Stream Ciphers

Enes Pasalic
University of Primorska
Koper, 2013

Contents

Chapter 1

Preface

This book has been written as lecture notes for students who need a grasp of the basic principles of stream ciphers.

The scope and level of the lecture notes are considered suitable for (under)graduate students of Mathematical Sciences and Computer Sciences at the Faculty of Mathematics, Natural Sciences and Information Technologies at the University of Primorska.

It is not possible to cover here in detail every aspect of stream ciphers, but I hope to provide the reader with an insight into the essence of the stream ciphers.

Main resource is Chapters 5 and 6 from Handbook of Applied Cryptography, by A. Menezes, P. Oorschot, S. Vanstone.

Enes Pasalic
`enes.pasalic@upr.si`

Chapter 2

Content Description

1. Introduction to stream ciphers, motivation and basic modes of operations.
2. Statistical testing of pseudorandom sequences. Introduction to cryptanalysis.
3. Simple cryptanalysis, and statistical testing (**Exercise**).
4. Generating sequences. Theory of LFSR, Berlekamp-Massey algorithm and linear complexity of sequences.
5. Examples of generation of periodic sequences, applications of Berlekamp-Massey algorithm (**Exercise**).
6. Stream cipher design. Simple LFSR-based ciphers: nonlinear combining generator, nonlinear filtering generator. Theory of Boolean functions. Stream ciphers based on FCSR (Feedback Carry Shift Registers).
7. Boolean functions, design and cryptographic properties. 2-adic theory (**Exercise**).
8. Other LFSR-based stream ciphers. Shrinking, summation and alternating generator. Application examples: E_0 Bluetooth encryption and GSM encryption algorithm A5. Modern design of hardware oriented stream ciphers: Trivium and Grain-128.
9. Software oriented stream ciphers: SNOW 2.0, RC4. Design principles and suitable cryptographic primitives: S-boxes, modular addition. Some theory of groups and finite fields.
10. S-box design, cryptographic properties. Selecting suitable mappings over fields in S-box design. (**Exercise**).
11. Theory of algebraic attacks. Classical and fast algebraic attacks.
12. Distinguishing, fast correlation attacks, synchronization and side-channel attacks.
13. Examples of algebraic attacks and application of distinguishing attacks (**Exercise**).
14. Cryptanalysis of real-life ciphers; how do we break the ciphers.
15. Cryptanalysis of real-life ciphers; eSTREAM candidate ciphers in focus.

Chapter 3

Introduction to Stream Ciphers

Content of this chapter:

- History of stream ciphers.
- Synchronous and asynchronous stream ciphers.
- Practical usage and design principles.
- Time-memory-data trade-off attack on stream ciphers.

Stream ciphers - overview

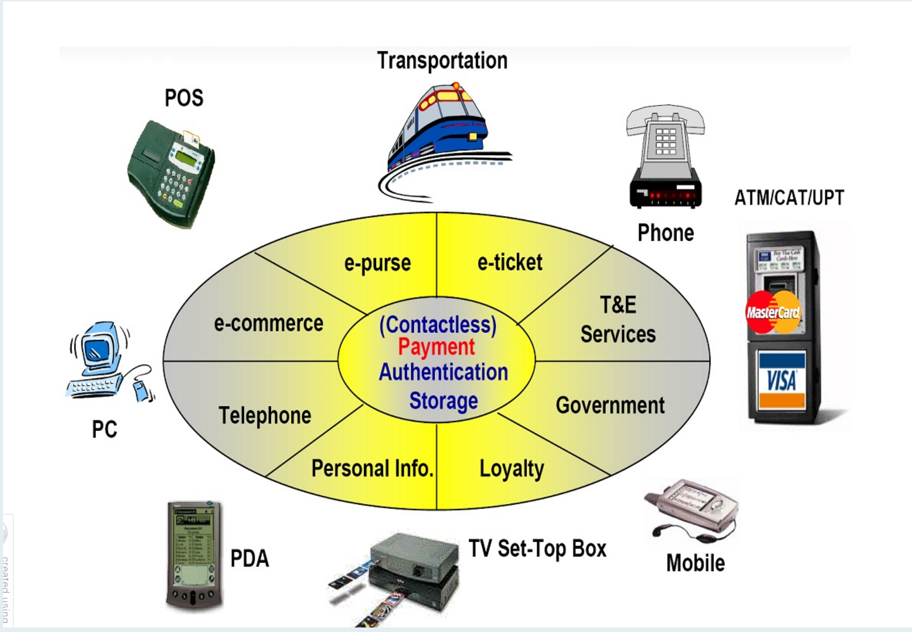
- History of stream ciphers
- Synchronous and asynchronous stream ciphers.
- Practical usage and design principles.
- Time-memory-data trade-off attack on stream ciphers.

Cryptographic services

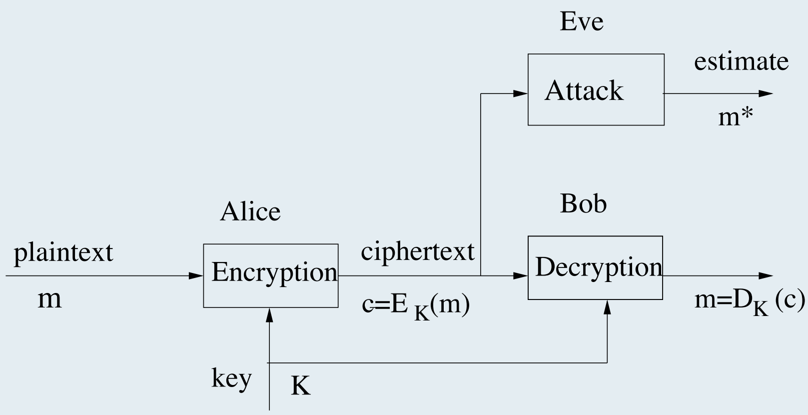
- Cryptography is there to protect the information. "Endless" range of applications. Fundamental security aspects are:
  - Confidentiality
  - Integrity
  - Non-repudation
  - Authentication

• Confidentiality (secrecy) is usually achieved throug symmetric-key primitives. These primitives can also support integrity service (through MAC feature).

Applications of cryptography



Model of a classic cryptosystem



Model of a classic cryptosystem

Symmetric-key cryptography

- Commonly divided into:
  - block ciphers
  - stream ciphers (we focus on this technique)
- Sender and reciever share the same key. N users must exchange $\binom{N}{2} = N(N-1)/2$ keys !
- A fast implementation both in hardware and software.
- Relatively short key length compared to public key cryptography for the 'same security' level.

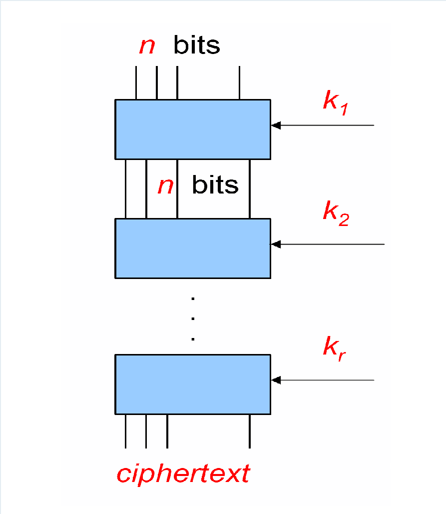
Why stream ciphers . . . ?

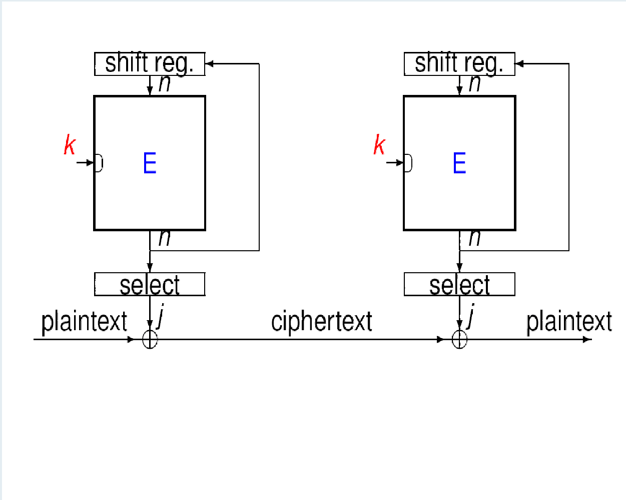
- . . . when we have so many good block ciphers ?
  - IDEA, KASUMI, FEAL . . .
  - DES (never broken, been around for 25 years)
  - Triple DES (incresed key length)
  - AES (Advanced Encryption Standard)
- Dozens of block ciphers; do we need dozens of stream cipher ?

"Stream ciphers - Dead or Alive"
Asiacrypt 2004, invited talk by Adi Shamir

- Block ciphers are : well understood and analyzed, standardized, and can work in stream cipher mode .

Block ciphers in stream cipher mode





- Block cipher in OFB (Output Feedback Mode) = stream cipher.

Requirements on the design of stream cipher

- Towards standardization, open competition (Call for Proposals) was initiated by eSTREAM project,

  `http://www.ecrypt.eu.org/stream/`.

- To compete with block ciphers two profiles of stream cipher:
  - Faster in software applications than BC (Profile I SW)
  - Stream cipher for low circuit complexity (Profile II HW).

- Main advantage of stream ciphers - speed . Important when encrypting a huge ammount of data , e.g. video streaming, hard disc encryption
- E.g. RFID tag - no more than 5000 gates (AES 4-10.000 gates).

eSTREAM proposals - SW

- There were 34 submissions,
  - 32 synchronising and 2 self-synchronising
  - 7 submissions offering encryption and integrity

Phase 3	Archived Phase 2	Archived
CryptMT v3	ABC	F-FCSR
DRAGON	DICING	FROGBIT
HC-128	Phelix	Fubuki
LEX v2	Polar Bear	Hermes8
NLS v2	Py	MAG
Rabbit		Mir-1
Salsa20		POMARANCH
SOSEMANUK		SSS
		TRBDK3 YAEA
		Yamb
(8)	(5)	(10)

eSTREAM proposals - HW

- Hardware proposals seem to be more resistant to cryptanalysis

Phase 3	Archived Phase 2	Archived
DECIM v2	Achterbahn	MAG
Edon-80	Hermes8	SFINKS
F-FCSR-H v2	LEX	SSS
Grain v1	NLS	TRBDK3 YAEA
MICKEY v2	Phelix	Yamb
MOUSTIQUE	Polar Bear	
POMARANCH v3	Rabbit	
Trivium	Salsa20	
	TSC-4	
	VEST	
	WG	
	ZK-Crypt	
(8)	**(12)**	**(5)**

Prominent stream ciphers

- Prominent applications include:
  - E_0 stream cipher - privacy in Bluetooth applications
  - A5 family of ciphers - encryption in GSM standard
  - RC4 cipher - used in SSL/TLS, WEP

(Wired Equivalent Privacy), wireless network security standard.

DESIGN IDEA : Process a finite key to derive very long stream of pseudorandom keystream bits.

- In between Vigenere and Vernam cipher.

Vigenère cipher- symmetric key scheme

- Polyalphabetic *substitution* stream/block cipher.
- Message and key from the English alphabet, i.e., $\mathcal{M}, \mathcal{K} \in \{A, B, \ldots, Z\}$. Then $\mathbf{m} = m_0, m_1, \ldots$ is encrypted to $\mathbf{c} = c_0, c_1, \ldots$ as follows.
- Make a transformation

$$A \leftrightarrow 0, B \leftrightarrow 1, \ldots, Z \leftrightarrow 25.$$

- The same transformation is applied to the key

$$\mathbf{K} = K_0, K_1, \ldots, K_{l-1}.$$

- Corresponding message and key sequence are denoted $\mathbf{m}'$ and $\mathbf{K}'$, respectively.

Vigenère cipher cont.

- Encrypted integer sequence $\mathbf{c}' = c'_0, c'_1, \ldots$ is obtained using,

$$c'_i = m'_i + K'_{i \bmod l} \bmod 26, \quad i = 0, 1, 2, \ldots. \tag{1}$$

- Ciphertext $\mathbf{c}$ is derived from $\mathbf{c}'$ using the reverse transformation,

$$0 \leftrightarrow A, 1 \leftrightarrow B, \ldots, 25 \leftrightarrow Z$$

- To recover the sequence of the original message one applies

$$m'_i = c'_i + (26 - K'_{i \bmod l}) \bmod 26, \quad i = 0, 1, 2, \ldots.$$

- The same transformation as above is applied to $\mathbf{m}'$ to retrieve $\mathbf{m}$.

Vigenère cipher example.

- Let $\mathbf{m} =$ THISCIPHERISNOTSECURE, and $K =$ UNSECURE.

- First derive,

$$\begin{array}{lcccccccccccccccc}
\mathbf{m}' = & 19, & 7, & 8, & 18, & 2, & 8, & 15, & 7, & 4, & 17, & 8, & 18, & 13, & 14, & \cdots; \\
\mathbf{K}' = & 20, & 13, & 18, & 4, & 2, & 20, & 17, & 4| & 20, & 13, & 18, & 4, & 2, & 20, & \cdots.
\end{array}$$

- The encrypted sequence $\mathbf{c}'$ is then computed,

$$\mathbf{c}' = 13, 20, 0, 22, 4, 2, 6, 11, 24, 4, 0, 22, 15, 8, 10, 22, 24, 15, 12, 21, 6.$$

- The resulting ciphertext is $\mathbf{c} =$ NUAWECGLYEAWPIKWYPMVG.

Vernam and One time pad ciphers

- Consider the encryption scheme of message $m \in \{0,1\}^*$ using the key $k \in \{0,1\}^*$ given by,

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

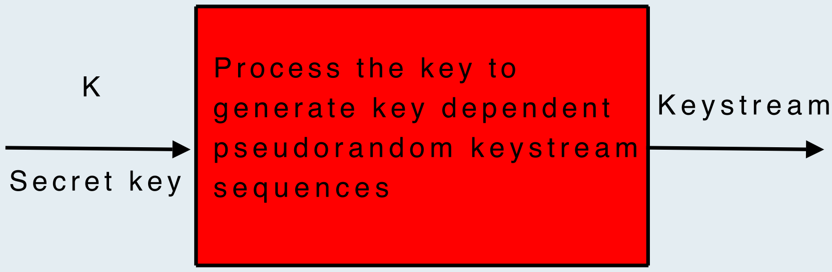
- Known as Vernam cipher - example of a stream cipher .
- Decryption is performed using $c_i \oplus k_i = m_i \oplus k_i \oplus k_i = m_i$.
- If the keystream bits are generated independently and at random then we call the cipher *one-time pad*..

One time pad impractical

- It is unconditionally secure provided that the length of the key is at least as the length of the message, and the key is never reused.
- A practical scheme would use a key of finite length (only $k = 128$ bits) and generate a pseudo random sequence (keystream) for encryption and decryption.
- Such a scheme is never unconditionally secure , and the aim is to make it computationaly secure .

Design perspectives

- The idea is to expand a fixed lenth key into pseudorandom keystream sequence. The keystream should look as random as possible !



- Processing through a Finite State Machine (FSM); periodicity.

