Stream Ciphers

Why Stream Ciphers ?

- Speed
  - Initialization
  - Keystream generation
- Resources – memory, power, cpu
- Hardware, software suitability

One-Time Pad

- Developed by Gilbert Vernam in 1918, another name: ***Vernam Cipher***
- The key
  - a truly random sequence of 0's and 1's
  - the same length as the message
  - use one time only
- The encryption
  - adding the key to the message modulo 2, bit by bit.

Encryption $$c_i = m_i \oplus k_i \qquad i = 1,2,3,\ldots$$

Decryption $$m_i = c_i \oplus k_i \qquad i = 1,2,3,\ldots$$

m_i : plain-text bits.
k_i : key (key-stream) bits
c_i : cipher-text bits.

Example

- **Encryption:**
- 1001001 1000110 plaintext
- 1010110 0110001 key
- 0011111 1110110 ciphertext

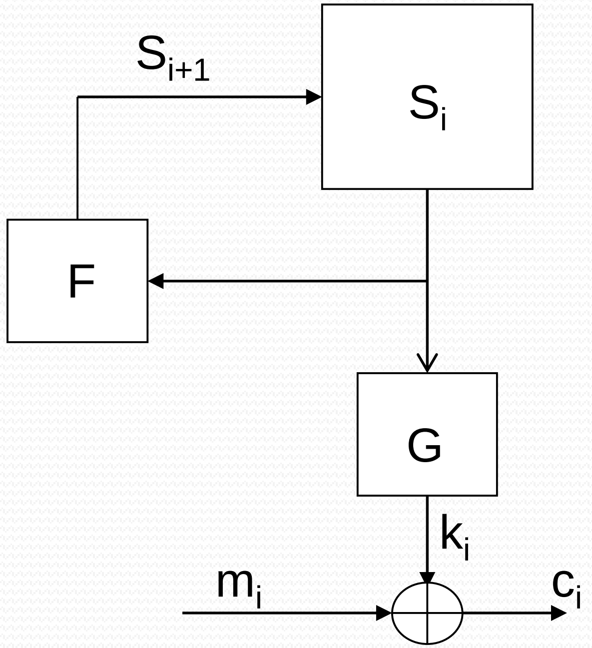
- **Decryption:**
- 0011111 1110110 ciphertext
- 1010110 0110001 key
- 1001001 1000110 plaintext

One-Time pad practical Problem

- Key-stream should be as long as plain-text
- Difficult in Key distribution & Management
- **Solution :**
  - Stream Ciphers
  - Key-stream is generated in pseudo-random fashion form Relatively short secret key

Stream Cipher Model

- **Output function** appears random



S_i : state of the cipher at time t = i.

F : state function.

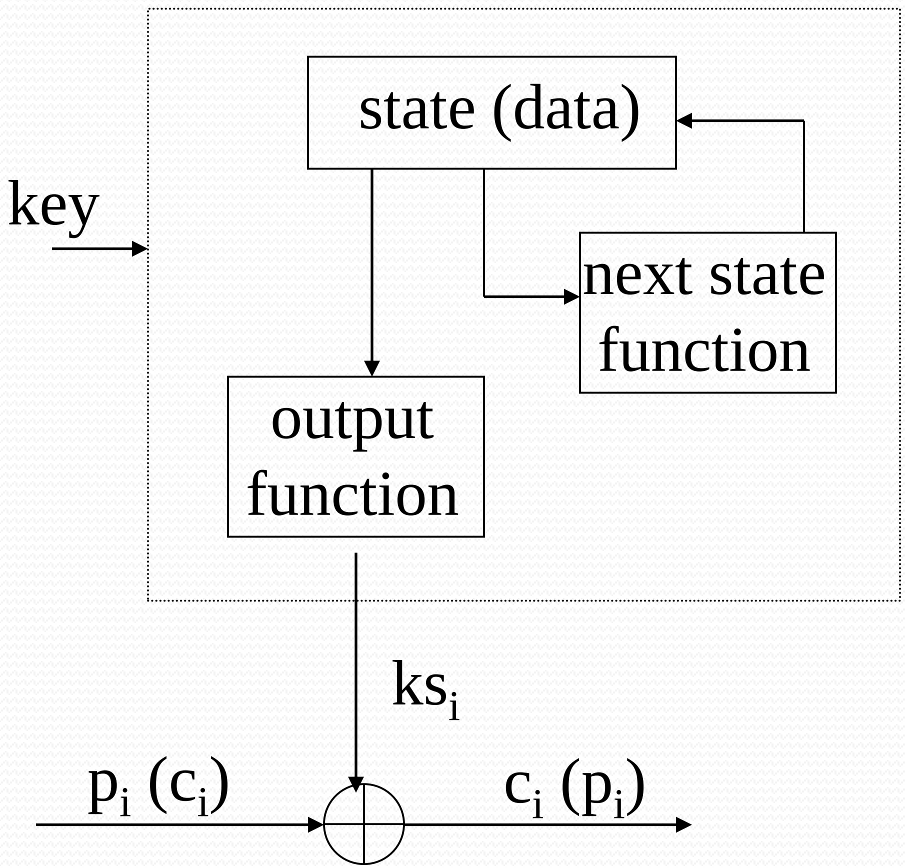
G : output function.

