1,1,1,0,.... (periodic)

LCP = 1,1,1,3,3,3,3,5,5,5,6,6,6,8,8,8,9,9,10,10,11,11,11,11,
14,14,14,14,15,15,15,17,17,17,18,18,19,19,19,19,...

Stream Ciphers



Stream Ciphers

Complexity Profile of s

next discrepancy d_N – difference between s^N and the N+1st term generated by the LFSR: $d_N = (s^N \oplus \sum_{1 \le i \le L} c_i s^{N-i}) \bmod 2$

The LFSR that generates s^N also generates s^{N+1} if and only if the next discrepancy $d_N = 0$.

If $d_N = 0$ then $L(s^{N+1}) = L(s^N)$

If $d_N = 1$, suppose m is greatest such that $L(s^m) < L(s^N)$, let C(D), B(D) be the connection polynomial for s^N, s^m then $C(D) + B(D) \cdot D^{N-m}$ is the connection polynomial for smallest LFSR that generates s^{N+1}

Stream Ciphers

Berlekamp-Massey Algorithm:

INPUT: a binary sequence $s^n = s_0; s_1; s_2; : : : ; s_{n-1}$ of length n.

OUTPUT: the linear complexity $L(s^n)$ of s^n, $0 \leq L(s^n) \leq n$.

1. Initialization. $C(D) \leftarrow 1$, $L \leftarrow 0$, $m \leftarrow -1$, $B(D) \leftarrow 1$, $\mathrm{N} \leftarrow 0$.
2. While ($\mathrm{N} < n$) do the following:

2.1 Compute the next discrepancy d: $d \leftarrow (s_{\mathrm{N}} + \sum c_i s_{\mathrm{N}-i}) \bmod 2$.

2.2 If $d == 1$ then do the following:

$$T(D) \leftarrow C(D),\ C(D) \leftarrow C(D) + B(D) \cdot D^{\mathrm{N}-m}.$$

If $L \leq \mathrm{N}/2$ then $L \leftarrow \mathrm{N} + 1 - L$, $m \leftarrow \mathrm{N}$, $B(D) \leftarrow \mathrm{T}(D)$.

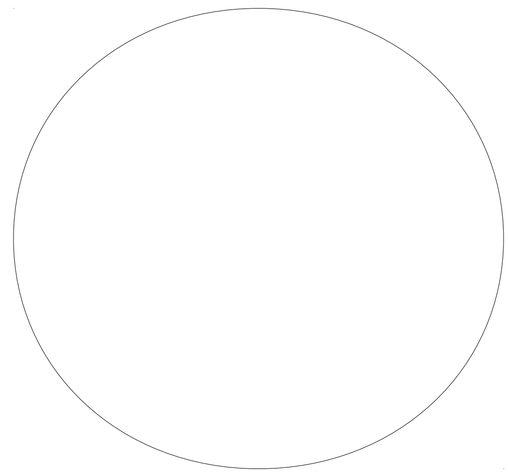
2.3 $\mathrm{N} \leftarrow \mathrm{N} + 1$.

3. Return(L).

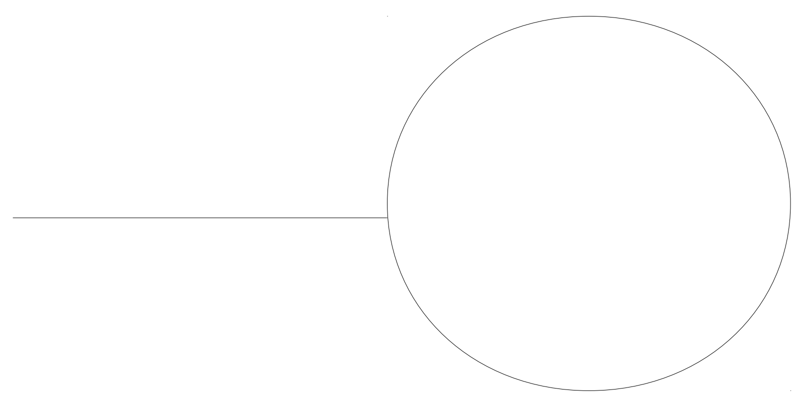
At the end of each iteration of step 2, $C(D)$ is a non-singular LFSR of smallest length which generates s^{N}.