Security objectives

- The keystream sequnce should satisfy:
  - Large period
  - Good statistical properties
  - Low correlation between keystream sequence and secret state/key
  - Security against current cryptanalysis (and forthcoming)
  - Speed and/or hardware efficiency

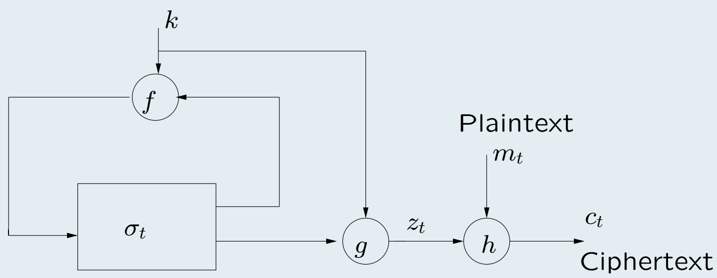
Implementation targets

- To compete in hardware complexity less challenging than speed in software.
  - Not iterated cipher as block cipher (10 rounds - 20 rounds)
  - Unrolling the loop (iteration) increases speed but also hardware complexity.
  - Streamciphers process small amount of data (bit level)
- Processing on a bit level in software is disadvantegous

- AES generates blocks of 128 bits, works on a byte level.

Software and hardware design substantially differ (rare examples satisfy both)

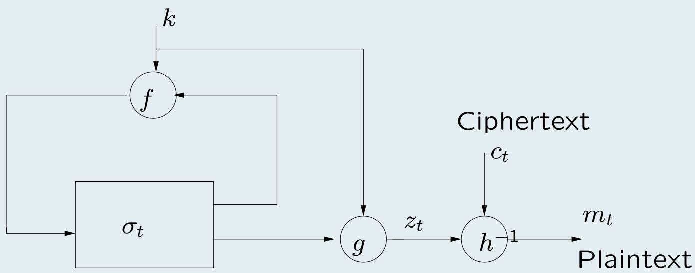
General structure of a synchronous stream cipher



$$\begin{aligned}\sigma_{t+1} &= f(\sigma_t, k),\\ z_t &= g(\sigma_t, k),\\ c_t &= h(z_t, m_t),\end{aligned}$$

- Keystream does not depend on plaintext and ciphertext.

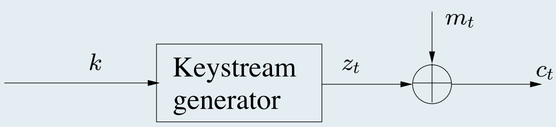
Decryption procedure



$$
\begin{aligned}
\sigma_{t+1} &= f(\sigma_t, k), \\
z_t &= g(\sigma_t, k), \\
m_t &= h^{-1}(z_t, c_t),
\end{aligned}
$$

Additive stream ciphers

- No complicated mechanism is required for function h, just to be invertible.
- Take h to be bitwise modulo two addition, i.e. $h(z_t, m_t) = z_t \oplus m_t = c_t$, and h^{-1} is same as h.



General model of an additive stream cipher

Two time pad

- One problem with additive stream ciphers is that using the same key in multiple sessions you get the same keystream blocks.
- Then these keystream blocks induce,

  $$c_i^1 = m_i^1 \oplus z_i, \;\; c_i^2 = m_i^2 \oplus z_i \quad \Rightarrow c_i^1 \oplus c_i^2 = m_i^1 \oplus m_i^2, \;\; i = 1, 2, \ldots$$

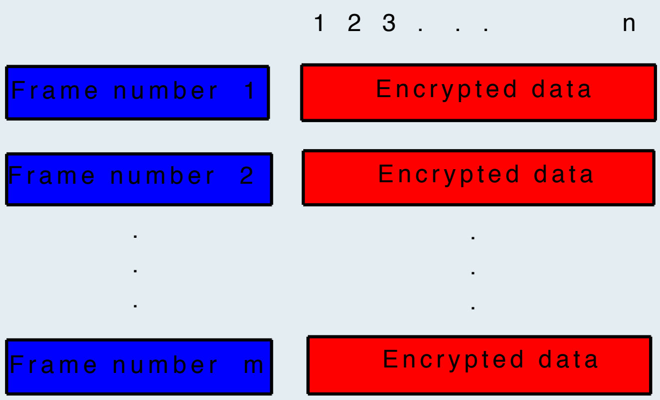
  Using redudancy of plaintext we can recover m^1 and m^2.
- Changing the key for each session or lost synchronization is impractical, e.g. encryption of streaming video sequence.
- Solution (**first reason**) is to use initialization vector (IV) which is changed upon re-synchronization.

Properties of synchronous stream ciphers

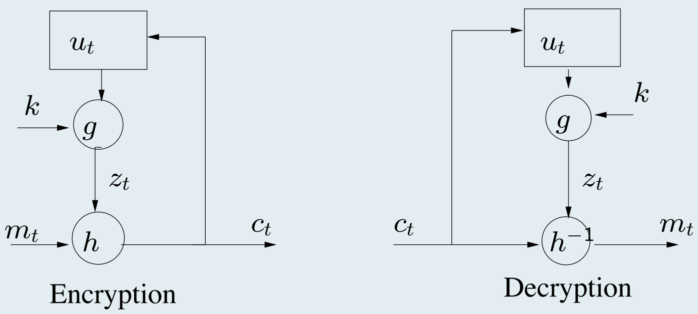
- Each symbol is encrypted independently, limited error propagation due to transmission or malicious modification.
- The problem is deletion or insertion of symbols. All decrypted plaintext is erroneous. Need for perfect synchronization.Possibility of detecting data manipulation.
- One solution is to split the message into *frames* numbered with frame numbers. Need for Initialization Vector (IV) publicly known (**second reason**).
- Each new frame uses new IV value together with the same key.

Use of frames for resynchronization

- IV value is derived from frame number through pubicly known algorithm



General structure of a self-synchronous stream cipher



- Keystream depends on say v bits of ciphertext . Worse error propagation, a single bit error causes v incorrect bits .
- However it recovers from the loss of synchronisation after v correct cyphertext symbols.

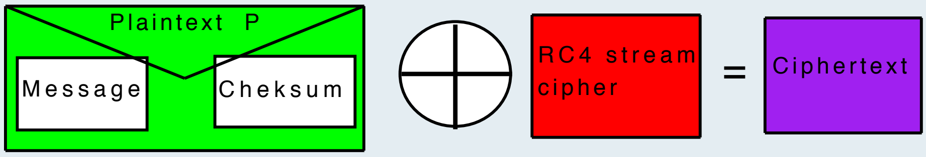
Comparison of main features - synchronous vs. asynschronous

- Asynchronous stream are better in case of synchronization loss.
- Synchronous ciphers have no error propagation.
- Larger error propagation implies that it is easier to detect mallicious modification of ciphertext.
- Harder to detect insertion or deletion of ciphertext digits due to self-synchronization property than in case of synchronous ciphers. Need for data integrity protection as well.

Malleability of binary additive stream ciphers

- What happens if we intentionally flip one bit in the ciphertext?
- Say $c_j^* = c_j \oplus 1$, where $c_j = p_j \oplus z_j$. Then $p_j^* = c_j^* \oplus z_j = p_j \oplus 1$.
- There is no way to detect this if the flip is suitably placed , 200$ may become 2000$!
- This can be prevented by MAC (message authentication code) appended to the message.

Appending MAC for stream ciphers



- Flipping a bit in the ciphertext still result in the change of one bit in plaintext !
- But the computed MAC on the plaintext is not the same as one transmitted. Enough for protection ?

Attacking MAC

1 0 0 1 0 0 1 1 1 1	Plaintext
0 1 1 0 0 0 1 1 0 1	Keystream
1 1 1 1 0 0 0 0 1 0	Ciphertext

- Append 2 bit linear CRC; (first bit = sum of odd bits, second bit = sum of evens).

1 1 1 1 0 0 0 0 1 0	1 0	Ciphertext with CRC
0 1 1 1 0 0 0 0 1 0	0 0	Flip 1st bit and 1st bit of CRC
0 0 0 1 0 0 1 1 1 1		Decrypted plaintext

- CRC check is approved at intended receiver !

Selecting CRC

- Do we solve the problem by encrypting CRC ?
  - If the CRC is linear then encryption does not help.
  - Encrypted CRC provides redundancy to test the key

• Encrypt nonlinear CRC: Typical CRC lengths are 16 or 32 bits. On average 2^8 or 2^{16} bit flips enough (birthday paradox).

• Increase the value of CRC, overload the frame and smaller efficiency of the cipher.

• Stream cipher may provide data integrity but these seems to be hard to design.

Operation phases

- Solution is to use initialization vector (IV) which is changed upon re-synchronization or for multiple sessions.

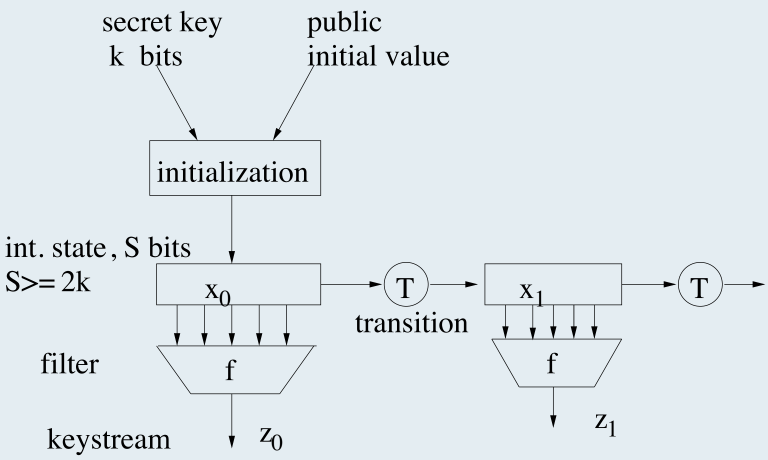
  1. In the *setup phase*, the key and publicly known IV are used to initialize the internal state of the cipher.

  2. In encryption/decryption phase the next state is updated (compute σ_{t+1}) and next block of encrypted/decrypted data is generated.

  3. In the case of synchronisation loss or for a new session with the same key use another (known) IV and the same key.

State of the cipher

- IV is added to the key so that the cipher has larger initial state (prevents from TMD attacks), usually $S \geqslant 2k$.



Selecting IV

- The value of IV is not secret , can be sent through a public channel.
- Still, it **should be chosen at random** to avoid precomputation attacks.
- **IDEA** If IV is known in advance to the attacker, he may precompute the output keystream sequence for different keys . *Birthday paradox* implies the complexity well beyond brute force attacks.
- In practice IV is commonly of same length as key, the internal state of the cipher $S = 2k$.

Reusing IV

- There must me a mechanism to prevent from resuing IV with same key !
- Main reason for the **failure** of many protocols, such as WEP.
- Reusing IV with the same key implies generating the same keystream sequence. Two-time pad applies.
- Even worse if $c_i^{(1)} = p_i^{(1)} \oplus z_i$ and $c_i^{(2)} = p_i^{(2)} \oplus z_i$ then the knowledge about $p_i^{(1)}$ gives:

$$p_i^{(2)} = c_i^{(1)} \oplus c_i^{(2)} \oplus p_i^{(1)}$$

RC4 example

- Byte oriented popular cipher used in e.g. SSL/TLS, WEP. Designed by Ron Rivest (**R**ivest **C**ipher 4); Rivest is the first R in RSA.
- It's been around from 1987; designed at RSA security.
- The main feature is its exceptional speed both in software and hardware. Approximately 2-3 times faster than AES.
- Measurement depends on the platform, optimization etc. You may expect around 120 MB/sec as encryption speed.
- Requires a proper use to avoid known attacks

RC4 example

- The early design did not include the IV vector; easiest to append IV to key - might lead to certain weaknesses.
- Consists of 256 element array of 8-bit integers, state vector denoted by $\mathbf{S}$.
- The design of WEP is flawed as IV is only 24 bits long. We assume that length of key and IV are same 128 bits.
- The content of IV and key is treated as a sequence of 16 integers in the range 0 to 255.

RC4 example cont.

1. Initialize state vector $\mathbf{S}[0\ 1\ 2 \ldots 255] = [0\ 1\ 2\ \ldots 255]$
2. Use temporary vector T of 256 bytes, and fill with key and IV values.
3. Use vector T to produce the **initial permutation** of $\mathbf{S}$:

$j = 0$

for $i = 0$ to 255

$j = (j + \mathbf{S}[i] + T[i]) \pmod{256}$

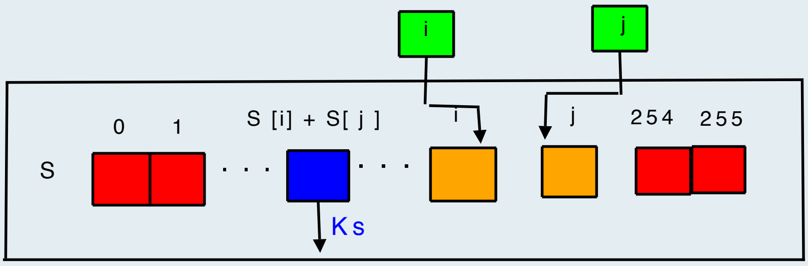
SWAP $(\mathbf{S}[i], \mathbf{S}[j])$

RC4 example keystream generation

```
i = 0;  j = 0

while true

i = ( i+1) % 256;
j = ( j+ S[ i ]) % 256;
swap (S [ i ], S [ j ]);
t = ( S [ i ] + S [ j ]) % 256;
Ks = S [ t ];
```



RC4 summary

- RC4 starts as identity permutation but changes with time - the only operation is SWAP.
- The internal state at any time is $256 \times 8 + 2 \times 8 = 2064$ bits. The latter stands for indices i, j.
- However, efficient state size (entropic) is

$$|S| = \log_2(256!(2^8)^2) \approx 1700 bits.$$

- Knowledge of all $|S|$ state bits enough to predict remaining keystream sequence. Though hard to go backwards and deduce the key from given state.
- Knowledge about key reveals of course everything. Period is difficult to estimate, empirically very long.

eSTREAM performance testing framework

Encryption rate for long stream Likely the most important criterion,

$$\text{EncSpeed} = \frac{\text{NmbofBytes}}{250\mu s} = \text{cycles/byte.}$$

Stream cipher candidates should be superior to AES

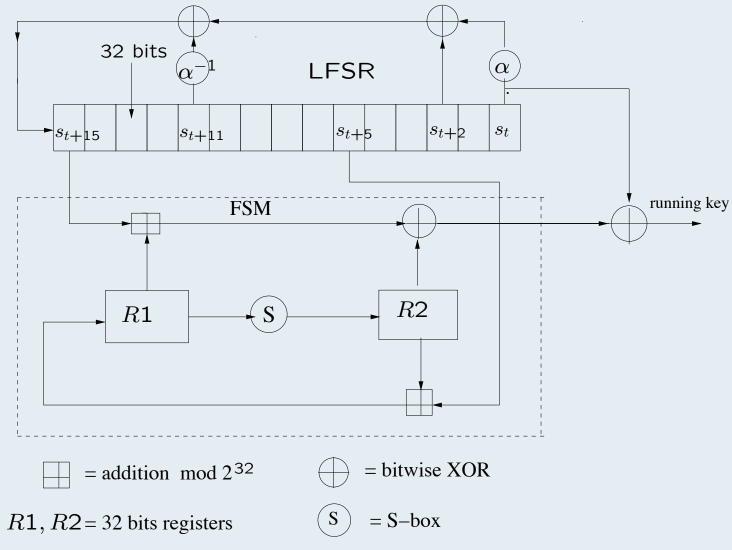
Packet encryption rate Measure performance for packet transmission (40, 576, 1576 Bytes)

Agility Encryption of many streams on a single processor: initiate many sessions (16MB RAM) and encrypt 250 Byte for each session

Key and IV setup + MAC generation Least crucial test, effciency of IV setup is already in packet rate. Key setup is negligible to key generation and exchange.