Initial state, output and state functions are controlled by the secret key.

Stream Ciphers

- Synchronous stream cipher
  - Sender and receiver must be in-synch
  - Lost bit garbles all subsequent bits unless synch up
  - Flipped bit garbles only one bit
  - Can precompute key stream
  - Example: RC4, block cipher in OFB mode
- Self-synchronizing stream ciphers
  - Use n previous ciphertext bits to compute keystream
  - Lost bit: synch up after n bits
  - Flipped bit : ≤ next n bits garbled
  - Can't precompute keystream
  - Example: Block cipher in ciphertext feedback (CFB) mode

Stream Ciphers – General Concept

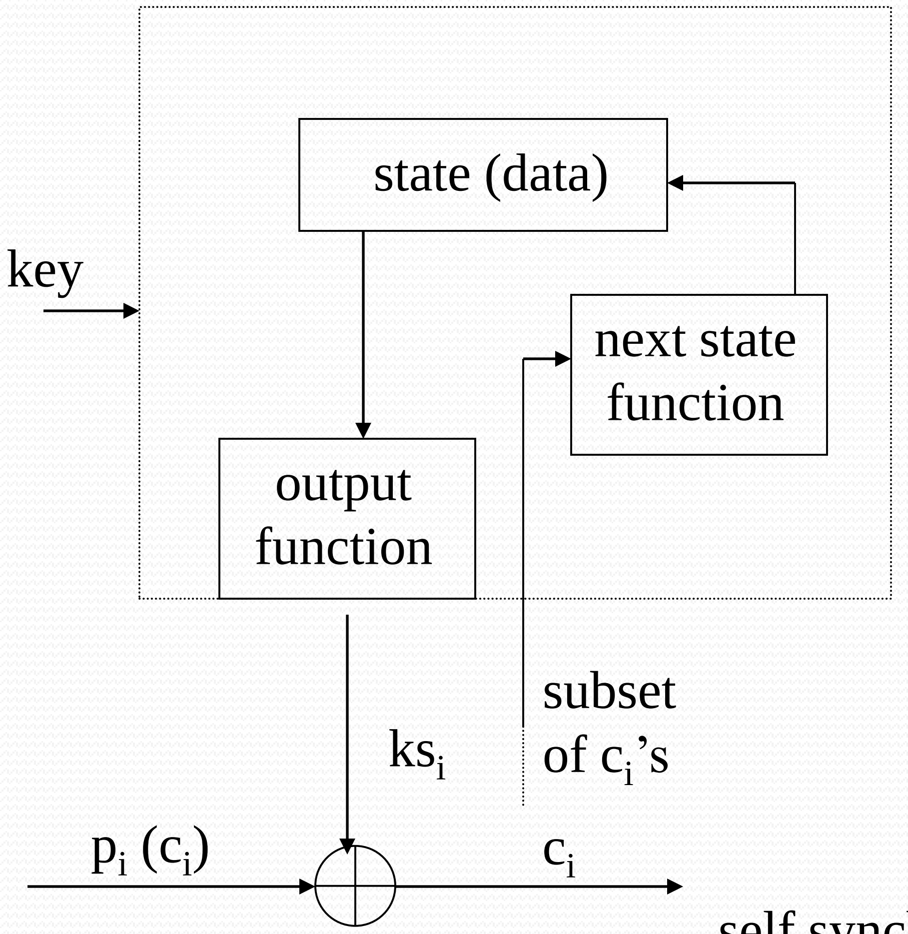


synchronous

State Updates

- FSR based (SOBER, LILI)
- Array Permutations (RC4)

Stream Ciphers – General Concept



- error propagation
- block cipher in CFB mode

self synchronizing

Keystream Properties

- Period
  - Period of 2^{32} repeats after ~ 8.5 minutes when encrypting 1MB/sec
- Random appearance:
  - Runs of 1's or 0's: ½ with length 1, ¼ with length 2, 1/8 have length 3 …
  - Test – little or no compression
  - Dissipates statistics of plaintext
- Complexity:
  - Low ability to define a bit as a linear expression (or algebraic expression) of bits < period bits away
  - No discernable relation to key (seed/initial state) bits

Random Numbers

- Many uses of **random numbers** in cryptography
  - Nonce as Initialize Vector
  - Session keys
  - Public key generation
  - Keystream for a one-time pad
- In all cases these values be
  - statistically random, uniform distribution, independent
  - unpredictability of future values from previous values