Stream Ciphers

Non-singular



Singular



Stream Ciphers

Properties of Linear Feedback Shift Registers:

1. they are well-suited to hardware implementation
2. they can produce sequences of large period
3. they can produce sequences with good statistical properties
   - the distribution of patterns having fixed length of at most L is almost uniform for certain c_i.
4. they can be readily analyzed using algebraic techniques
   - the average length of the shortest LFSR that generates a sequence having a random string of n bits as output is about $n/2$ and its variance is about 1.

Stream Ciphers

Unfortunately, the output sequences of LFSRs are also easily predictable!

Stream Ciphers

Unfortunately, the output sequences of LFSRs are also easily predictable!

The connection polynomial $C(D)$ of an LFSR of length L which generates string s can be efficiently determined using the Berlekamp-Massey algorithm from any (short) subsequence t of s having length at least $n = 2L$. Having determined $C(D)$, the LFSR can then be initialized with any substring of t having length L, and used to generate s (beginning at t).

Stream Ciphers

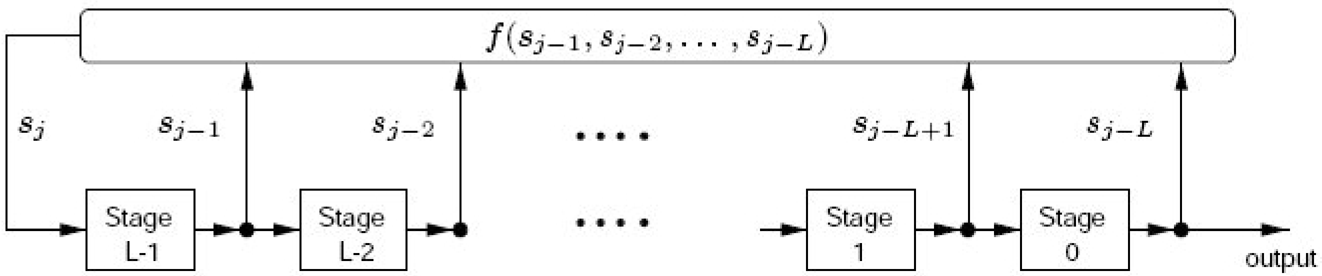
Unfortunately, the output sequences of LFSRs are also easily predictable!

The connection polynomial $C(D)$ of an LFSR of length L which generates string s can be efficiently determined using the Berlekamp-Massey algorithm from any (short) subsequence t of s having length at least $n = 2L$. Having determined $C(D)$, the LFSR can then be initialized with any substring of t having length L, and used to generate s (beginning at t).

An adversary may obtain the required subsequence t of s by mounting a known or chosen-plaintext attack on the stream cipher. If the adversary knows the plaintext subsequence $m_1, m_2, \ldots, m_n$ corresponding to a ciphertext sequence $c_1, c_2, \ldots, c_n$, the corresponding keystream bits are obtained as $m_i \oplus c_i$. The keystream bits are t.

Stream Ciphers

Non-Linear Feedback Shift Register (FSR)



An FSR is non-singular iff $f = s_{j-L} \oplus g(s_{j-1},\ldots,s_{j-L})$ for some boolean function g.

The period of a length L non-singular FSR could be 2^L

In that case the output sequence is a DeBruijn sequence

Example:

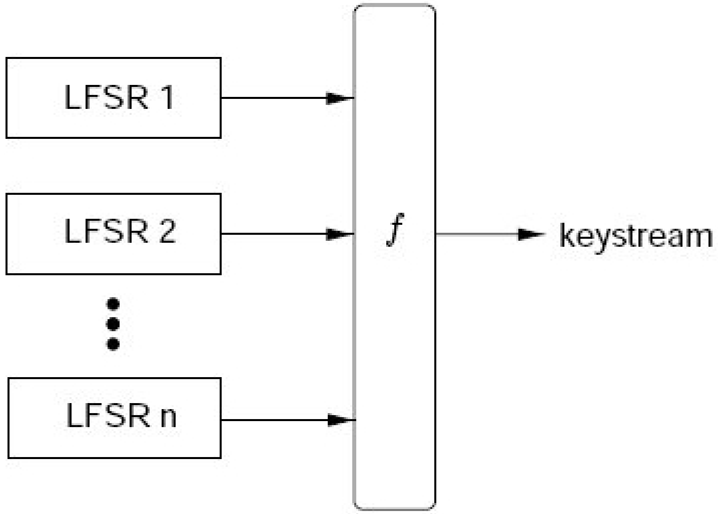
00010111 →

000, 001, 010, 101, 011, 111, 110, 100

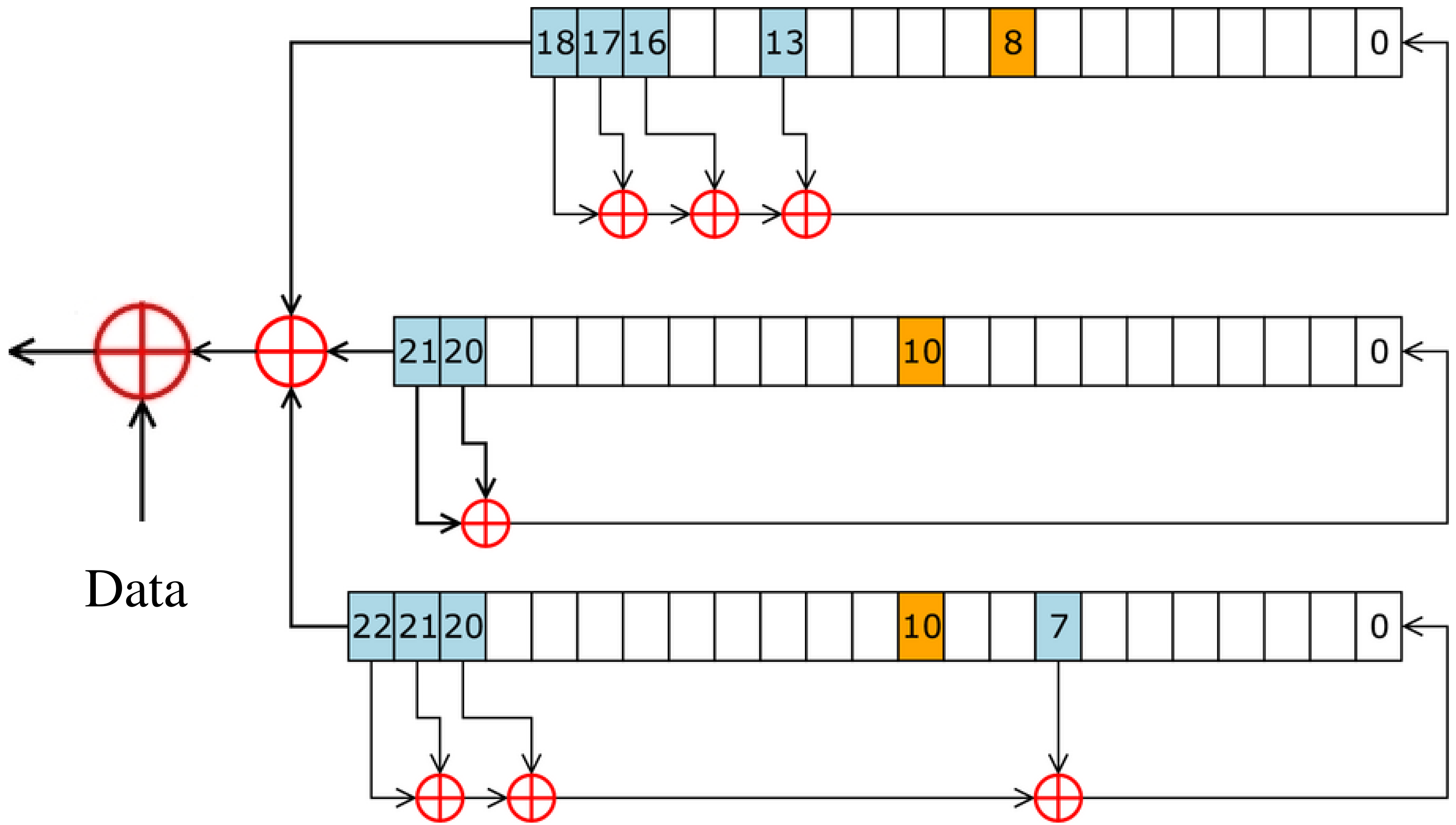
Stream Ciphers

Methods for mitigating the predictability:

1. Use a non-linear combining function on the outputs of several LFSRs.



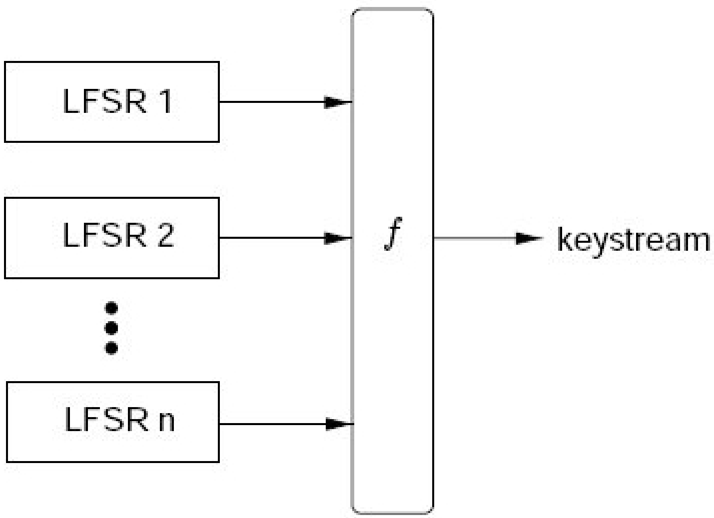
Stream Ciphers

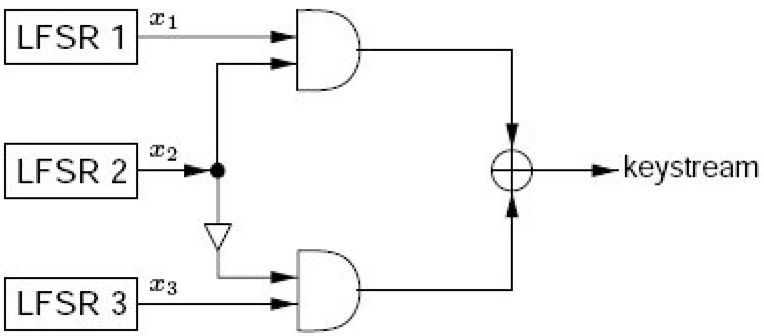


Stream Ciphers

Methods for mitigating the predictability:

1. Use a non-linear combining function on the outputs of several LFSRs.





Period: $(2^{L1}-1)(2^{L2}-1)(2^{L3}-1)$

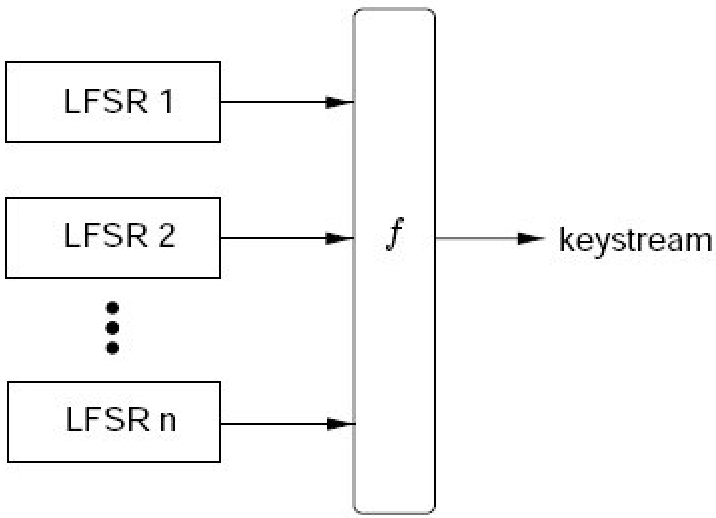
LinComplexity: L1*L2 + L2*L3 + L3

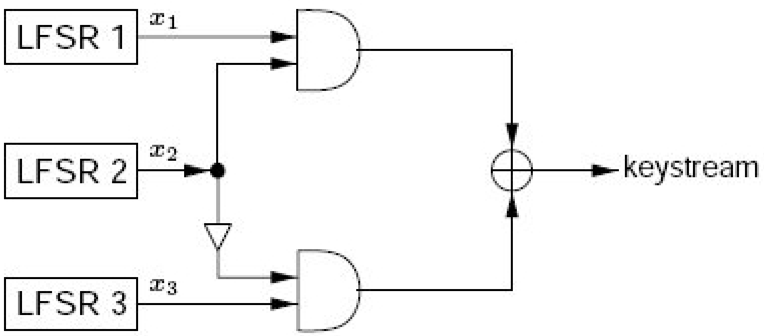
Geffe Generator

Stream Ciphers

Methods for mitigating the predictability:

1. Use a non-linear combining function on the outputs of several LFSRs.





Correlation attack!!