SNOW 2.0 - software oriented stream cipher

SW oriented cipher, designed in Lund - eSTREAM candidate.



Software performance of SNOW 2.0

- Key=IV=128 bits; CPU speed 1.65 GHz

Encrypted 22 blocks of 4096 bytes (under 1 keys, 22 blocks/key)
Total time: 415015 clock ticks (244.87 μsec)
Encryption speed (cycles/byte): 4.61
Encryption speed (Mbps): 2943.95

Encrypted 350 packets of 40 bytes (under 10 keys, 35 packets/key)
Total time:411499 clock ticks(242.80 μsec)
Encryption speed (cycles/packet): 1175.71
Encryption speed (cycles/byte): 29.39
Encryption speed (Mbps): 461.29
Overhead: 538.2%

IV setup - performance bottleneck

- In a frame based transmission IV is different for each frame.
- Stream cipher must induce IV and Key - all state bits are complex functions of key and IV bits.

Simple key initialization $\rightarrow$ faster cipher; security may be compromized

- Excellent example is usage of RC4 in WEP protocol. Only 3 Bytes of IV (?!) are prepended to the Key; enough to break RC4 in WEP !

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

Scenario of attacks

ciphertext-only The cryptanalyst tries to recover the key , or a part of the key, or more plaintext by only observing the ciphertext.

known-plaintext The goal is to recover the key or a part of the key, having some plaintext and the corresponding ciphertext .

chosen-plaintext Like above but cryptanalyst is able to choose any plaintext and to obtain the corresponding ciphertext.

chosen-ciphertext Attacker has access to the decryption equipment ; can decrypt any ciphertext. The goal is to find the key, (securely embedded in the equipment), from the ciphertext-plaintext pairs.

Measuring attacks' complexity

Time (computational) complexity Expected number of operations to perform the attack.

Memory complexity Expected number of storage units .

Data complexity Expected number of data needed for attack (ciphertext blocks, ciphertext/plaintext pairs ...).

Definition No attack on stream cipher should have all the complexities less than brute force attack , i.e. checking for all possible keys.

Trading the complexities

There are always trade-offs between these complexities !

Computational brute force Check exhaustively all states and compare with the keystream. Data and memory complexity $\mathcal{O}(S)$, time complexity $\mathcal{O}(2^S)$.

Memory brute force Precompute $\mathcal{O}(S)$ keystream bits for any possible state. On-line attack, find the keystream sequence in the list. Time and data complexity $\mathcal{O}(S)$, memory complexity $\mathcal{O}(2^S)$.

Can we do better than that ?

Time-Memory-Data trade-off attack

- There are 2^S different initial states of the cipher.

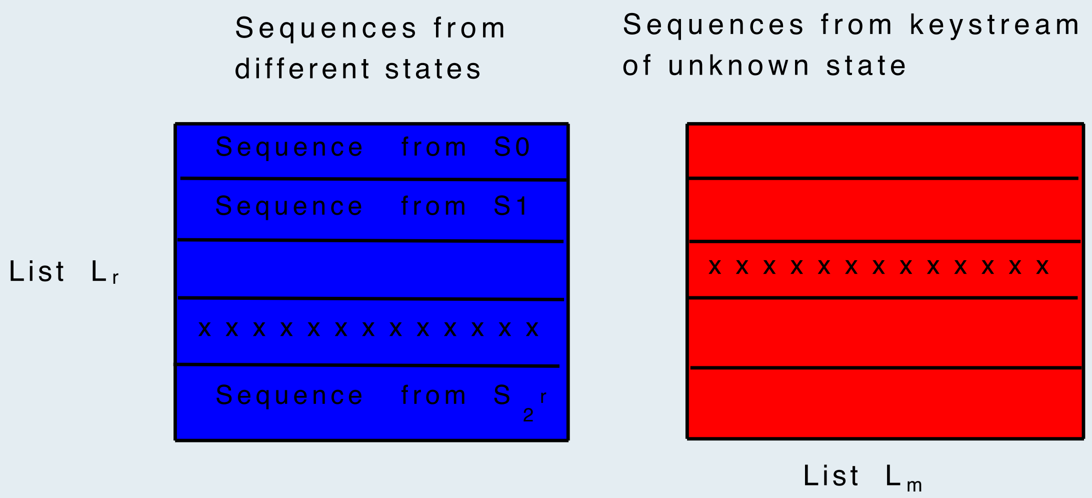
  1. **Generate** 2^r **different states and observe** S **keystrem bits** $z_1, \ldots, z_S$. Call this list L_r.

  2. **Observe a keystream of length** $2^m + S - 1$ **from unknown state** S_0. Collect 2^m overlapping sequences of length S, list L_m.

  3. If we **find the same sequence in the lists**, go backwards to get S_0.

  4. Probability of match is 0.5 for $r + m = s$ (birthday paradox)!

Visualizing the lists



Time-Memory-Data trade-off attack cont.

Time complexity is $\mathcal{O}(rS2^m)$ – test 2^m sequences of length S using logarithmical search in list L_r.

Memory complexity $\mathcal{O}(S2^r)$ for storing the list L_r.

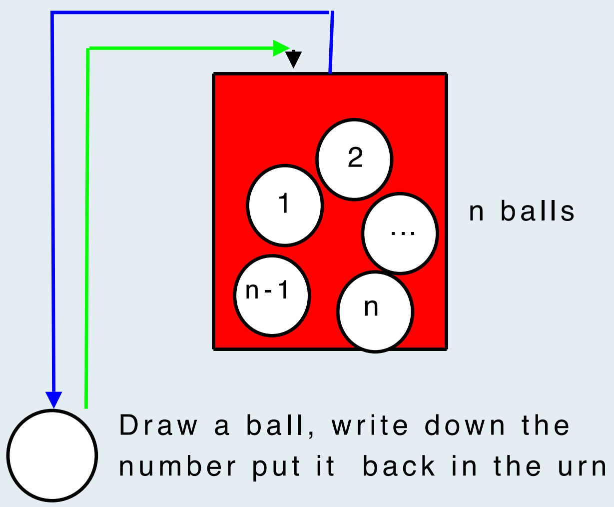
Data complexity $\mathcal{O}(2^m)$, observed keystream.

- The best trade-off is to take $r = m$ then the all complexities are $\mathcal{O}(2^{S/2})$ using $r + m = S$.
- Thus if $S < 2k$, we have an attack that breaks the cipher.

Size of the state

- Increased state size does protect cipher from TMD attacks (and some similar) BUT
  - Stream ciphers are supposed to be extremely fast
  - Loss of synchronization is a common scenario; re-initialization
- The larger the state size the more time is needed for initialization. Thus, choose the state size reasonably large !

Ocupancy problem



- What is the probability, after drawing u balls, that there was at least one collision.

Birthday paradox

- SETUP: Given an urn with n balls numbered z_1 to z_n. Suppose that $u < n$ balls are drawn one at the time, with replacement and the numbers are listed. What is the probability of at least one coincidence ?
- Assume we draw u balls, say $z_{i_1}, \ldots, z_{i_u}$, where $i_1, \ldots, i_u \in [1, n]$.
- The choice of z_{i_1} is arbitrary. The probability that $z_{i_2} \neq z_{i_1}$ is $1 - \frac{1}{n}$.
- Also $Pb(z_{i_3} \neq z_{i_2}, z_{i_3} \neq z_{i_1}) = 1 - \frac{2}{n}$, and so on.

Birthday paradox cont.

- The probability of no collision is,

$$(1 - \frac{1}{n})(1 - \frac{2}{n}) \cdots (1 - \frac{u-1}{n}) = \prod_{i=1}^{u-1} (1 - \frac{i}{n}). \quad (2)$$

- The series expansion of exponential function gives,

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$$

- For a small x ($x \approx 0$) we have $e^{-x} \approx 1 - x$.
- Then the equation (2) becomes

$$\prod_{i=1}^{u-1} (1 - \frac{i}{n}) = \prod_{i=1}^{u-1} e^{-\frac{i}{n}} = e^{-\frac{u(u-1)}{2n}}$$

Birthday paradox cont.

- Hence the probability of at least one collision is $\epsilon = 1 - e^{-\frac{u(u-1)}{2n}}$.
- Easy to show that $u \approx \sqrt{2nln\frac{1}{1-\epsilon}} = c\sqrt{n}$.
- For $n = 365$ and $u = 23$ this probability is ≈ 0.5. That is, among 23 persons in the same room with probability 0.5 two persons were born on the same day.
- Identifying $n = 2^S$ we need to store $u = 2^{\frac{S}{2}}$ sequences then with probability $\frac{1}{2}$ we have a collision.

Further reading

- Section 6.1 in Menezes et al. "Handbook of applied cryptography"
- Internet survey articles such as:
  - "On the role of the inner state size in stream ciphers", by Eric Zenner, available at http://eprint.iacr.org
  - " Some Thoughts on Time-Memory-Data Tradeoffs", by Alex Biryukov.

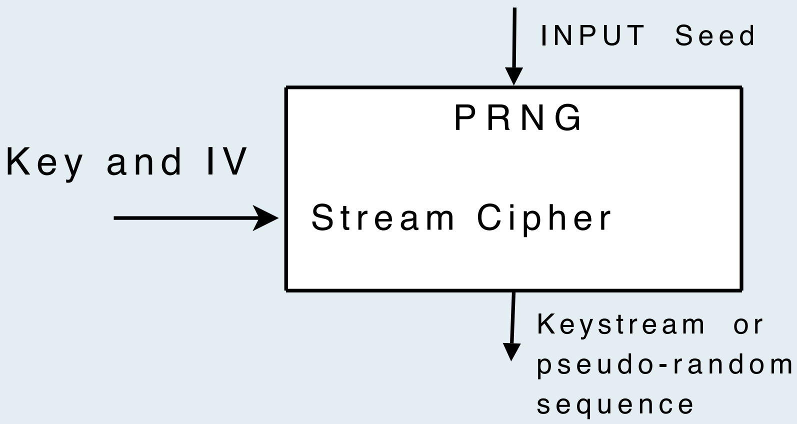
Chapter 4

Statistical Tests

Content of this chapter:

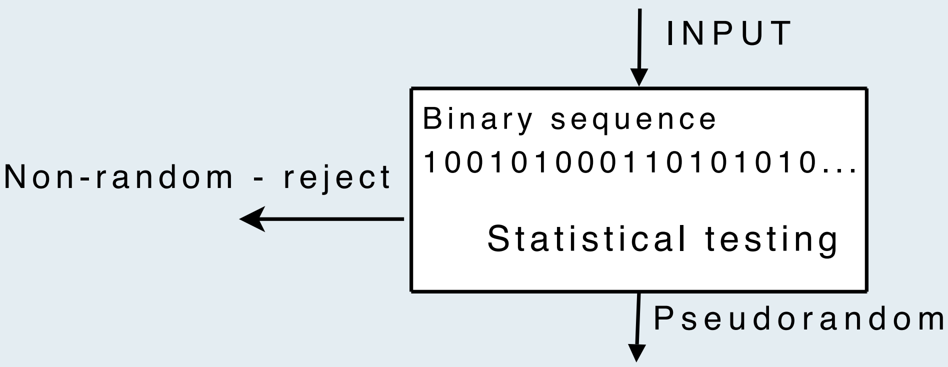
- Statistical properties of sequences.
- NIST statistical test.
- PRNG examples and cryptographically secure PRNG.
- LFSR and its usage in stream cipher design.
- Introduction to cryptanalysis.

PRNG and Stream ciphers



Almost identical notions, though we reserve the usage of stream cipher for fast and IV related applications.

Pseudorandomness of the sequence



Idea: Input sufficiently long sequence (keystream) generated by stream cipher. Apply as many statistical tests as necessary to establish pseudorandom properties (sequence behaves almost as random sequence)

Statistical testing

- Many statistical tests, five basic tests suite, Mauer's universal statistical test, NIST statistical test. Well-known (not sufficient) is Golomb's tests

  1. Measure the number of zeros and ones in the sequence, $\#1 = \#0$
  2. Define run $(1, \ldots, 1)$ (or $(0, \ldots, 0)$) of length k. Half the runs have length 1, a quater have length 2 etc.
  3. The autocorrelation function should be small (look for similarity with shifted sequence)

$$r(\tau) = \frac{1}{T}\sum_{i=0}^{T-1}(-1)^{s_n + s_{n+\tau}}.$$

Performing statistical testing

- Two hypothesis :
- A Null hypothesis H_0: Sequence being tested is *random*
- An alternative hypothesis H_a:Sequence being tested is *not* random

- For each applied test either accept or reject null hypothesis - apply relevant randomness statistic that has a distribution of possible values.

1. A theoretical reference distribution calculated using mathematical models
2. From this a critical value is determined (typically, this value is "far out" in the tails of the distribution, say 99 % point).
3. Compute test statistic value on the sequence - if this value exceeds the critical value, reject the null hypothesis.

Statistical hyptothesis testing

TRUE SITUATION	CONCLUSION	
	Accept H_0	Accept H_a (reject H_0)
Data is random (H_0 is true)	No error	Type I error
Data is not random (H_a is true)	Type II error	No error

- Type I error has less significant impact - random sequence did not pass the test (not often happens). The probability of Type I error is called level of significance , denoted α.
- The value of α is set prior to test, commonly $\alpha = 0.01$
- Probability of Type II error is denoted β. A "bad" generator produced sequence that appears random. The value of β is harder to compute than α due to many possible ways of nonrandomness.

Statistic on random sample

- Probabilities α, β and sample size n are related so that specifying two of them the third value can be computed.
  - Select sample size n and α (probability of Type I error)
  - Choose cutoof point for acceptability so that β is minimized.
- Each test is based on a calculated test statistic value S which is a function (statistic) of the data.

Important to specify proper statistic:

Statistic is efficiently computed

Follows approximately $N(0,1)$ or χ^2 distribution

Probability density function

Definition: If the result X of an experiment can be any real number, then X is said to be a continuous random variable .