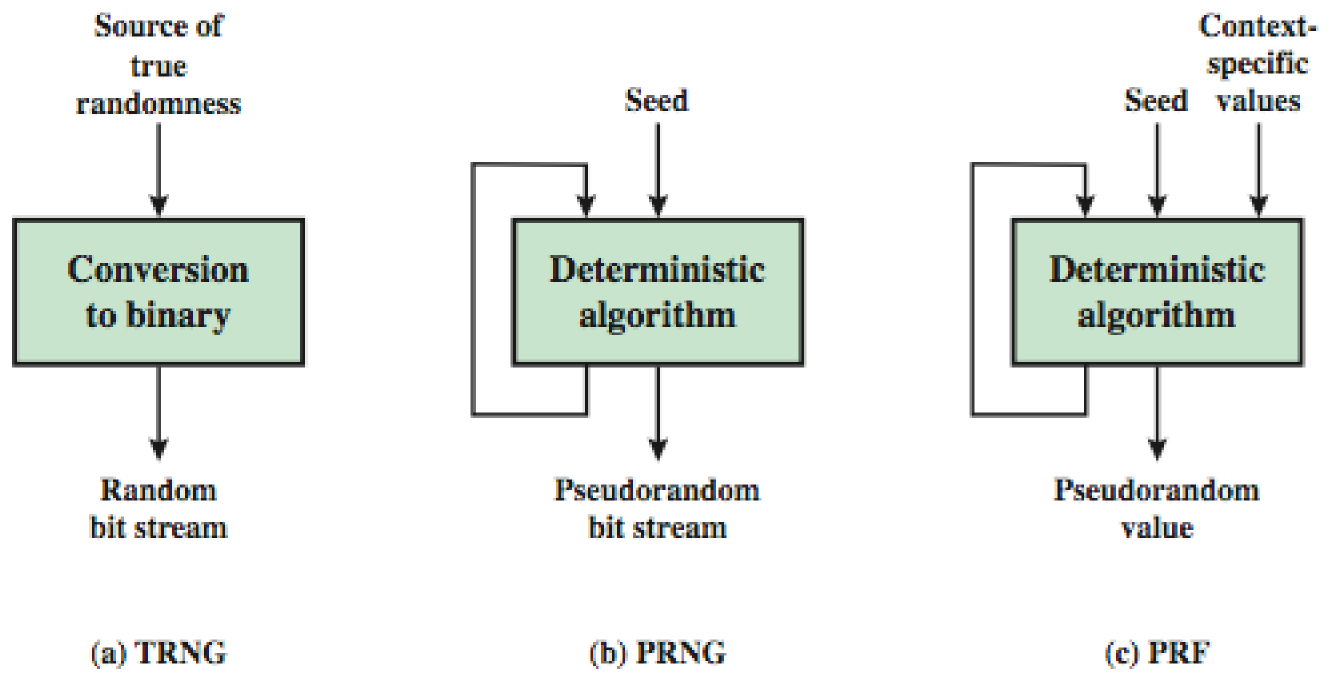
Pseudorandom Number Generators (PRNGs)

- Often use deterministic algorithmic techniques to create "random numbers"
  - although are not truly random
  - can pass many tests of "randomness"

  (ref: Knuth 2nd Volume, Art of Computer Science)

- Known as "Pseudorandom Numbers"
- Created by "Pseudorandom Number Generators (PRNGs)"

Random & Pseudorandom Number Generators

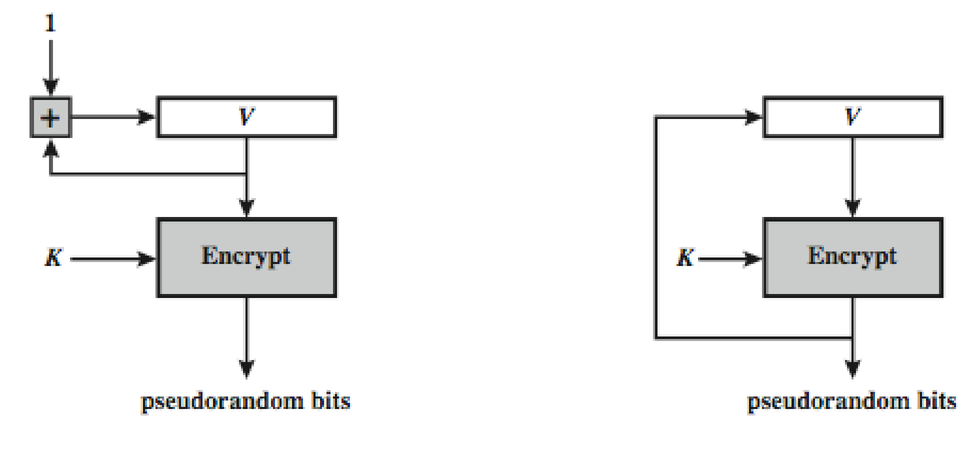


PRNG Requirements

- Randomness
  - uniformity, scalability, consistency
- Unpredictability
  - forward & backward Unpredictability
  - use same tests to check
- Characteristics of the seed
  - Secure
  - if known adversary can determine output
  - so must be random or pseudorandom number

Using Block Ciphers as PRNGs

- For cryptographic applications, can use a block cipher to generate random numbers
- Often for creating ses
- **CTR**
  - $X_i = E_K[V_i]$
- **OFB**
  - $X_i = E_K[X_i-1]$