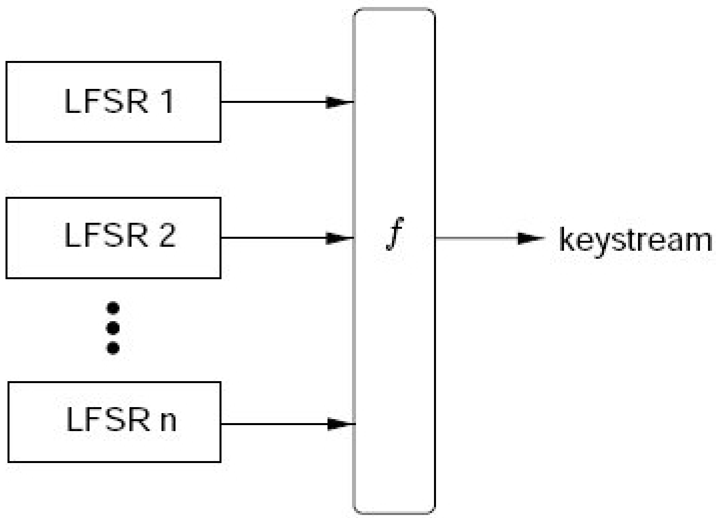
correlation between output sequence of x_1 and the keystream: $\Pr(x_1(t) = k(t)) = 0.75$

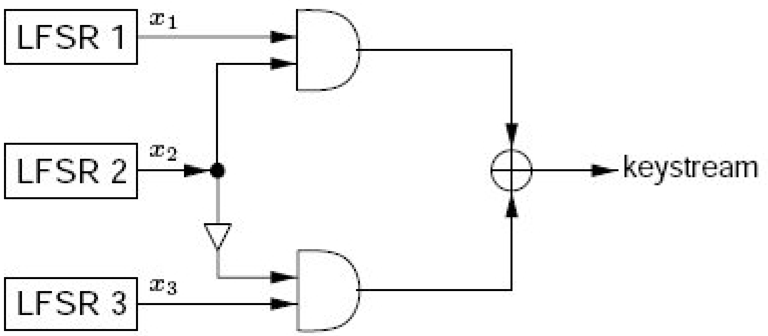
If sufficiently long segment of keystream is known, the initial state of LFSR1 can be deduced by counting the number of coincidences between the keystream and all possible shifts of the output sequence of LFSR1 until this number agrees with the correlation probability.

Stream Ciphers

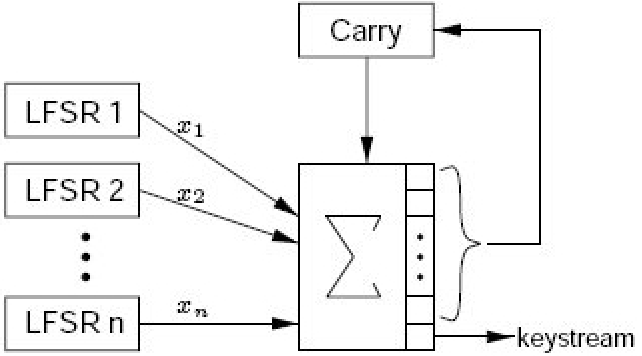
Methods for mitigating the predictability:

1. Use a non-linear combining function on the outputs of several LFSRs.





Try to eliminate or reduce correlation

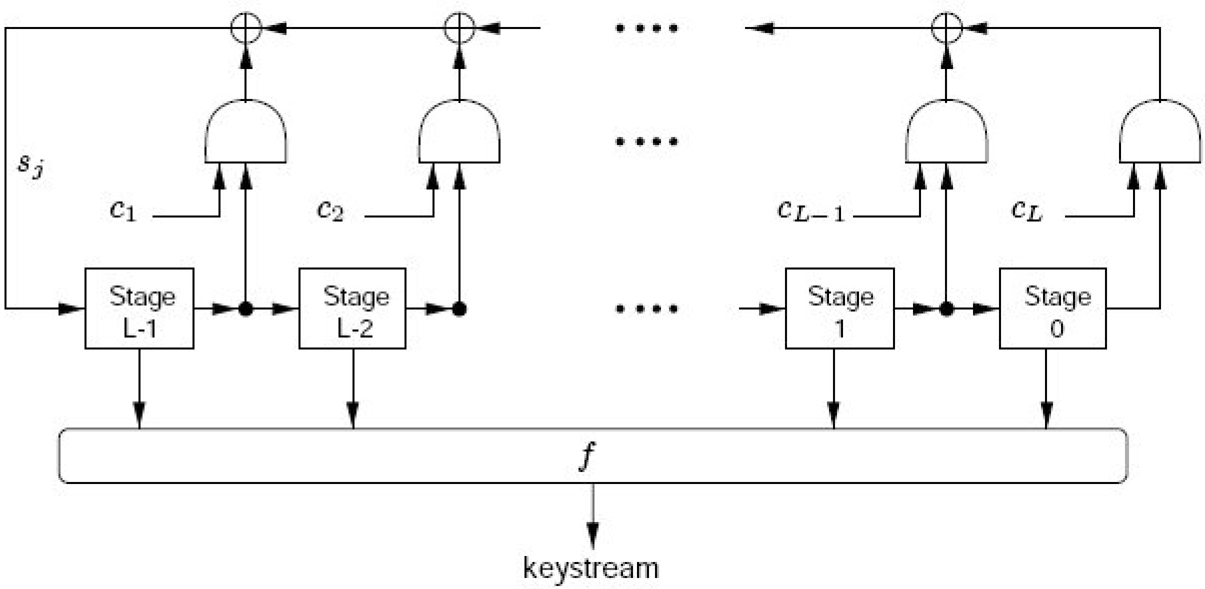


Period: $\Pi(2^{Li}-1)$ **LinComp**: nearly the same

Stream Ciphers

Methods for mitigating the predictability:

2. Use a non-linear filtering function on the contents of a single LFSR.

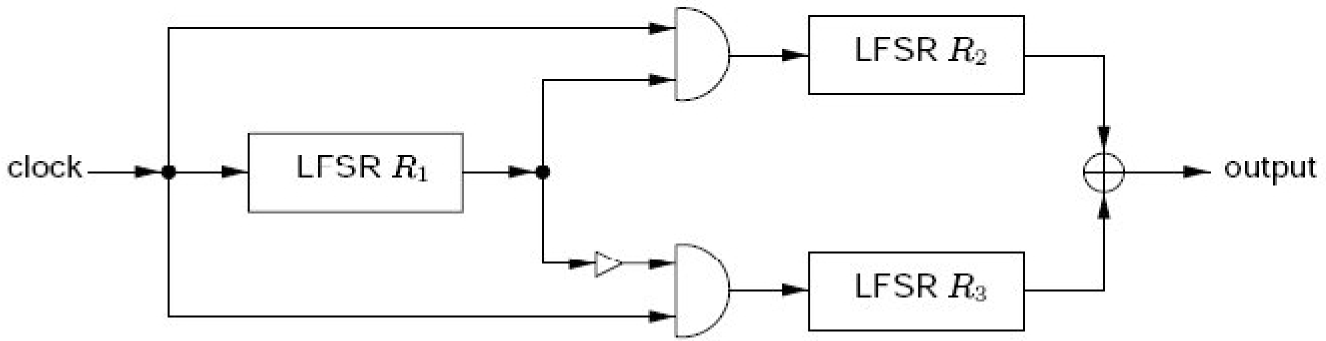


Function f represents some NP-complete problem for example knapsack - the key is a bunch of weights, f is the sum of those weights corresponding to the 1 bits of s_j Trying to find the bits of that sum is same as solving knapsack.

Stream Ciphers

Methods for mitigating the predictability:

3. Use the output of one or more LFSRs to control the clock of one or more other LFSRs.



Lengths of LFSRs should be relatively prime.
Divide and conquer attack on R1 is best known – exponentially many steps are required though

Period: $2^{L1}(2^{L2}-1)(2^{L3}-1)$ **LinComp**: at least $2^{L1-1}(L2+L3)$

Stream Ciphers

Register R1 is clocked

If the output bit of R1 is 1 do this:

- R2 is clocked; R3 is not clocked but its previous output bit is repeated

If the output bit of R1 is 0 do this:

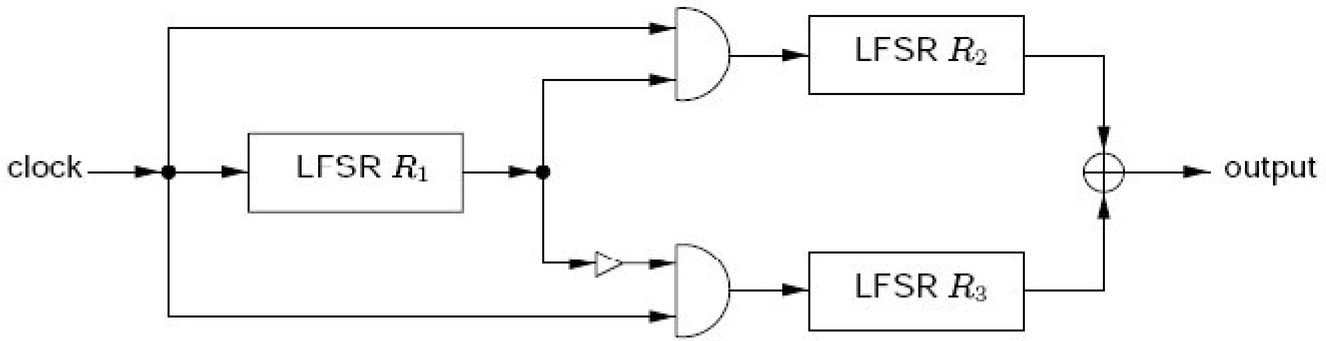
- R3 is clocked; R2 is not clocked but its previous output bit is repeated

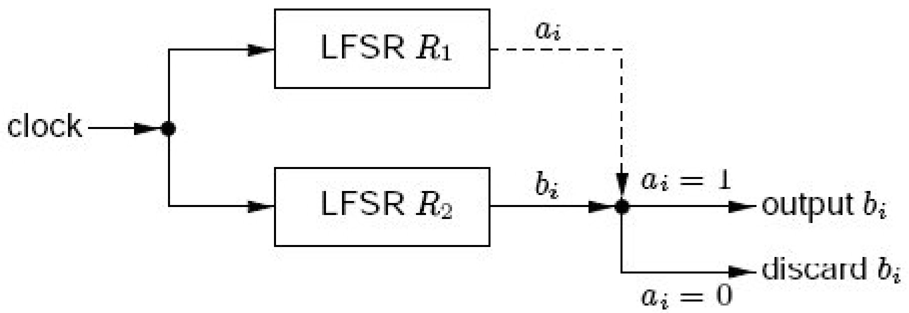
Output bits of R2 and R3 are xored to form the keystream

Stream Ciphers

Methods for mitigating the predictability:

3. Use the output of one or more LFSRs to control the clock of one or more other LFSRs.





R1 is used to select a portion of R2

Stream Ciphers

Registers R1 and R2 are clocked
If the output bit of R1 is 1 do this:
    output a bit from R2
If the output bit of R1 is 0 do this:
    discard the next bit from R2

Stream Ciphers

Known vs. Secret Connection Polynomials:

1. Known – secret key is the initial state of the LFSR
2. Secret – secret key is used both to initialize the LFSR and to provide the c_i's
3. Secret – Provides more security.
4. Known – simpler hardware implementation.

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems
4. But is extraordinarily simple – requires 8-16 machine operations per output byte

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems
4. But is extraordinarily simple – requires 8-16 machine operations per output byte
5. Has a period greater than 10^{100}

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems
4. But is extraordinarily simple – requires 8-16 machine operations per output byte
5. Has a period greater than 10^{100}
6. Does not use LFSRs and works well in software or hardware since it requires only byte operations, not bit operations.

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems
4. But is extraordinarily simple – requires 8-16 machine operations per output byte
5. Has a period greater than 10^{100}
6. Does not use LFSRs and works well in software or hardware since it requires only byte operations, not bit operations.
7. The state machine uses only 256 bytes of memory

Stream Ciphers

RC4:

1. One of the most widely used stream ciphers
2. Shows up in WEP, WPA, SSL, BitTorrent, MS P-P, ...
3. Has many problems – not considered secure – not used in new cryptosystems
4. But is extraordinarily simple – requires 8-16 machine operations per output byte
5. Has a period greater than 10^{100}
6. Does not use LFSRs and works well in software or hardware since it requires only byte operations, not bit operations.
7. The state machine uses only 256 bytes of memory
8. Modulo 256 operations – can be done with bitwise AND

Stream Ciphers

RC4 - Machinery:

1. 256 byte = 2048 bit state machine for over 10^{500} states
2. Two 8 bit index pointers labeled *i* and *j* – these contain indices into the byte array of the state machine.