Definition: A probability density function of X is a function $f(x)$ which can be integrated and satisfies:

$$\begin{array}{ll} \text{(i)} & f(x) \geqslant 0, \text{ for all } x \in \mathbb{R} \\ \text{(ii)} & \int_{-\infty}^{\infty} f(x)dx = 1; \\ \text{(iii)} & \forall a, b \in \mathbb{R}, P(a < X \leqslant b) = \int_a^b f(x)dx. \end{array}$$

Normal distribution

Definition: Random variable X has a normal distribution with *mean value* μ and *variance* σ^2 if its probability function is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\{\frac{-(x-\mu)^2}{2\sigma^2}\}, \ -\infty < x < \infty$$

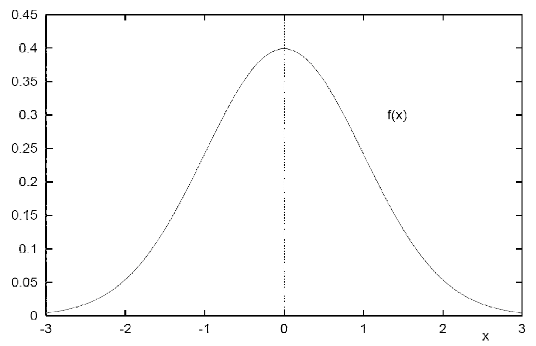


Figure 5.1: *The normal distribution* $N(0, 1)$.

Testing with normal distribution

We can compute probability $P(X > x) = \alpha$,

α	0.1	0.05	0.025	0.01	0.005	0.001
x	1.2816	1.6449	1.9600	2.3263	2.5758	3.0902

Entry $\alpha = 0.05, x = 1.645$ means X exceeds 1.645 about 5% of time

- Assume statistic X of **random sequence** follows $N(0,1)$ distribution.

1. Given significance level α compute a treshold (critical value) x_α so that $P(X > x_\alpha) = P(X < -x_\alpha) = \alpha/2$.
2. Compute X_S for the sample output sequence:
   - If $X_S > x_\alpha$ or $X_S < -x_\alpha$ the sequence fails the test .
   - Otherwise sequence pass the test .

Example of modelling with normal distribution

Autocorrelation test: Given a binary sequence $s = s_0, \ldots, s_{n-1}$. Let $1 \leqslant d \leqslant \lfloor n/2 \rfloor$.

- Calculate autocorrelation function,

$$A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$$

- The suitable statistic is,

$$X_S = 2\left(A(d) - \frac{n-d}{2}\right)/\sqrt{n-d},$$

  which approximately follows $N(0,1)$ distribution.
- Both small and large values of $A(d)$ are unexpected, two-sided test should be performed.

χ^2 distribution

When squares of v independent normal random variables are summed.

Definition: Random variable X has a χ^2 distribution with v degrees of freedom if its probability function is:

$$f(x) = \frac{1}{\Gamma(v/2)2^{v/2}} x^{v/2-1} e^{-x/2},\ x \geqslant 0,$$

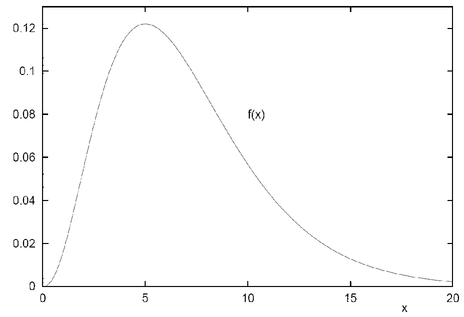


Figure 5.2: *The χ^2 (chi-square) distribution with $v = 7$ degrees of freedom.*

χ^2 distribution - example

Frequency (monobit) test: Given a binary sequence $s = s_0, \ldots, s_{n-1}$. Detremine if the number of 0's and 1's in s are approximately the same.

- Let n_0 and n_1 be the number of 0's and 1's
- The suitable statistic is,

$$X_S = \frac{(n_0 - n_1)^2}{n},$$

which approximately follows χ^2 distribution with 1 degree of freedom.

- For $\alpha = 0.01$ the probability $P(X > x) = \alpha$ gives $x_\alpha = 6.635$. One-sided test of course !

Modelling with normal and χ^2 distribution

- The test statistic for the standard normal distribution is of the form $z = (x - \mu)/\sigma$, where x is the sample test statistic value, and μ and σ^2 are the expected value and the variance of the test statistic.

AC test: $\mu = (n-d)/2$ and $\sigma^2 = (n-d)/4$; $X_S = 2(A(d) - \frac{n-d}{2})/\sqrt{n-d}$

- The χ^2 distribution is used to compare the goodness-of-fit of the observed frequencies of a sample measure to the corresponding expected frequencies of the hypothesized distribution.

- The test statistic is of the form $\sum((o_i - e_i)^2/e_i)$, where o_i and e_i are the observed and expected frequencies of occurrence of the measure, respectively.

Run test with χ^2 distribution

- **Purpose:** Determine whether the number of runs (of either zeros of ones) of various lengths in the sequence s is as expected for a random sequence.

Random seq.- Expected nmb. of runs of length i: $e_i = (n - i + 3)/2^{i+2}$.

- Run is either BLOCK or GAP, e.g. length 3 : 01110 or 10001
- Let k largest integer i for which $e_i \geqslant 5$.
- Let B_i, G_i be the number of blocks and gaps, respectively, of length i in s for each i, $1 \leqslant i \leqslant k$. The statistic used is:

$$X_S = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}.$$

Statistical testing NIST suite I

1. The Frequency (Monobit) test determines whether the number of ones and zeros in a tested sequence are approximately 1/2, as is expected for a truly random fair binary sequence. All subsequent tests depend on the passing of this test.

2. The Test for Frequency within a Block determines whether the frequency of ones in an M-bit block of the tested sequence is approximately M/2.

3. The Runs test : determine whether the total numbers of runs of ones and zeros of various lengths is as expected for a random sequence. Oscillation between zeros and ones is too fast or too slow.

4. The Test for the Longest-Run-of-Ones in a Block determines whether the longest run of ones within an M-bit block of the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence of M bits.

Statistical testing NIST suite II

5. The Random Binary Matrix test uses disjoint submatrices formed from the entire sequence to check for linear dependence among fixed length substrings.

6. The Discrete Fourier Transform (Spectral) test uses the peak heights of the Discrete Fast Fourier Transform of the tested sequence to detect periodic features.

7. The Non-overlapping Template Matching test determines whether there are too many occurrences of predefined aperiodic patterns.

8. The Overlapping Template Matching test also determines whether there are too many occurrences of predefined patterns.

Statistical testing NIST suite

9. The Maurer's "Universal Statistical" test determines whether or not the tested sequence can be significantly compressed without loss of information.

10. The Lempel-Ziv Compression test examines the number of cumulatively distinct patterns to determine how far the sequence can be compressed.

11. The Linear Complexity test determines whether or not the sequence is complex enough to be considered random: Determine the length of an LFSR that would produce the sequence. A short feedback register implies non-randomness.

12. The Serial test checks for the uniformity of the distribution(s) of overlapping m-bit patterns for varying pattern lengths,m. Random sequences exhibit uniformity: every m-bit pattern should appear as frequently as every other m-bit pattern, on average.

Statistical testing NIST suite IV

13. The Approximate Entropy test compares the frequency of overlapping blocks of two consecutive lengths (m and m+1) against the expected result for a random sequence.

14. The Cumulative Sums (Cusums) test determines whether the maximum absolute value of the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of such a cumulative sum for random sequences.

15. The Random Excursions test determines if the numbers of visits of the cumulative sum random walk to integer levels (‘‘states’’) between successive zero level crossings distribute as expected for a truly random sequence.

16. The Random Excursions Variant test extends the Random Excursions test by examining level crossing distributions across multiple excursions between zero level crossings.

Linear congruential generator

Produces pseudorandom sequence of numbers,

$$x_n = ax_{n-1} + b \pmod{m},\ n \geqslant 1,$$

using secret seed x_0, and generator parameters.

- **Cryptographic security :** NONE. Given a part of the output sequence, the remainder of the sequence can be reconstructed (linear recursion).

- No need to know a, b, m. Commonly a single bit is outputed.

▶ Still it passes many statistical tests, except few sophisticated ones.

- **MORAL:** Statistical tests are necessary but not sufficient.

(k, l)-Linear congruential generator

Let $M \geqslant 2$, and $1 \leqslant a, b \leqslant M - 1$. Define $k = \lceil \log_2 M \rceil$ and let $k + 1 \leqslant l \leqslant M - 1$. For a seed x_0, $0 \leqslant x_0 \leqslant M - 1$, define

$$x_i = ax_{i-1} + b \pmod{M}$$

for $1 \leqslant i \leqslant l$, and define

$$f(x_0) = (z_1, z_2, \ldots, z_l),$$

where

$$z_i = x_i \bmod 2,\ \ 1 \leqslant i \leqslant l.$$

Then f is a (k, l)-Linear congruential generator.

$(5,10)$-LCG (toy example)

Example: We can obtain $(5,10)$-PRBG by taking $M = 31$, $a = 3$ and $b = 5$ in the LCG.
Associated linear mapping is

$$x \mapsto 3x + 5 \pmod{31}$$

For instance the seed $x_0 = 1$ gives sequence, (anything but 13 !)

$$\begin{array}{cccccccccc} 8, & 29, & 30, & 2, & 11, & 7, & 26, & 21, & 6, & 23, \ldots \\ 0, & 1, & 0, & 0, & 1, & 1, & 0, & 1, & 0, & 1, \ldots \end{array}$$

The sequence is easily distinguished from truly random sequence

- We construct a next bit predictor for the sequence.

Constructing the next bit predictor

Next bit predictor: Probabilistic algorithm $B_i$predicts i-th bit with $Pb \geqslant 1/2 + \epsilon$, based on observation of previous $i - 1$ bits.

Theorem: Let f be a (k,l)-PRBG. Then B_i is an ϵ-next bit predictor iff,

$$\sum_{(z_1,\ldots,z_{i-1}) \in (\mathbb{Z})_2^{i-1}} p(z_1,,\ldots,z_{i-1}) \times p(z_i = B_i | (z_1,,\ldots,z_{i-1})) \geqslant 1/2 + \epsilon.$$

Example: For any $1 \leqslant i \leqslant 9$ define $B_i = 1 - z_{i-1}$, B_i predicts that $0 \to 1$ or $1 \to 0$ is more likely than $0 \to 0$ or $1 \to 1$.

- Each B_i for our $(5,10)$-LCG is a $9/62$- next bit predictor. The next bit predicted correctly with $Pb = 1/2 + 9/62 = 20/31$.

Constructing distinguisher from the next bit predictor

Input: an l-tuple $(z_1, \ldots, z_l)$
1. Compute $z := B_i(z_1, \ldots, z_{i-1})$
2. **if** $z = z_i$ **then**

$$\mathbf{A}(z_1, \ldots, z_l) = 1$$
$$\textbf{else}$$
$$\mathbf{A}(z_1, \ldots, z_l) = 0$$

Theorem: Let B_i be an ϵ-next bit predictor of a (k, l)-PRBG f. Let p_1 be the probability distribution induced on $(\mathbb{Z}_2)^l$ by f, and p_0 uniform probability distribution on $(\mathbb{Z}_2)^l$. Then, $\mathbf{A}$ is an ϵ-distinguisher of p_1 and p_0.

Meaning $|E(p_0) - E(p_1)| \geqslant \epsilon$; $E(p_j)$ is expected value of the output of $\mathbf{A}$ over distributions p_j.

$1/P$ pseudo random generator

Cryptographically insecure, given here for historical reasons. Modern variant is called FCSR.

- Usual setup is:
  - Prime P and base b related to expansion of $1/P$, $\gcd(b, P) = 1$.
  - Sequence of base b with period $P - 1$.

Example: Let $b = 10$, $P = 503$. Then

$$1/P = \underbrace{00198\ 80715\ \ldots\ 43339\ 96023\ \ldots\ 33001\ 98807}_{502 \text{ digits}} \ldots$$

- We only need a segment of $k = \lceil \log_{10}(2 \cdot 503^2) \rceil = 6$ to recover P, and extend segment back and forward .
- Need just basic continued fraction representation.

Recovering P in $1/P$ pseudo random generator

$$\frac{433,399}{1,000,000} = 0.433399 = \cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{1+\cfrac{1}{16+\cfrac{1}{6+\cfrac{1}{1+\ldots}}}}}}}$$

- From the sequence :

$$\frac{1}{a_1}, \ldots, \frac{1}{a_5} = \frac{1}{2}, \frac{1}{3}, \frac{1}{3}, \frac{1}{1}, \frac{1}{16}$$

construct the convergents using the rule:

$$\frac{A_1}{B_1} = \frac{1}{a_1}; \frac{A_2}{B_2} = \frac{a_2}{a_1 a_2 + 1}; \frac{A_i}{B_i} = \frac{a_i A_{i-1} + A_{i-2}}{a_i B_{i-1} + B_{i-2}};$$

until the first $k = 6$ digits are $q_{m+1} \ldots q_{m+k} = 433399$.

$$\frac{A_1}{B_1} = \frac{1}{2} = 0.5; \frac{A_2}{B_2} = \frac{3}{7} = 0.48\ldots; \frac{A_2}{B_2} = \frac{10}{23} = 0.434\ldots;$$

$$\frac{A_3}{B_3} = \frac{13}{30} = 0.43333\ldots \frac{A_4}{B_4} = \frac{218}{503} = 0.433399$$

Cryptographically secure pseudorandom bit generator

Definition: A pseudorandom bit generator (PRGB) is said to pass the **next bit test** if there is no polynomial time algorithm that using the first l bits of sequence s can predict $(l+1)$-bit with probability significantly greater than $1/2$.

Definition: A PRGB that passes the next bit test is called a cryptographically secure pseudorandom bit generator (CSPRBG).

- Universality of the next-bit test : A pseudorandom bit generator passes the next-bit test if and only if it passes all polynomial-time statistical tests. (see Stinson)

Blum-Blum-Shub CSPRBG

- We can construct CSPRBG assuming that integer factorization is intractable.

1. Generate two large primes $p, q \equiv 3 \pmod 4$ and compute $n = pq$.
2. Select a random seed $s \in [1, n-1]$ such that $gcd(s, n) = 1$; and compute $x_0 \leftarrow s^2 \pmod n$.
3. For i from 1 to l do the following:

$$\begin{aligned} x_i &\leftarrow x_{i-1}^2 \pmod n \\ z_i &\leftarrow \text{the least significant bit of } x_i (x_i \pmod 2) \end{aligned}$$

Blum-Blum-Shub number theory framework

The condition $p, q \equiv 3 \pmod 4$ implies $n \equiv 1 \pmod 4$, each quadratic residue has exactly one square root also a quadratic residue (exercise).

1. **Forward direction** Knowledge of N sufficient to generate $x_0, x_1, \ldots$, and $z_0, z_1, \ldots$. Complexity roughly $\mathcal{O}(\log n)^2$- low efficiency.
2. **Backward direction** Given n the factors of n are necessary and sufficient to compute sequence backwards, $x_0, x_1, x_2, \ldots$ (exercise)
3. The factors of n are necessary to find an poly time ϵ-distinguisher to guess parity of x_{-1} given x_0 (see Stinson)
4. **Period** Select n s.t. $ord_{\lambda(n)/2}(2) = \lambda(\lambda(n))$ and the seed x_0 s.t. $ord_n = \lambda(n)/2$. Then $T(x_0) = \lambda(\lambda(n))$.