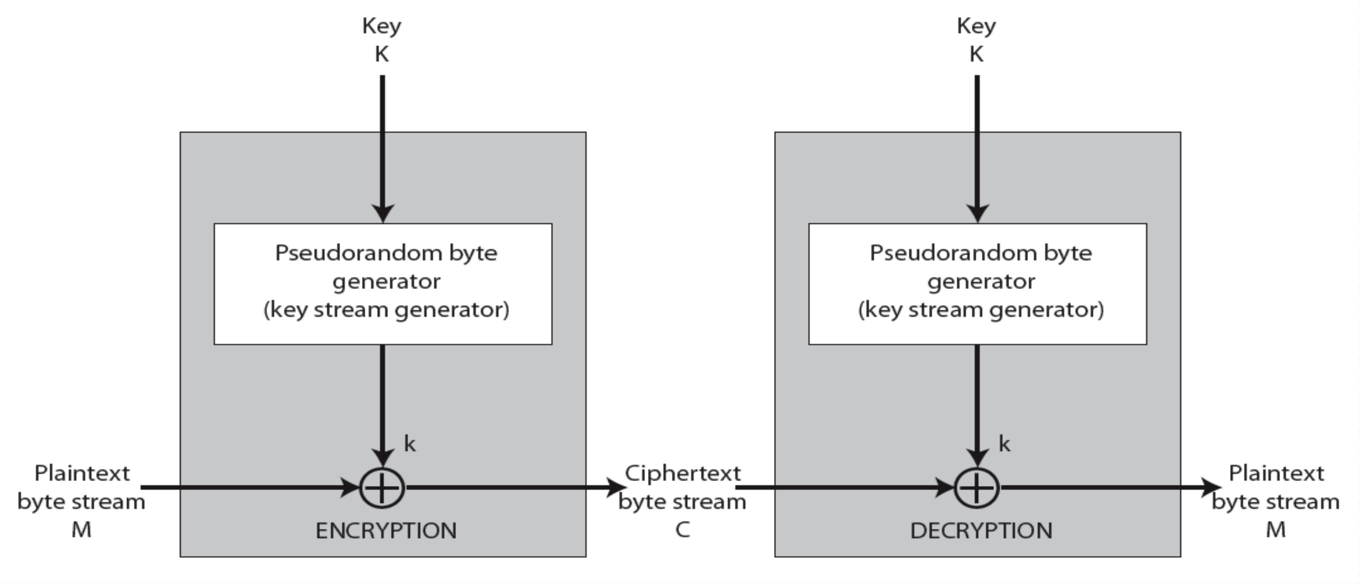
(a) CTR Mode (b) OFB Mode

Stream Ciphers

- Generalization of **one-time pad**
- Stream cipher is initialized with short **key**
- Key is “stretched” into long **keystream**
  - have a pseudo random property
- Keystream is used like a one-time pad
  - XOR to encrypt or decrypt

Stream Cipher Structure

- Randomness of stream key completely destroys statistically properties in message
- Must **never reuse** stream key
  - otherwise can recover messages