Stream Ciphers

RC4 - Machinery:

1. 256 byte = 2048 bit state machine for over 10^{500} states
2. Two 8 bit index pointers labeled *i* and *j* – these contain indices into the byte array of the state machine.

Algorithms:

1. Key scheduling algorithm (KSA) which is used to initialize the state of the state machine – variable size key - may be 40 to 256 bytes in length. Steps:
   a. A common starting state is set – 1 to 255 in each byte
   b. Machine is run for 256 clock cycles, mixing in the key no initialization vector!

Stream Ciphers

RC4 - Machinery:

1. 256 byte = 2048 bit state machine for over 10^{500} states
2. Two 8 bit index pointers labeled *i* and *j* – these contain indices into the byte array of the state machine.

Algorithms:

1. Key scheduling algorithm (KSA) which is used to initialize the state of the state machine – variable size key - may be 40 to 256 bytes in length. Steps:
   a. A common starting state is set – 1 to 255 in each byte
   b. Machine is run for 256 clock cycles, mixing in the key no initialization vector!
2. Pseudo Random Generator Algorithm (PRGA) which gens the keystream. Each step: modifies the state, outputs a keystream byte.

Stream Ciphers

KSA:

```
for i from 0 to 255
    S[i] := i
endfor

j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap(S[i],S[j])
endfor
```

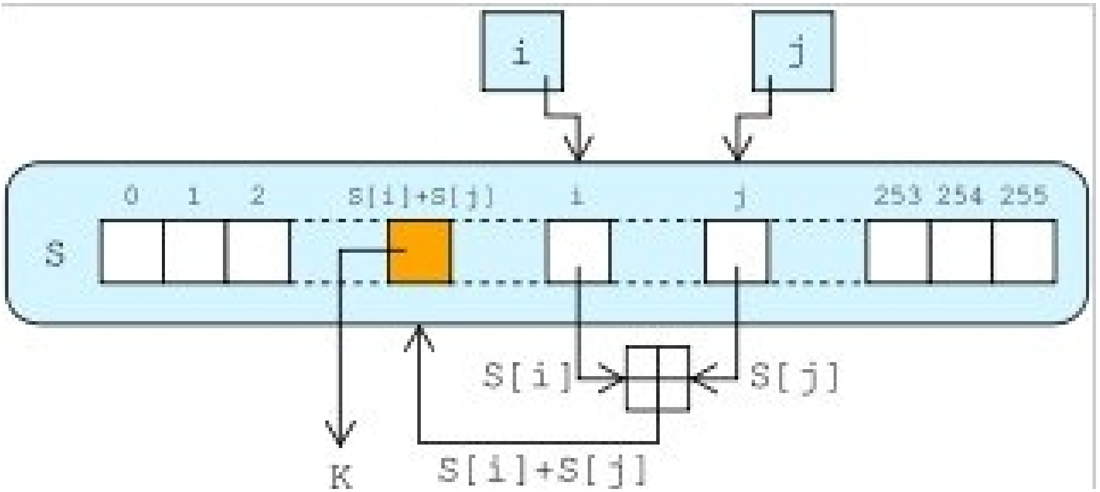
Stream Ciphers

PRGA:

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
endwhile
```

Stream Ciphers

PRGA:




```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
endwhile
```

Stream Ciphers

RC4 Problems:

1. The first few bytes of the keystream are seriously non-random and leak information about the key. By analyzing a large enough number of encrypted messages with the same key, the key can be reconstructed. This is the cause of failure of WEP security. (2001)

2. More correlations were discovered in 2005 leading to the deployment of aircrack-ptw which cracks 128 bit WEP in under a minute (85000 frames with 95% probability).

Stream Ciphers

Other Stream Cipher Considerations:

1. **The same key should never be used twice**:

   Let M_1 and M_2 be two messages sent using key K which produces cipher stream C_K

   Let $E_1 = M_1 \oplus C_K$, $E_2 = M_2 \oplus C_K$

   Both E_1 and E_2 are observed by attacker

   Attacker computes $E_1 \oplus E_2 = M_1 \oplus M_2$

   If M_1 is longer than M_2 then part of M_1 can be computed

   If M_2 is known, then M_1 can be computed

2. **Do not assume that successful decryption means integrity**:

   Let M_O be an original section, M_S the substituted section

   $C_K \oplus M_O \oplus M_O \oplus M_S = C_K \oplus M_S$ (which receiver gets)