Usage of Blum-Blum-Shub

- There are two main reasons why we do not use BBS keystream generator:
  - Though only one modular squaring is needed; BBS is much slower than well-designed stream cipher.
  - The key is much larger; 1024 bits are used for a secure RSA setup.
- One may extract $j = \log\log n$ least significant bits (asymptotically); while not compromizing the security of BBS generator. Still, it is not sufficiently fast.

- Good for generation of cryptographic keys

Asymptotically secure bounds

- Notions of *asymptotically secure in sense of indidstinguishibility from random sequence* and next bit polynomial time ϵ-distinguisher are equivalent (Yao).

Definition: For $n =$ size of seed and $M =$ length of sequence, define: G is said to be (T, ϵ)-secure in the sense of indistinguishability if there is no algorithm (statistical test) with running time bounded by T that distinguish the sequence from truly random sequence with $P \geqslant 1/2 + \epsilon$.

$$(T, \epsilon/M) - \text{ next bit secure } \Rightarrow (T, \epsilon) - \text{indisting. secure}$$

Expected running time for the number fieeld sieve to factor n-bit Blum integer is,

$$L(n) \approx 2.8 \cdot 10^{-3} e^{1.9229 (n \ln 2)^{1/3} (\ln(n \ln 2))^{2/3}}.$$

Asymptotical bounds for BBS

- For $j = 1$ (n large) BBS generator is (T, ϵ)-secure in the sense of indistinguishability if,

$$T \leqslant \frac{L(n)(\epsilon/M)^2}{6n \log n} - \frac{2^7 n(\epsilon/M)^{-2} \log(8n(\epsilon/M)^{-1})}{\log n}$$

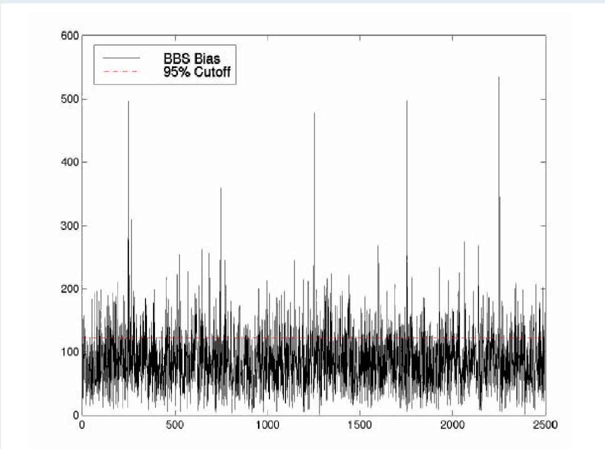
- For $j > 1$ BBS generator is (T, ϵ)-secure if,

$$T \leqslant \frac{L(n)}{36n(\log n)\delta^{-2}} - 2^{2j+9} n \delta^{-4}; \quad \delta = (2j-1)^{-1}(\epsilon/M).$$

n	$L(n)$	AB $j = 1$	AB $j = \log n$
1024	2^{78}	2^{-79}	-2^{-199}
2048	2^{108}	2^{-80}	-2^{-206}
3072	2^{130}	2^{-80}	-2^{-206}
7680	2^{195}	2^{115}	-2^{-213}
15360	2^{261}	2^{181}	-2^{-220}

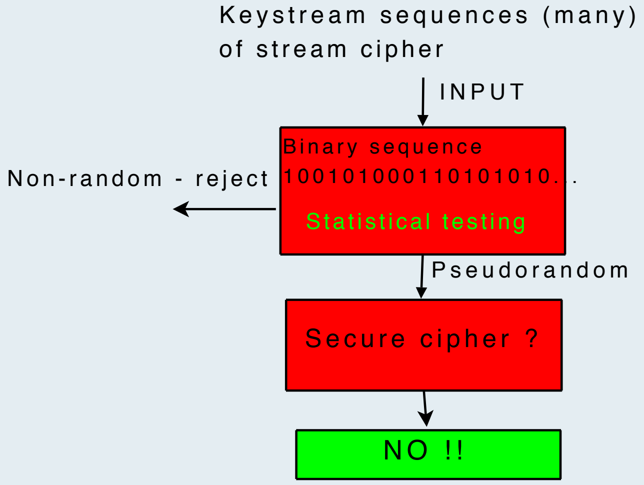
DFFT test on Blum-Blum-Shub generator

- Sequence of 5000 bits, from BBS. Every 10-th bit set to 1.
- At most 5% of peaks larger than 95% cutoff ($122 = \sqrt{3 \cdot 5000}$).



Statistical testing of stream ciphers

- CSPRNG property could be proved for BBS generator (intractability of factorization). For standard stream cipher schemes we cannot prove this.



Distinguishing attack

- An attack that distinguish the keystream sequence from a truly random sequence is called a *distinguishing attack*.
- If there is no distinction, then cipher acts as a One-time pad. Impossible to achieve with finite key length.
- In most cases, these attacks have no security implications on security of stream ciphers.
- In practice, distinguishing attacks on stream ciphers are usually impossible to mount - though reduced trust in cipher construction .

Distinguishing attack - some remarks

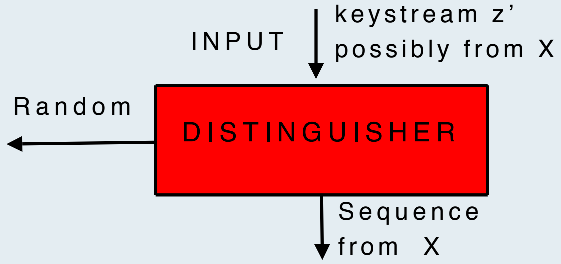
- For instance for the DES block cipher there is a straightforward distinguishing attack that needs 2^{32} blocks (words).
- This is due to the fact that DES belongs to the family of pseudo random permutations on 64 bits, and after 2^{32} encrypted blocks of data some block must be repeated (birthday paradox).
- However no information about the key is revealed.
- Sometimes distinguishing attacks may be turned into key recovary attacks.

Hypothesis testing

- Recognize a nonrandom behavior of keystream. Construct the cipher distribution P_C from the observed keystream sequence $z = z_1, z_2, \ldots$,

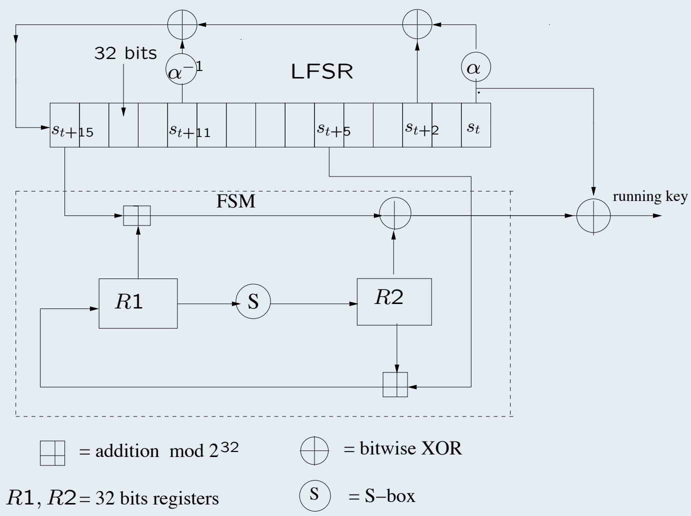
$$L_t(z) = \sum_{i=1}^{|I|} z_{t+I(i)} \quad \forall t,; \quad \text{I some index set}$$

- If approximations are "good" the samples from P_C are very noisy but not uniformly (randomly) distributed.



SNOW 2.0 - software oriented stream cipher

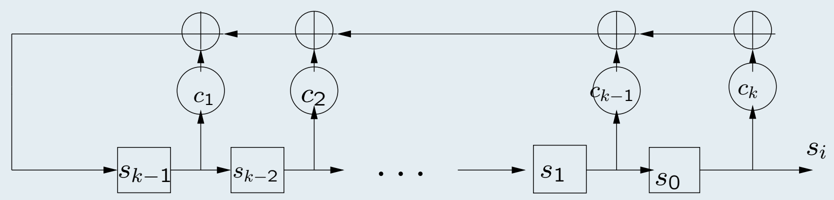
- Distinguishing attack 2^{177} bits of keystream and 2^{172} operations.



Design rationales - introduction

- No design strategies how to construct secure and fast stream cipher.
- An essential primitive is LFSR (Linear Feedback Shift Register):
  - fast in hardware and low hardware complexity
  - good statistical properties,
  - drawback - low linear complexity, relatively slow in software
- Other primitives include NFSR (Nonlinear FSR), FCSR, S-boxes, Boolean functions, addition $\pmod{2^n}$ etc.

Example - Linear Feedback Shift Registers (LFSR)



1. The content of stage 0 is output and forms a part of output sequence

2. The content of stage i is moved to stage $i-1$, for each $1 \leqslant i \leqslant k-1$.

3. The new content of stage $k-1$ is computed.

Pseudorandom properties of LFSR

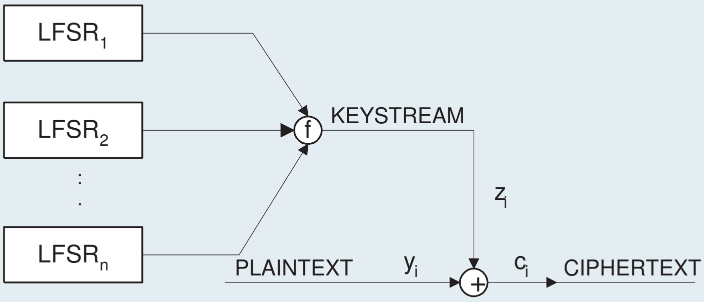
- LFSR is FSM so sequence is repeated after T (bits or blocks), i.e. $s_t = s_{t+T}, t \geqslant 0$.

- If $C(x) = 1 + c_1x + \cdots + c_kx^k \in \mathbb{F}_q[x]$ is *primitive* then the sequence is of *maximum length*, i.e. $T = q^k - 1$.

- A maximum length LFSR satisfies pseudorandom postulates but the problem is linear recursion. Given $2k$ output bits (blocks) one can recover the initial state using Berlekamp-Massey algorithm.

- To destroy nonlinearity one commonly applies a nonlinear filtering, e.g. a Boolean function.

Low hardware implementation using LFSR



Nonlinear combining generator

- Many requirements on the choice of Boolean function. Both for security reason and implementation (low number of gates).

Attacks on stream ciphers - preliminaries

- It is assumed that the encryption algorithm and the keystream sequence is known to the attacker (known-plaintext attack)
- Known-plaintext attack is a reasonable assumption, e.g. guess the ending or beginning of e-mail: Dear Sir, . . ., Sincerely yours.
- Having a huge ammount of keystream bits one can try (partial) key recovery attack or to distinguish the cipher from a truly random sequence ?
- Exhaustive key search tries each possible key and compares the resulting keystreams. The key of k binary bits gives 2^k operations.

Designers viewpoint

Speed - simple structure; Security - good confusion and diffusion

- Designers often oversee plausible cryptanalysis; "our design withstand current cryptanalysis ..."
- Security is often traded against speed (not intentionaly)
- Nonlinearity and pseudorandomness must be introduced in a clever way to acheive good performance and security.

General Shannon's attack

- Encyphering can be seen as $E = f(K, M)$.
- Given $M = m_1, m_2, \ldots, m_s$ and $E = c_1, c_2, \ldots, c_s$ cryptanalyst can set up equations for different key elements $k_1, k_2, \ldots, k_r$

$$\begin{aligned}
c_1 &= f_1(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r) \\
c_2 &= f_2(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r) \\
&\vdots \\
c_s &= f_s(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r).
\end{aligned}$$

- Each equation must be complex in k_i and involve many of them.

Shannon's attack applied to stream ciphers

- The scenario is usually known-plaintext attack. Thus, knowing c and m the keystream z is known and the system becomes:

$$\begin{aligned} z_1 &= f_1(s_1, s_2, \ldots, s_v) \\ z_2 &= f_2(s_1, s_2, \ldots, s_v) \\ &\vdots \\ z_l &= f_s(s_1, s_2, \ldots, s_v). \end{aligned}$$

- Each equation must be complicated equation in the secret state bits s_i and involve many of them.
- Closely related to algebraic attacks.

Generic attacks on stream ciphers

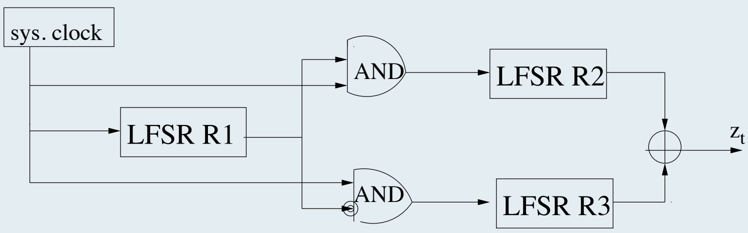
- Called generic since they are applicable to any stream cipher.
- Among others the most important ones are:

1. Correlation attacks
2. Algebraic attacks
3. Guess-and-determine attacks
4. Distinguishing attacks
5. Side channel attacks etc..

Guess and determine attacks

- Simple but can be very powerfull. Especially if the design is "bad".
- IDEA: Guess a part of internal state and try to determine the remaining key bits by observing the keystream.
- The guess is tested using some statistical method.

Guess and determine attacks - an example

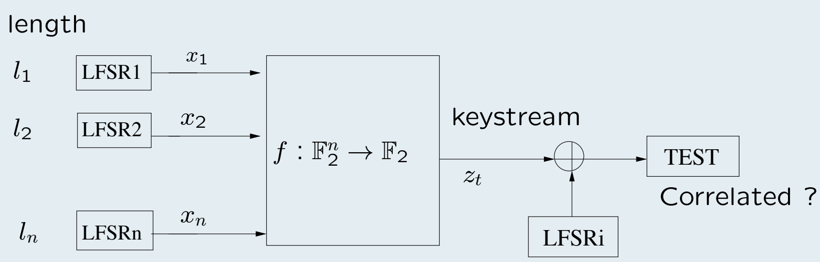


- Guess the content of LFSR R1.
- If the lengths of LFSR's are l_1, l_2, l_3 then observe $l_2 + l_3$ keystream bits. A linear system that can be easily solved. The time complexity is only 2^{l_1}.

Correlation attacks

- Correlation attacks are key recovary attacks , correlation between secret state/key bits and keystream.
- Especially applicable to LFSR-based stream ciphers .
- An eStream candidate, for a new stream cipher standard, ABC was successfully broken using correlation attacks.
- The attack might not be as obvious as in the following example.

Correlation attacks - an example



- Exhaustive search is performed trying $\prod_{i=1}^{n}(2^{l_i} - 1)$ different keys.
- Assume $Pb(x_i^t = z_t) = \frac{1}{2} + \varepsilon$. Try each state of LFSRi and measure the number of zeros in the XORed sequence. Complexity drops to

$$\sum_{i=1}^{n}(2^{l_i} - 1).$$

Defending correlation attacks

- Use function f which is resilient to this attack. This means that $Pb(x_i^t = z_t) = \frac{1}{2}$ for each i. No correlation between single LFSR's and the keystream.
- But similar attack may be performed by considering a pair of LFSR's, tripple of LFSR's ....
- A "good" cryptographic design of such functions will be treated in depth.