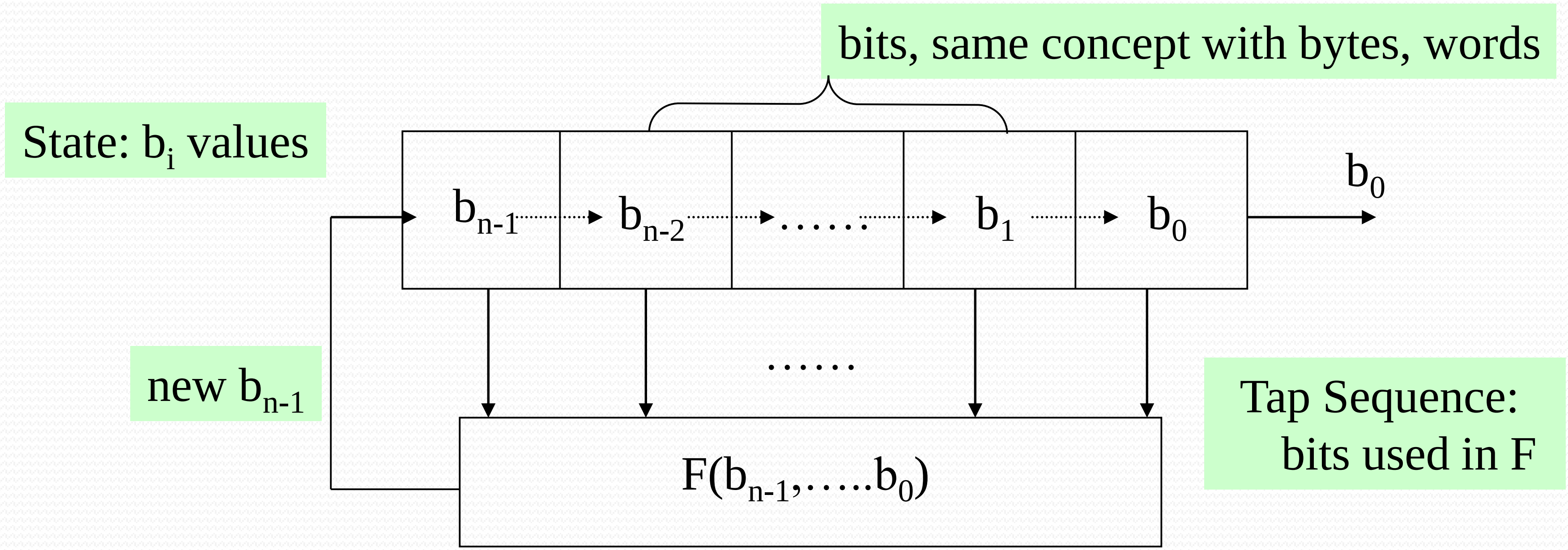
Stream Cipher Properties

- Some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
- Properly designed, can be as secure as a block cipher with same size key
- Benefit : usually ***simpler*** & ***faster***

Stream Ciphers - Approaches

- Feedback Shift Register (FSR) based – useful in hardware
- Block cipher – CTR, CFB, OFB modes
- Components similar to those found in block ciphers

Feedback Shift Register



- Linear F: $b_{n-1} = \bigoplus_{i=0,n-1} \alpha_i b_i$ for $\alpha_i \in \{0,1\}$
- Nonlinear F
- Feedback with Carry Shift (FCSR)

  F: $s = \left(\sum_{i=0,n-1} \alpha_i b_i + c\right)$ for $\alpha_i \in \{0,1\}$

  $b_{n-1} = s \bmod 2$

  $c = s/2 \bmod \log_2$ (# tap bits)

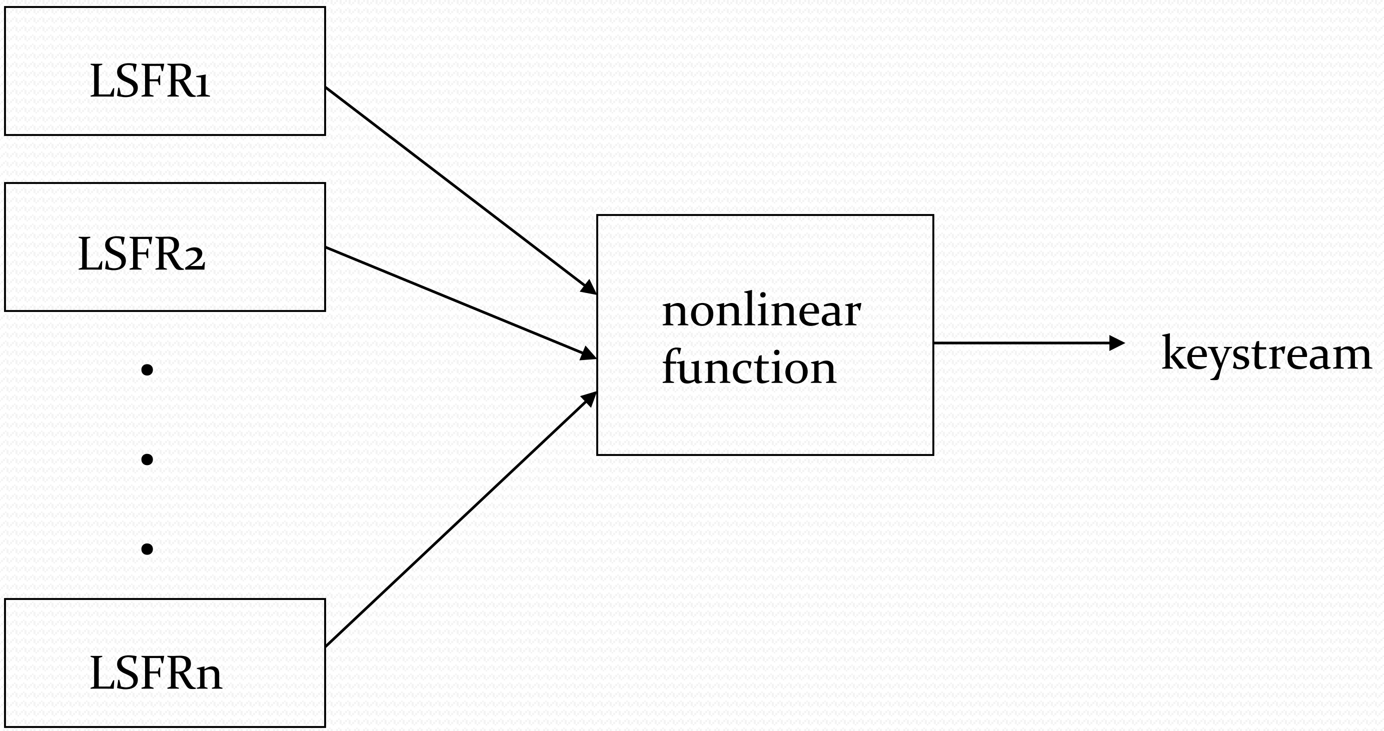
Feedback Shift Registers

- Period
  - LFSR of n bits: Maximum $2^n - 1$
  - FCSR: depends on initial state
  - Non-linear FSR: depends on function, initial state
- Inefficient in Software
  - Small # of bits in tap sequence, easier to break.
  - Large # of bits in tap sequence, slow.
- Security
  - Berlekamp-Massey Algorithm: 2n output bits needed to reproduce the LFSR in $O(n^2)$ time.
  - Non-linear FSR: avoid linear approximations

Variations Utilizing LFSR

Combination generator

- Output bit = nonlinear function on output of multiple LFSRs.
- May clock each LFSR differently
- Various combinations of AND,OR,Thresholds



Variations Utilizing LFSR

- Clock controlled generator
  - Move to next state only on some clock cycles.
  - Move to next state on every cycle but only output bit on some clock cycles.
  - 2nd LFSR may control clock.
- Clock control that affects output is also called stuttering

NESSIE Stream Cipher Submissions

- **None recommended**
- BMGL – too slow, small internal state – time/memory tradeoff attack
- Leviathan - distinguishing attack
- LILI-128 – attack $O(2^{71})$
- SNOW – distinguishing attack
- SOBER-t16 – distinguishing attack
- SOBER-t32 – distinguishing attack
- Both Sober algorithms thought to be subject to side channel analysis

ECRYPT's eStream Contest

- Just ended (3rd round of evaluations finished, winners selected)
  - 4 for software, 4 for hardware
- In third round of evaluations
  - 16 candidates
- 3+ years from time of call for proposals to final report
  - originally November 2004 to January 2008
  - Just ended

- *ECRYPT: European Network of Excellence for Cryptology*

eStream Overview

- Categories
  - key length of 128 bits and an IV length of 64 and/or 128 bits
  - key length of 80 bits and an IV length of 32 and/or 64 bits
- Separate software and hardware categories within each
- Evaluation
  - Security
  - Free of licensing requirements …
  - Performance, range of environments
- Committee is only collecting submissions. Evaluations are done by the general cryptographic community.

eStream Evaluation

- Security Criteria
  - Any key-recovery attack should be at least as difficult as exhaustive search.
  - Distinguishing attacks
    - Interest to the cryptographic community
    - Relative importance of high complexity distinguishing attacks is an issue for wider discussion
  - Clarity of design
- Implementation Criteria
  - Software and hardware efficiency
  - Execution code and memory sizes
  - Performance
  - Flexibility of use

eSTREAM Phase 3 Candidates

Profile 1 (SW)	Profile 2 (HW)
CryptMT (CryptMT Version 3)	**DECIM** (DECIM v2 and DECIM-128)
Dragon	**Edon80**
HC (HC-128 and HC-256)	**F-FCSR** (F-FCSR-H v2 and F-FCSR-16)
LEX (LEX-128, LEX-192 and LEX-256)	**Grain** (Grain v1 and Grain-128)
NLS (NLSv2, encryption-only)	**MICKEY** (MICKEY 2.0 and MICKEY-12 2.0)
Rabbit	**Moustique**
Salsa20	**Pomaranch** (Pomaranch Version 3)
SOSEMANUK	**Trivium**

http://www.ecrypt.eu.org/stream/phase3list.html
key lengths: 128 bits for SW and 80 bits for HW

eSTREAM Winners

Profile 1 (SW)	Profile 2 (HW)
HC (HC-128 and HC-256)	**F-FCSR** (F-FCSR-H v2 and F-FCSR-16)
Rabbit	**Grain** (Grain v1 and Grain-128)
Salsa20	**MICKEY** (MICKEY 2.0 and MICKEY-12 2.0)
SOSEMANUK	**Trivium**

http://www.ecrypt.eu.org/stream/
key lengths: 128 bits for SW and 80 bits for HW

Stream Cipher Examples

- Lists
  - http://en.wikipedia.org/wiki/Stream_cipher
  - http://www.ecrypt.eu.org/stream/
- RC4
- A5/1
- A5/3
- LILI
- Sober
- Trivium
- Lex

RC4

S-Box Creation

```
input key;
if (key < 256 bytes) {
   repeat key until 256 bytes;
}
for (i=0; i < 256; ++i) {
      S[i] = i; // initialize S-Box
      K[i] = ith key byte;
}
j = 0;
for (i = 0; i <256; ++i) {
      j = (j + S[i] + K[i]) mod 256;
      swap(S[i],S[j]);
}
```

Keystream Generator

```
i = 0; j = 0;
loop {
        i = (i+1) mod 256;
        j = (j+S[i]) mod 256;
        Swap(S[i],S[j]);
        t = (S[i] + S[j]) mod 256;
        ks_byte = S[t];
}
```

2 S-Box entries form index into S-Box
Output S-Box entry (byte)

S-Box: key dependent permutation of 0 to 255. (lookup table)

RC4 Cryptanalysis

- Initial keystream byte highly correlated with first few key bytes
  - Recommendations to discard first 256 or 512 output bytes
- Distinguish from random: $\geq O(2^{30.6})$ bytes needed
- Attempts to backtrack to initial state from keystream