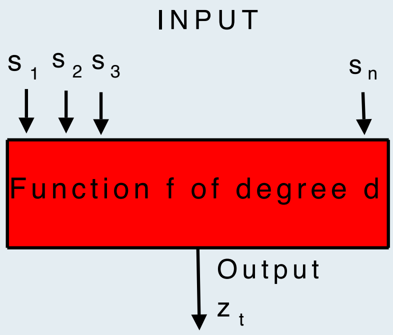
Algebraic attack on stream ciphers with linear transition

- Derived from Shannon's attack, lowering the degree of equations.
- Set up the enciphering equations:

$$\begin{aligned} z_0 &= f(k_0, k_1, \ldots, k_{n-1}) \\ z_1 &= f \circ L(k_0, k_1, \ldots, k_{n-1}) \\ &\vdots \\ z_t &= f \circ L^t(k_0, k_1, \ldots, k_{n-1}). \end{aligned}$$

- System of equations in n variables of degree $d = deg(f)$. The number of terms is $\leqslant \sum_{i=0}^{d} \binom{n}{i}$. Observe more than $\sum_{i=0}^{d} \binom{n}{i} \approx \frac{n^d}{d!}$ bits and solve in time $(\frac{n^d}{d!})^3$.

Algebraic attack - degree of equations



Output is then in the form:

$$f(s_1, s_2, \ldots, s_n) = s_1 s_2 \cdots s_d + s_2 s_3 \cdots s_{d+1} + \ldots + s_1 s_d + \ldots$$

Algebraic attacks- decreasing the degree of f

- Assume $d = deg(f) = 7$. Then if the key length is $k = 128$ bits and the size of internal state $n = 2k = 256$ then the time complexity is,

$$(\sum_{i=0}^{d} \binom{2k}{i})^3 \approx (\binom{256}{7})^3 = 2^{129}.$$

- Let now g s.t. $f(x)g(x) = 0$ and $deg(g) = 3$. Then if $z_i = 1$,

$$f(x) = z_i = 1 \Longrightarrow g(x) = 0.$$

- The time complexity becomes, $\left(\sum_{i=0}^{3} \binom{2k}{i}\right)^3 \approx \left(\binom{256}{3}\right)^3 = 2^{63}.$

Side channel attacks

- Utilize information leakage form other channels than keystream.
- Two examples are *power analysis* and *timing analysis*
- The power usage is measured for instance on a smart card when different operations are performed. More power for complicated operations.
- Timing attacks are similar but they measure execution time of various steps in algorithms.

Chapter 5

Pseudo-random Sequences

Content of this chapter:

- LFSR and generating functions.
- Design of periodic sequences.
- Berlekamp-Massey synthesis algorithm.
- Linear complexity of finite sequences.
- Applications of LFSR in stream ciphers.

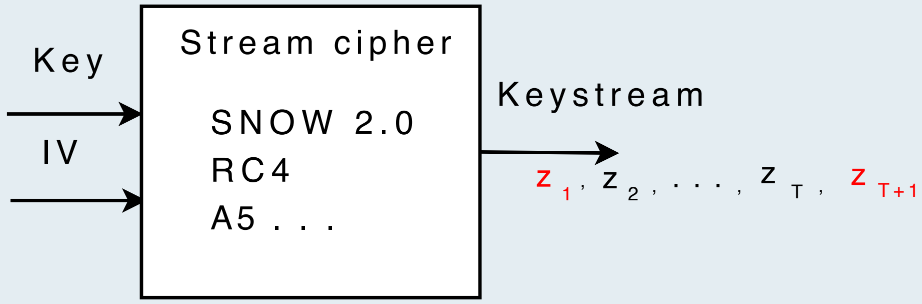
Pseudo-random sequences suitable for stream ciphers

- The main objective is to design cryptographically secure PRNG:
  - Keystream sequence should have large period
  - It should pass diverse statistical tests
- What are the suitable period and tests ?

Example: Assume encryption speed of 100MB/sec (realistic) and the period of sequence $T = 2^{32}$. A sequence repeat itself after only 5 seconds !

- For real-life applications the period should be at least 2^{60}

Repetition of keystream implies two-time pad



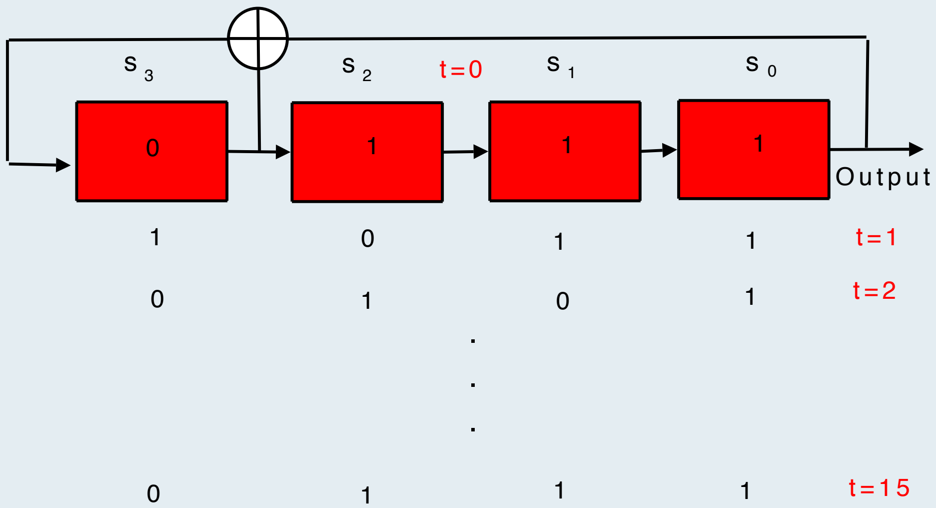
- Means that $z_{i+T} = z_i$ for all $i \geqslant 1$ thus,

$$c_i \oplus c_{i+T} = m_i \oplus z_i + m_{i+T} \oplus z_{i+T} = m_i \oplus m_{i+T}.$$

Ensuring the long period

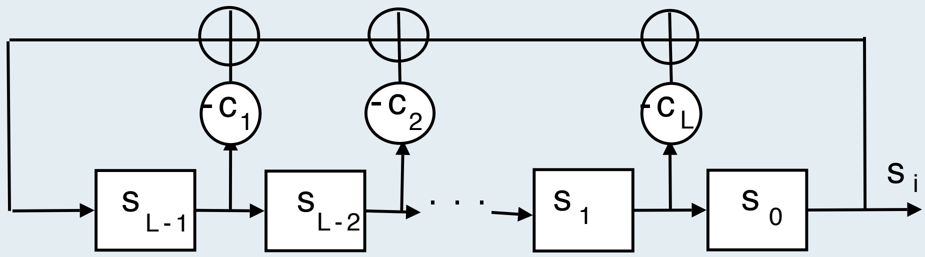
- It is desirable to obtain a lower bound on the period.
- How do we generate sequences of long period ?
- Common approach is to use simple finite state machines such as LFSR (Linear Feedback Shift Register)
- Given a number of stages L (length of LFSR) it can generate a maximum length sequences $2^L - 1$.

Linear Feedback Shift Registers (LFSR)



- The recurrence is $s_{t+4} = s_{t+3} + s_t, \ \ t \geqslant 0$.

LFSR - a general depiction



1. The content of stage 0 is output and forms a part of output sequence
2. The content of stage i is moved to stage $i-1$, for each $1 \leqslant i \leqslant L-1$.
3. The new content of stage $L-1$ is computed.

Some further notions related to LFSR-s

- If the initial content of stage i is $s_i \in GF(2^m)$ for each i, $0 \leqslant i \leqslant L-1$, then $[s_{L-1}, \ldots, s_1, s_0]$ is called *initial state* of LFSR.
- The polynomial $C(x) = 1 + c_1 x + \cdots + c_L x^L \in GF(2^m)[x]$ is called the *connection polynomial*.
- The output sequence $s = s_0, s_1, \ldots$ is uniquely determined via,

$$s_j = -(c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L}) \text{ for } j \geqslant L.$$

- The *state* at time t is $\mathbf{S}_t = (s_{t+L-1}, s_{t+L-2}, \ldots, s_t)$

Maximum length LFSR

- LFSR is FSM so sequence is repeated after T (bits or blocks), i.e. $s_t = s_{t+T}, t \geqslant 0$. Clearly, $1 \leqslant T \leqslant 2^L - 1$ as zero state cannot appear (why ?).
- In the previous example we get the sequence of maximum period $T = 2^4 - 1 = 15$.
- This is because we have chosen a primitive connection polynomial

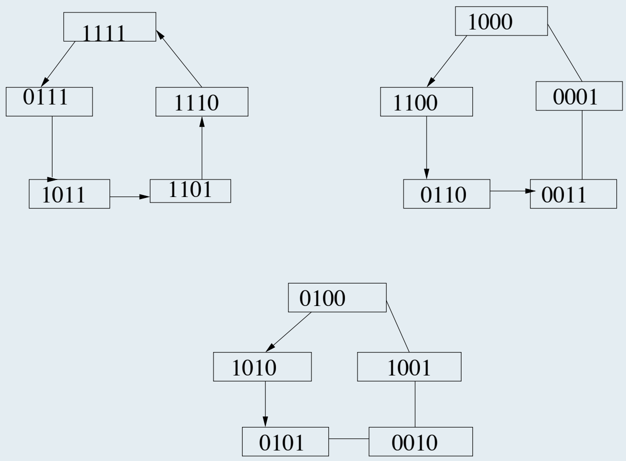
$$\begin{array}{rccc} C(x) = & 1+ & x+ & x^4 \\ & & \Updownarrow & \\ 0 = & s_{t+4}+ & s_{t+3}+ & s_t \end{array}$$

Example cont.

- In the previous example starting at $S = (0, 1, 1, 1)$ we have:

t	s_3	s_2	s_1	$s_0 = z_t$
0	0	1	1	1
1	1	0	1	1
2	0	1	0	1
3	1	0	1	0
4	1	1	0	1
5	0	1	1	0
6	0	0	1	1
7	1	0	0	1
8	0	1	0	0
9	0	0	1	0
10	0	0	0	1
11	1	0	0	0
12	1	1	0	0
13	1	1	1	0
14	1	1	1	1
15	0	1	1	1

Periodic cycles



Cycles for irreducible $C(x) = 1 + x + x^2 + x^3 + x^4$

Period of connection polynomial

- The **period** of $C(x) = \sum_{i=0}^{L} c_i x^i \in \mathbb{F}_q[x]$ is the least positive integer T s.t. $C(x)|x^T - 1$.
- Do long division of $\frac{1}{C(x)}$ until the rest is x^T; i.e. $\frac{1}{C(X)} = Q(x) + \frac{x^T}{C(x)}$.
- E.g. if $C(x) = 1 + x + x^2 + x^3 + x^4 \in \mathbb{F}_2[x]$ we have,

  $1 = (1+x)\cdot(1+x+x^2+x^3+x^4)+x^5 \Rightarrow 1+x^5 = (1+x)\cdot(1+x+x^2+x^3+x^4)$
- Thus the period of $C(x)$ is $T = 5$.

Generating functions

- We need a concept of generating functions .
- We say the sequence is causal if it starts at $t = 0$. A periodic sequence (of period T) is given as

$$s = s_0, s_1, \ldots, s_{T-1}, s_0 s_1 \ldots = [s_0, s_1, \ldots, s_{T-1}]^{\infty}.$$

- Let us represent s as a polynomial,

$$S(x) = s_0 + s_1 x + s_2 x^2 + \cdots = \sum_{k=0}^{\infty} s_k x^k.$$

Generating functions

- Then for given $S(x)$ and $C(x)$ we have,

$$\begin{aligned} S(x)C(x) &= \sum_{k=0}^{\infty} s_k x^k \sum_{i=0}^{L} c_i x^i = \sum_{k=0}^{\infty} \sum_{i=0}^{L} s_k c_i x^{k+i} = \quad j \leftarrow k+i \\ &= \sum_{j=0}^{\infty} [\sum_{i=0}^{L} c_i s_{j-i}] x^j. \end{aligned}$$

- But we also have $\sum_{i=0}^{L} c_i s_{j-i} = 0$ for $j \geqslant L$, hence

$$S(x)C(x) = \sum_{j=0}^{L-1} [\sum_{i=0}^{L} c_i s_{j-i}] x^j = P(x),$$

where $P(x) = p_0 + p_1 x + \cdots p_{L-1} x^{L-1}$ and

$$p_j = \sum_{i=0}^{L} c_i s_{j-i} = \sum_{i=0}^{j} c_i s_{j-i}, \quad j = 0, 1, \ldots, L-1.$$

Generating function of periodic sequence

- Thus any sequence from LFSR has the transform,

$$S(x) = \frac{P(x)}{C(x)}, \quad deg(P) < deg(C).$$

- Also $\underbrace{[1,0,0,\ldots,0]}_{T positions}^{\infty} \rightarrow 1 + x^T + x^{2T} + \cdots = \frac{1}{1-x^T}$.
- $\underbrace{[0,1,0,\ldots,0]}_{T positions}^{\infty} \rightarrow x + x^{T+1}x^{2T+1} + \cdots = \frac{x}{1-x^T}$
- Thus, the generating function of periodic sequence $[s_0, s_1, \ldots, s_{T-1}]^{\infty}$ is,

$$S(x) = \frac{s_0 + s_1 x + \cdots + s_{T-1}x^{T-1}}{1 - x^T}; \quad s_i \in \mathbb{F}_q.$$

Uniqueness of representation

- The coefficients p_j of $P(x)$ are expressed as,

$$p_j = \sum_{i=0}^{j} c_i s_{j-i}, \quad j = 0, 1, \ldots, L-1.$$

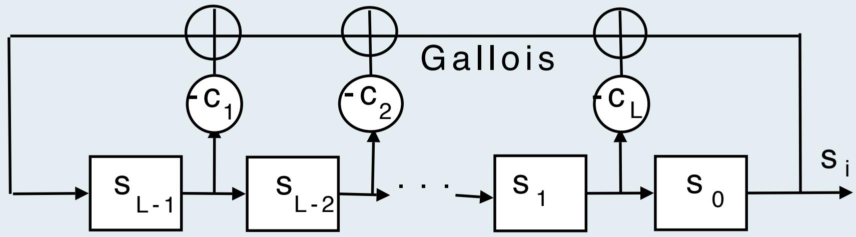
$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{L-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ c_1 & 1 & \cdots & 0 \\ & & \vdots & \\ c_{L-1} & c_{L-2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{L-1} \end{pmatrix}.$$

- Nonsingular matrix, unique solution for initial state given $[p_{L-1}, \ldots, p_0]$.