Keystream Generator

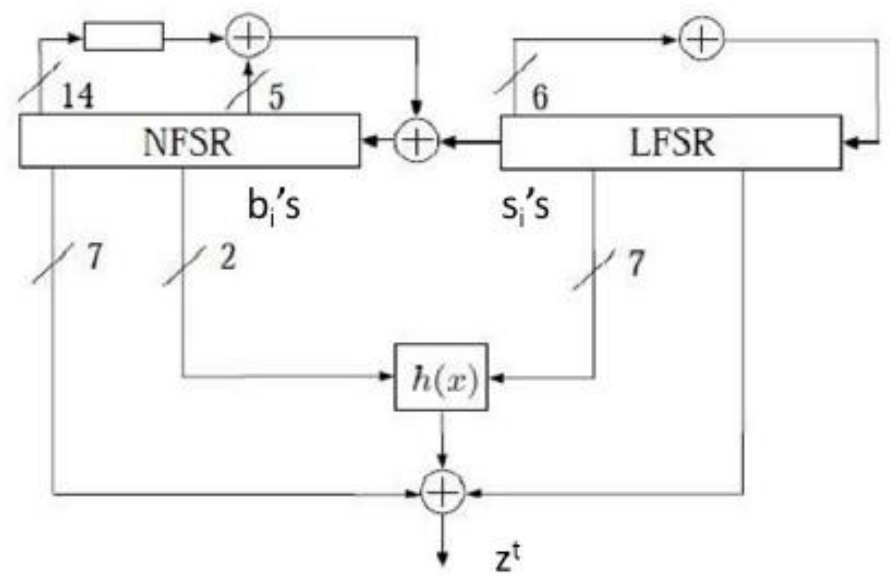
```
i = 0; j = 0;
loop {
        i = (i+1) mod 256;
        j = (j+S[i]) mod 256;
        Swap(S[i],S[j]);
        t = (S[i] + S[j]) mod 256;
        ks_byte = S[t];
}
```

Grain Family of Ciphers

- Grain-128 proposed is one of the four hardware based ciphers enlisted in the eStream portfolio.
- **Grain** is a stream cipher submitted to eSTREAM in 2004 by Martin Hell, Thomas Johansson and Willi Meier.
- A number of potential weaknesses in the cipher have been identified and corrected in Grain 128a.
- Provides 128-bit security.

Specification of Grain-128

- Grain-128 stream cipher consists of three main building blocks, namely, an NFSR, an LFSR and a nonlinear filter function, h(x).



Dept. of Computer Science & Engg.
IIT Kharagpur, India

Specification of Grain-128

- NFSR bits are $b_i, b_{i+1}, \ldots, b_{i+127}$ and
  LFSR bits are $s_i, s_{i+1}, \ldots, s_{i+127}$.
- The update function of the LFSR is given by,

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$$

- The NFSR is updated by,

$$\begin{aligned} b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} \\ + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84} \end{aligned}$$

- The nonlinear filter function h(.) is given by,

$$h = b_{i+12}s_{i+8} + s_{i+13}s_{i+20} + b_{i+95}s_{i+42} + s_{i+60}s_{i+79} + b_{i+12}b_{i+95}s_{i+95}$$

Specification of Grain-128

- The output bit z is defined as,

$$z_i = b_{i+2} + b_{i+15} + b_{i+36} + b_{i+45} + b_{i+64} + b_{i+73} + b_{i+89} + h + s_{i+93}$$

- An initialization phase is carried out before the cipher generates any keystream.
- During initialization the cipher is run for 256 rounds without producing any keystreams and the output bit is fed back to both LFSR and NFSR.
- The 128 bit key, $k = (k_0, k_1, \ldots, k_{127})$ and the 96 bit initialization vector $IV = (IV_0, IV_1, \ldots, IV_{95})$ are loaded in the NFSR and the LFSR respectively as, $b_i = k_i$; $0 <= i <= 127$ and $s_i = IV_i$; $0 <= i <= 95$, rest of the LFSR bits, $(s_{96}, s_{97}, \ldots, s_{127})$ are loaded with 1.

Security of Grain-128

- Algebraic Attack
  - Cube Attack

- Side Channel Attacks
  - Fault Attack
  - Scan attack

Fault injection techniques

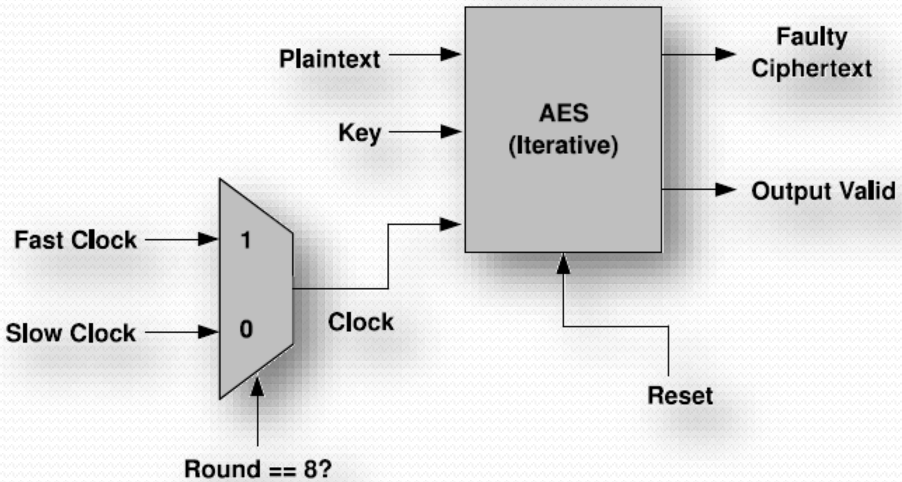
- Transient (provisional) and permanent (destructive) faults
  - Variations to supply voltage
  - Variations in the external clock
  - Temperature
  - White light
  - Laser light
  - X-rays and ion beams
  - Electromagnetic flux

Fault Attack Principle

- Fault attacks are one of the most efficient side channel attacks known till date.
- In this kind of attack, faults are injected during cipher operations.
- The attacker then analyzes the fault free and faulty cipher-texts or key-streams to deduce partial or full value of the secret key.
- The literature shows that both the block ciphers and stream ciphers are vulnerable against fault attack.
- Methods like clock glitch, laser shots have been shown to be a practical way of inducing faults in a crypto-system.

Fault Induction by Clock Glitch

- Non – invasive
- Strategy : Switch system clock
  - Determine max clock frequency
  - Use a faster clock to violate critical path
- Results in setup/hold violation = fault
- Easy/economical to mount
- Can be used to control
  # of faults
- Has been used to inject
  fault in AES



Dept. of Computer Science & Engg.
IIT Kharagpur, India

Fault Injection by Scan -Chain

- Non – invasive
- Strategy : Use scan-chain to insert fault
  - Scan out intermediate state in test-mode
  - Flip-bits in scan-in pattern
  - Run in normal mode = fault in positions of bit-flips
- Easy/economical to mount
- Highest form of controllability over fault
- May not work if some special scan-chain protection schemes are in place
- Often protection schemes are avoided to reduce overhead
  - Very useful fault injection in such cases

Attacks on Grain

- Cryptanalysis based on dynamic cube attack and one fault based attack (LFSR as a fault target) are the only known weaknesses of Grain-128.
- We inject faults in the NFSR and show that secret key of the cipher can be recovered.
- Complexity of our attack is better than the previous one.

Fault Analysis Model

- The attacker is able to induce faults at random positions of the NFSR of the Grain-128 implementation (hardware or software).
- A fault is a bit-flip in the internal register.
- The fault affects exactly one bit of the NFSR at any cycle of operation.
- A fault to an NFSR bit can be reproduced at any cycle of operation, once, it is created.
- The attacker is able to determine and control the cycles of operation of the implementation, i.e., the timing of the implementation is under control of the attacker.
- The attacker can reset the implementation to its original state.
- The attacker can run the implementation with different IV, without changing the key.

Overview of the Attack

- The following steps are carried out in the attack:
  - Determining Fault Location in NFSR
  - Pre-computation of Fault Traces
  - Determining NFSR bits
  - Determining LFSR bits
  - Inversion of states to obtain key

Complexity of Our Attack

- Experimentally it is seen that, 56 faults on an average are required to obtain full internal NFSR bits of the cipher.
- Maximum 256 faults are needed to obtain full LFSR state of the cipher.
- The time complexity of the attack is $O(2^{21})$.

Complexity

Ciphers	*Grain v1*	*Grain-128*	*Grain-128a*
Online Cost	(#Faults, Cost)	(#Faults, Cost)	(#Faults, Cost)
Location	(0,11)	(0,14)	(0,14)
Pre-computation	$(0, 2^{12})$	$(0, 2^{15})$	$(0, 2^{15})$
NFSR	$(70, 2^{10})$	(56, 128)	(128, 128)
LFSR	$(80, < 2^{21})$	$(256, 2^{21})$	$(256, 2^{21})$
Inversion	$(0, r(T))$	$(0, r(T))$	$(0, r(T))$
Total	$(150, < 2^{21})$	$(312, 2^{21})$	$(384, 2^{21})$

Reference

1 Alexandre Berzati et. al. Fault Analysis of Grain-128. Hardware Oriented Security and Trust, IEEE International Workshop on, 0:7-14, 2009.
2. Itai Dinur and Adi Shamir. Breaking Grain-128 with Dynamic Cube Attacks. Cryptology ePrint Archive: Report 2010/570.

Our Attack:

3. Sandip Karmakar and Dipanwita Roy Chowdhury, **Fault analysis of Grain-128 by targeting NFSR**, AFRICACRYPT '11, 298–315, 2011.
4. Sandip Karmakar, Dipanwita Roy Chowdhury, **Fault Analysis of Grain Family of Stream Ciphers.** IACR Cryptology ePrint Archive 2014: 261 (2014)

Recently the following work reports an attack on Grain ciphers with 10 faults or less, but it uses rekeying and requirements of keystreams are higher.

5. Santanu Sarkar et. al. , Differential Fault Attack Against Grain Family with very few Faults and Minimal Assumptions, Cryptology ePrint Archive , 2013/494. http://eprint.iacr.org.