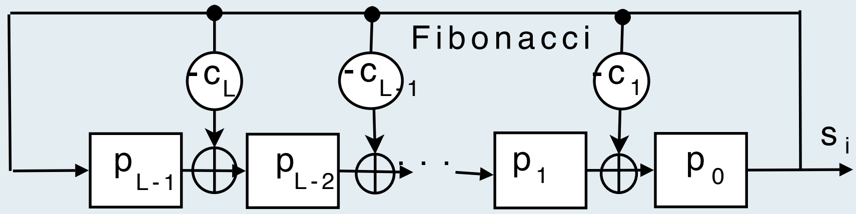
Theorem: Given LFSR and $C(x)$ the set of all possible sequences that can be generated is the set of sequences with gen. function

$$S(x) = \frac{P(x)}{C(x)}, \quad deg(P) < deg(C).$$

Fibonacci versus Galois model



- The two models are equivalent. Galois takes $[s_{L-1},\ldots,s_1,s_0]$ as initial state and Fibonacci $[p_{L-1},\ldots,p_1,p_0]$.



Equivalence of Fibonacci and Galois model

The same recursion is valid for Fibonacci model,

$$s_j = -(c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_L s_{j-L}) \text{ for } j \geqslant L.$$

- Do we really get $s_0,\ldots,s_{L-1}$ using initial state $[p_{L-1},\ldots,p_1,p_0]$?

$$\begin{aligned} p_0 &= s_0 \\ p_1 &= s_1 + c_1 s_0 \\ p_2 &= s_2 + c_1 s_1 + c_2 s_0 \\ &\vdots \\ p_{L-1} &= s_{L-1} + c_1 s_{L-2} + \cdots + c_{L-1} s_0 \end{aligned}$$

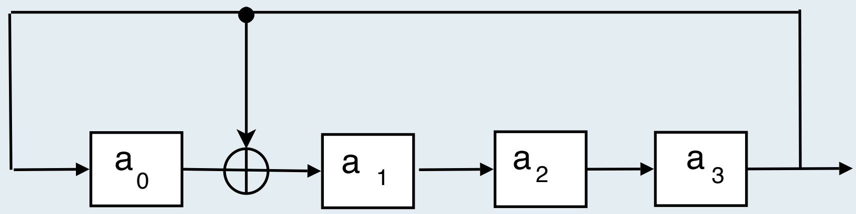
We compute outputs as follows,

$$\begin{aligned} z_0 &= p_0 = s_0 \\ z_1 &= -c_1 s_0 + p_1 = -c_1 s_0 + s_1 + c_1 s_0 = s_1 \\ &\vdots \end{aligned}$$

Some other applications of LFSR

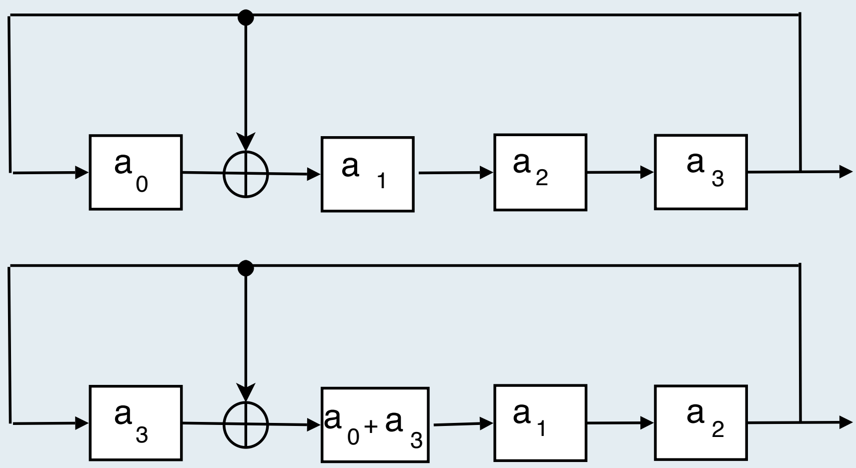
- Easy to implement multiplication in the field with Fibonacci model.

▶ Assume we want to multiply $\beta = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3$ with α in the $GF(2^4)$, α primitive element, the root of $f(x) = x^4 + x + 1$.



▶ Since $\beta = \alpha^i$ the content of LFSR will be $\alpha^i, \alpha^{i+1}, \alpha^{i+2}, \ldots$.

Multiplication with α cont.



- Exactly what we need as (using $\alpha^4 = \alpha + 1$),

$$\alpha(a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0) = a_2\alpha^3 + a_1\alpha^2 + (a_0 + a_3)\alpha + a_3.$$

Period of sequence

Theorem: If $\gcd(P(x), C(x)) = 1$ then the period of $C(x)$ is the **same** as period of

$$S(x) = \frac{P(x)}{C(x)}, \quad deg(P) < deg(C)$$

Proof: Assume period of $C(x)$ is T and period of $S(x)$ is T'. So there is $Q(x) \in \mathbb{F}_q[x]$ s.t. $C(x)Q(x) = 1 - x^T$. So,

$$S(x) = \frac{P(x)}{C(x)} = \frac{P(x)Q(x)}{C(x)Q(x)} = \frac{s_0 + s_1 x + \cdots + s_{T-1}x^{T-1}}{1 - x^T},$$

for some $(s_0, \ldots, s_{T-1})$. Hence $T' \leqslant T$.

Period of sequence cont'd

Proof: But also, $s = [s_0, s_1, \ldots, s_{T'-1}]^\infty$ so we have,

$$S(x) = \frac{P(x)}{C(x)} = \frac{s_0 + s_1 x + \cdots + s_{T'-1}x^{T'-1}}{1 - x^{T'}},$$

which can be written as,

$$(1 - x^{T'})P(x) = (s_0 + s_1 x + \cdots + s_{T'-1}x^{T'-1})C(x).$$

But as $\gcd(P(x), C(x)) = 1$ then $C(x) | 1 - x^{T'}$. Since the period of $C(x)$ is T it must be that $T \leqslant T'$ so $T = T'$.

Primitive versus irreducible connection polynomial

- Thus irreducible $C(x)$ implies $\gcd(C, P) = 1$ so the period of S is actually *the order* of $C(x)$, i.e. the least $e \leqslant q^k - 1$ such that $C(x) | x^e - 1$. We recall a result from finite field theory:

Theorem: A polynomial $C \in \mathbb{F}_q[x]$ of degree k is primitive if and only if $\operatorname{ord}(C) = q^k - 1$.

- Thus, only primitive polynomials have the largest possible period.

Golomb postulates

- Number of zeros and ones in the sequence, $\#1 = \#0 \pm 1$
- A run of length k is $(1, \ldots, 1)$ (or $(0, \ldots, 0)$). Half the runs have length 1, a quater have length 2 etc.
- The autocorrelation function of a binary periodic sequence defined by

$$r(\tau) = \frac{1}{T} \sum_{i=0}^{T-1} (-1)^{s_n + s_{n+\tau}},$$

should be small.

- All above satisfied by maximum-length LFSR sequence.

Statistical properties - example

- In our example $C(x) = 1 + x + x^4$ and the sequence was:

$$s = 111010110010001|11101\cdots$$

  - Number of ones is 8, and nmb. of zeros 7.
  - Number of runs is 8. For instance there are two runs of length 2, in red colour.
  - For any $\tau \neq 0$ we get $r(\tau) = \pm 1$.
- Knowing previous values in the sequence does not help in deducing current value !

Period of $s_1 + s_2$

Theorem: Let s_1 and s_2 be periodic sequences with

$$S_1(x) = \frac{P_1(x)}{C_1(x)} \text{ and } S_2(x) = \frac{P_2(x)}{C_2(x)}$$

so that

$$\gcd(P_1(x), C_1(x)) = \gcd(P_2(x), C_2(x)) = \gcd(C_1(x), C_2(x)) = 1.$$

Then the period of $s = s_1 + s_2$ is $T = \mathrm{lcm}(T_1, T_2)$.

Proof: Let $\lambda = \mathrm{lcm}(T_1, T_2)$. Clearly $s_1^{i+\lambda} = s_1^i$ and $s_2^{i+\lambda} = s_2^i$ for all $i \geqslant 0$. Therefore,

$$s^{i+\lambda} = s_1^{i+\lambda} + s_2^{i+\lambda} = s_1^i + s_2^i = s_i \quad i \geqslant 0,$$

thus $T \leqslant \lambda$.

Period of $s_1 + s_2$ cont'd

Proof: Further we have,

$$S(x) = S_1(x) + S_2(x) = \frac{P_1(x)C_2(x) + P_2(x)C_1(x)}{C_1(x)C_2(x)} = \frac{P(x)}{C(x)}.$$

The condition on relative primality gives $\gcd(P, C) = 1$. The period of s is the same as period of $C(x)$, and we must have,

$$C(x) = C_1(x)C_2(x)|x^T - 1 \Longrightarrow \ C_1(x)|x^T - 1 \ \wedge \ C_2(x)|x^T - 1.$$

This means,

$$T_1 \mid T \quad T_2 \mid T \Longrightarrow T \geqslant \mathrm{lcm}(T_1, T_2) = \lambda.$$

Therefore, $T = \lambda$.

Linear complexity of sequence

- Let $s = s_0, s_1, \ldots$ be an infinite sequence over $\mathbb{F}_q$. **Linear complexity** is the length of the shortest LFSR that generates s.
- Sequence s of linear complexity L and t finite subsequence of length at least $2L$.
  - If L unknown recover LFSR by Berlekamp-Massey.
  - If L known either BM or direct linear algebra.
- Known plaintext attack on a stream cipher based purely on LFSR is easily performed with knowledge of $2L$ consecutive bits (blocks).

LFSR synthesis

- Let $s^N = s_0, s_1, \ldots, s_{N-1}$ denote the first N symbols of the sequence $s = s_0, s_1, \ldots$.
- The main problem is to find the shortest LFSR that generates these N symbols.
- Trivially, any LFSR of length $L \geqslant N$ can be used as we can assign the initial state of LFSR with s^N.
- We assume $L < N$, and also we allow that $deg(C(x)) \leqslant L$. Hence LFSR is specified by $(C(x), L)$.

Deciding whether LFSR can generate given sequence

- Since $L < N$ check if $s_L, \ldots, s_{N-1}$ satisfies LFSR equation, that is,
$$\sum_{i=0}^{L} c_i s_{j-i} = 0 \text{ for } j = L, L+1, \ldots, N-1.$$
- Thus if $(C(x), L)$ can generate s^N we have to check for s^{N+1},
- Define $d = s_N - \hat{s}_N = s_N - \sum_{i=1}^{L}(-c_i)s_{N-i}$, where s_N is the $(N+1)$-th bit of s and $\hat{s}$ is generated by $(C(x), L)$.
- Then $(C(x), L)$ can generate s^{N+1} iff $d = 0$. But if $d \neq 0$ we have to find another $(C^*(x), L^*)$ that generates s^{N+1}.

Massey's lemma

Lemma: If $(C(x), L)$ can generate s^N but not s^{N+1}, which can be generated by $(C^*(x), L^*)$ then,

$$L^* > N - L \quad \text{or} \quad L_{N+1} \geqslant N + 1 - L_N.$$

Example: The LFSR of length 2 and $C(x) = 1 + x^2$ can generate

$s^9 = 1, 0, 1, 0, 1, 0, 1, 0, 1$ but not $s^{10} = 1, 0, 1, 0, 1, 0, 1, 0, 1, 1$.

Recurence is $s_i = s_{i-2}$ for $i \geqslant 2$

Then any LFSR which can generate s^{10} has the length

$$L^* > N - L = 9 - 2 = 7.$$

Massey's theorem

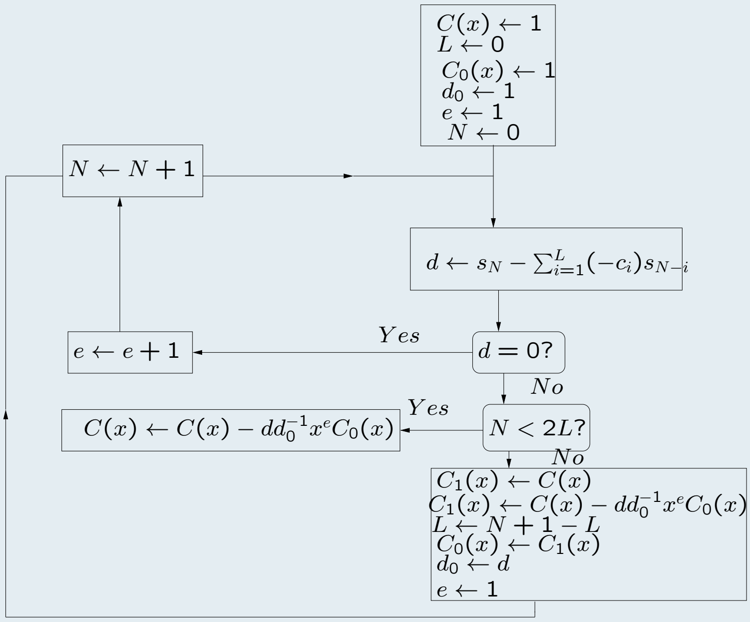
- L_N is nondecreasing function, so Massey's Lemma gives $L_{N+1} \geqslant \max[L_N, N + 1 - L_N]$.

  Theorem: If an LFSR with $C(x)$ can generate S^N and is of length L_N then,

  $$L_{N+1} = \begin{cases} L_N, & d_N = 0; \\ \max(L_N, N + 1 - L_N), & d_N \neq 0. \end{cases}$$

- This result implies that $L_{N+1} > L_N$ if and only if $N \geqslant 2L_N$.
- For a sequence with $LC = L_N$ we need $2L_N$ bits to recover the LFSR.

The Berlekamp-Massey algorithm



The Berlekamp-Massey algorithm -example

LFSR synthesis for $s = 1,0,0,1,1,1,0,1$ looks like:

s_N	d	$C_1(x)$	C (x)	L	Shift register	$C_0(x)$	d_0	e	N
-	-	-	1	0		1	1	1	0
1	1	1	1 + x	1		1	1	1	1
0	1	"	1	"	0	"	"	2	2
0	0	"	"	"	"	"	"	3	3
1	1	1	$1+x^3$	3		1	1	1	4
1	1	"	$1+x+x^3$	"		"	"	2	5
1	0	"	"	"	"	"	"	3	6
0	0	"	"	"	"	"	"	4	7
1	0	"	"	"	"	"	"	5	8

Known plaintext attack on LFSR-based stream ciphers

- Assume we use LFSR of length L and $C(x)$ as a connection polynomial.
  - The knowledge of $2L$ consecutive bits of m and c gives

$$\begin{aligned}&m_k, m_{k+1}, \ldots, m_{k+2L-1}\\&c_k, c_{k+1}, \ldots, c_{k+2L-1}\\&z_k, z_{k+1}, \ldots, z_{k+2L-1}, \quad z_i = m_i + c_i.\end{aligned}$$

- Berlekamp-Massey algorithm returns L and $C(x)$.
- Feed the LFSR found by BM and generate the remainder of the sequence.

Linear complexity of LFSR

- Linear complexity of infinite binary sequence s is defined as the shortest linear recurrence that generates s; such was $s_{t+4} = s_{t+3} + s_t$.
- The definition coincides with LFSR's structure; linear complexity of LFSR of length L is L.
- Berlekamp-Massey algorithm has running time $\mathcal{O}(n^2)$ applied to a sequence of length n.
- Implies that linear complexity $> 2^{30}$ for practical applications.

Linear complexity - example

- High linear complexity is necessary but not sufficient requirement

Example: Every sequence of period T satisfies $s_{i+T} = s_i$ for all $i \geqslant 0$. Let s^T be infinite sequence of period T,

$$s_T = 1\underbrace{00\cdots0}_{T-1}1\underbrace{00\cdots0}_{T-1}\dots$$

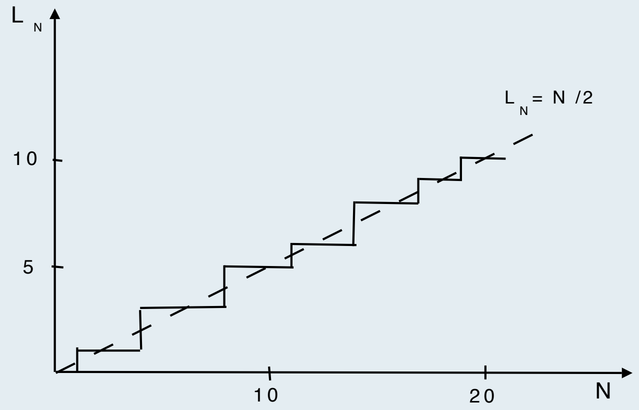
- The linear complexity is T as there is no linear relation shorter than $s_{i+T} = s_i$.
- Clearly the sequence is completely useless for cryptographic use. Apply several tests , e.g. sequence does not pass Golomb's tests !

Linear complexity profile

- Given a binary sequence $s = s_0, s_1, s_2, \dots$, denote $s^N = s_0, s_1, \dots, s_{N-1}$ and L_N its corresponding linear complexity.
- Linear complexity profile is calculated for each new bit added to the sequence, starting from the first bit. Profile is then plotted as a function of sequnce's length.
- It was established that linear complexity profile for perfectly random source closely follows the line $y = \frac{x}{2}(y = L_N, x = N)$.
- Good pseudo-random sequence should have linear complexity $\approx N/2$ for its first N bits.

Linear complexity profile - example

For a periodic $s^{20} = 10010011110001001110$ we may plot:



LCP of a non-random sequence s defined as $s_i = 0$ unless $i = 2^j - 1$ for $j \geqslant 0$, also follows the line $L_N = N/2$ closely !

Increasing the linear complexity of sequence

- What is the linear complexity of $S_1 + S_2$ and of $S_1 S_2$?
- Linear complexity increases for a proper choice of connection polynomials and length of LFSRs !
- Enough to prove the cases $S_1 + S_2$ and $S_1 S_2$, the rest by induction easily, $S_1 S_2 S_3 = (S_1 S_2) S_3$!

Linear complexity of $S_1 + S_2$

- Let $S_1(x) = \frac{P_1(x)}{C_1(x)}$ and $S_2(x) = \frac{P_2(x)}{C_2(x)}$. Then,

$$S_1(x) + S_2(x) = \frac{P_1(x)C_2(x) + P_2(x)C_1(x)}{C_1(x)C_2(x)} = \frac{P(x)}{\text{mcm}(C_1(x)C_2(x))}$$

- Hence $L(S_1 + S_2) \leqslant deg(\text{mcm}(C_1, C_2))$. We also have

$$deg(\text{mcm}(C_1, C_2)) \leqslant deg(C_1) + deg(C_2) - deg(\text{gcd}(C_1, C_2)),$$

so if $\text{gcd}(C_1, C_2) = 1$ then $L(S_1 + S_2) \leqslant deg(C_1) + deg(C_2)$.

Linear complexity of $S_1 + S_2$ cont.

- Then $L(S_1 + S_2) = deg(C_1) + deg(C_2)$ exactly when $\text{gcd}(C_1, C_2) = 1$, assuming $\text{gcd}(P_1, C_1) = \text{gcd}(P_2, C_2) = 1$.

- Assume $Q|C_1C_2$ in,

$$S_1(x) + S_2(x) = \frac{P_1(x)C_2(x) + P_2(x)C_1(x)}{C_1(x)C_2(x)},$$

then either $Q|C_1$ or $Q|C_2$ as $\text{gcd}(C_1, C_2) = 1$.

- So let $Q|C_1$. Then if $Q|P_1C_2 + P_2C_1$ it must divide P_1C_2, a contradiction, as $Q \not| P_1$.

Linear complexity of S_1S_2

- Note that we do not consider $S_1(x)S_2(x)$ - not memoryless. Our sequence is $s_{1,0}s_{2,0}, s_{1,1}s_{2,1}, s_{1,2}s_{2,2}\cdots$

- We assume (for simplicity) that $C_1(x)$ and $C_2(x)$ are primitive over $\mathbb{F}_q$ of coprime degree n_1 resp. n_2. The LFSR recursion is,

$$s_j^{(i)} = -(c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_{n_i} s_{j-n_i}) \text{ for } j \geqslant k, \quad i = 1, 2.$$

- Recall that $C^i(x) = 1 + c_1 x + \ldots c_{n_i} x^{n_i}$.

- We treat sequences using finite field theory: now look at the characteristic polynomial (Galois $\rightarrow$Fibonacci) $C^i(x) \leftarrow x^{n_i} C^i(1/x) = c_{n_i} + c_{n_i - 1} x + \ldots + x^{n_i}$.

The linear complexity of S_1S_2 cont.

- Both polynomials splits into linear factors over $\mathbb{F}_{q^{n_1 n_2}}$,

$$C_1(x) = \prod_{j=1}^{n_1} (x - \alpha_j), \qquad C_2(x) = \prod_{i=1}^{n_2} (x - \beta_i); \qquad \alpha_j, \beta_i \in \mathbb{F}_{q^{n_1 n_2}}.$$

  Roots of C_1 are in $\mathbb{F}_{q^{n_1}}$ given by $\alpha, \alpha^q, \ldots, \alpha^{q^{n_1 - 1}}$ and roots of C_2 are in $\mathbb{F}_{q^{n_2}}$, $\beta, \beta^q, \ldots, \beta^{q^{n_2 - 1}}$.

- Then Selmer proved that $C(x) = \prod_{i,j}(x - \alpha_j \beta_i)$ is a degree $n = n_1 n_2$ primitive polynomial over $\mathbb{F}_q$!!!

Linear complexity of $S_1 S_2$

- **Idea :** Instead of time recursion, the sequence is given as linear combinations of the roots of characteristic polynomial !

Theorem: Assume C irreducible with roots $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{q^n}$, then,

$$s_i = \sum_{j=1}^{n} \beta_j \alpha_j^i, \quad i = 0, 1, \ldots,$$

where $\beta_1, \ldots, \beta_n$ are uniquely determined by s_i and are in the splitting field of $C(x)$ over $\mathbb{F}_q$.

- But also $C(x) = c_n + c_{n-1}x + \ldots + x^n$ implies the time recursion,

$$c_n s_{i+n} + c_{n-1} s_{i+n-1} + \ldots + s_i = 0$$

Linear complexity of $S_1 S_2$

Proof: $\beta_1, \ldots, \beta_n$ are obtained from the system of linear equations,

$$s_i = \sum_{j=1}^{n} \beta_j \alpha_j^i, \quad i = 0, 1, \ldots,$$

Vandermonde determinant $\neq 0$, so $\beta_1, \ldots, \beta_n$ unique in the splitting field of $C(x)$ over $\mathbb{F}_q$.

Now we check that $\sum_{j=1}^{n} \beta_j \alpha_j^i$ satisfy the time recurence above,

$$c_n \sum_{j=1}^{n} \beta_j \alpha_j^{i+n} + c_{n-1} \sum_{j=1}^{n} \beta_j \alpha_j^{i+n-1} + \ldots + \sum_{j=1}^{n} \beta_j \alpha_j^i = \sum_{j=1}^{n} \beta_j C(\alpha_j) \alpha_j^i = 0.$$

Summary on combining sequences

- We have actually proved that $L(S_1S_2) = deg(C) = n_1n_2$.
- Combining several LFSR-s $L_1, \ldots, L_n$ (primitive connection polynomials of cooprime degree) we get,
  1. Linear complexity of Boolean function applied to LFSR's sequences $S_1, \ldots, S_n$ is
  $$L(f(S_1, \ldots, S_n)) = f(S_1, \ldots, S_n).$$
  2. Period is equal to $\prod_{i=1}^{n}(2^{L_i} - 1)$.

Increasing linear complexity - example

Example: Take $L = 128$ be a length of LFSR, thus given 256 output bits one reconstruct initial state.

Now given 8 maximum-length LFSR of co-prime lengths

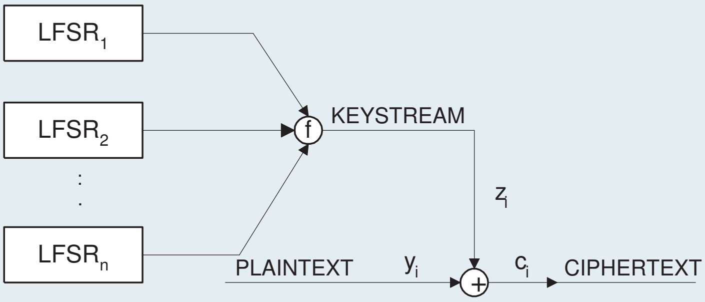
$$7, 9, 11, 13, 17, 19, 23, 29 \Rightarrow \sum L_i = 128$$

linear complexity of the sequence $S_1S_2 \cdots S_8 + S_2S_3 \cdots S_7$ is

$$LC = \prod_{i=1}^{8} L_i + \prod_{i=2}^{7} L_i \approx 2^{30}$$

BM algorithm needs: $2 \cdot 2^{30}$ keystream bits and runs in complexity 2^{60}.

Nonlinear combiner



- Period of length $\prod_{i=1}^{n}(2^{L_i}-1)$.
- Linear complexity is evaluation of Boolean function over integers !

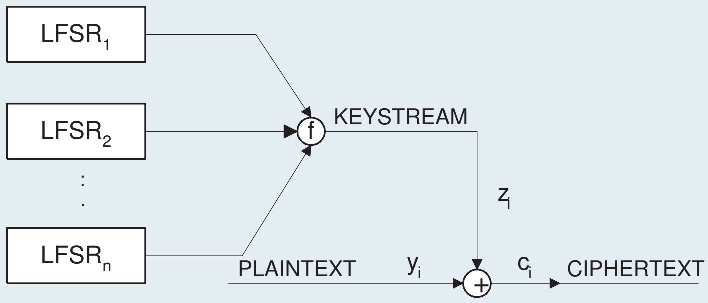
Chapter 6

Nonlinear Combiners and Boolean functions

Content of this chapter:

- Nonlinear combiners (repetition).
- Introduction to Boolean functions.
- Correlation attacks and correlation immunity.
- Nonlinear filtering generators.
- Feedback Carry Shift Register (FCSR) and stream ciphers.

Nonlinear combiner



- Period of length $\prod_{i=1}^{n}(2^{L_i}-1)$.
- Linear complexity is evaluation of Boolean function over integers !

Increasing linear complexity - example

Example: Take $L = 128$ be a length of LFSR, thus given 256 output bits one reconstruct initial state.

Now given 8 maximum-length LFSR of co-prime lengths

$$7, 9, 11, 13, 17, 19, 23, 29 \Rightarrow \sum L_i = 128$$

linear complexity of the sequence $S_1S_2\cdots S_8 + S_2S_3\cdots S_7$ is

$$LC = \prod_{i=1}^{8} L_i + \prod_{i=2}^{7} L_i \approx 2^{30}$$

BM algorithm needs: $2 \cdot 2^{30}$ keystream bits and runs in complexity 2^{60}.

Nonlinear combiners - susceptibility to correlation attacks

- Evaluation over integers due to previous results, meaning,

$$f(x_1, \ldots, x_n) = \bigoplus_a c_a x^a \rightarrow LC = \sum_a c_a x^a,$$

where x^a means $x_1^{a_1} \cdots x_n^{a_n}$.

- Problem is correlation attacks which are completely defended only if combining function is linear.

- Solution is e.g. combiners with memory (summation generators), E_0 in Bluetooth.

Implementation complexity vs. linear complexity

- Note that a fixed total length of LFSRs gives different LC depending on n (exercise)

- It turns out that increasing linear complexity implies increased complexity of implementation (quite intuitive) !

- Nonlinear combiner aims for efficient implementation thus too large n is not acceptable.

Boolean functions definitions

- Boolean functions map n binary inputs to a single binary output.
- More formaly $f : \mathbb{F}_2^n \to \mathbb{F}_2$ maps ($\mathbb{F}_2^n = GF(2)^n$)

$$(x_1, \dots, x_n) \in \mathbb{F}_2^n \mapsto f(x) \in \mathbb{F}_2$$

- Since $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is a mapping it can be represented as a polynomial in the ring $\mathbb{F}_2[x_1, \dots, x_n]/ < x_1^2 = x_1, \dots, x_n^2 = x_n >$.
- This ring is simply a set of all polynomials with binary coefficients in n indeterminates with property that $x_i^2 = x_i$.

Boolean functions-definitions II

- This may be formalized further by defining,

$$f(x) = \sum_{c \in \mathbb{F}_2^n} a_c x^c = \sum_{c \in \mathbb{F}_2^n} a_c x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}, \quad c = (c_1, \dots, c_n)$$

- Thus f is specified by the coefficients a_c
- There are 2^n different terms $x_1^{c_1} x_2^{c_2} \cdots x_n^{c_n}$ for different c's. As a_c is binary it gives 2^{2^n} different functions in n variables $x_1, \dots, x_n$.
- For $n = 3$ there are $2^8 = 256$ distinct functions specified by a_c,

$$B_3 = \{a_0 1 \oplus a_1 x_1 \oplus a_2 x_2 \oplus a_3 x_3 \oplus a_4 x_1 x_2 \oplus a_5 x_1 x_3 \oplus a_6 x_2 x_3 \oplus a_7 x_1 x_2 x_3\}$$

Boolean functions- Algebraic normal form

- We usualy skip $\oplus$ notation and use $+$.
- Let us specify the function $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ as,

$$f(x_1, x_2, x_3) = x_3 + x_1x_2 + x_2x_3.$$

- That is, $a_{(001)} = 1 \to x_3$,$a_{(110)} = 1 \to x_1x_2$, $a_{(011)} = 1 \to x_2x_3$, or w.r.t. definition of $\mathcal{B}_3$

$$a_0 = 0,\ \ a_1 = 0,\ \ a_2 = 0,\ \ a_3 = 1,\ \ a_4 = 1,\ \ a_5 = 0,\ \ a_6 = 1,\ \ a_7 = 0.$$

Truth table -Example

Definition: The *truth table* of f is the evaluation of function for all possible inputs.

x_3	x_2	x_1	$f(x)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

The truth table of the Boolean function $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3$.

Truth table and ANF correspondence

- From ANF to truth table is easy. In other direction it can be verified that,

$$f(x) = \sum_{\alpha | f(\alpha)=1} \prod_{i=1}^{n} (1 + x_i + \alpha_i), \quad \alpha \in \mathbb{F}_2^n.$$

- For the previous example we have

$$f(\alpha) = 1 \Leftrightarrow (\alpha_1, \alpha_2, \alpha_3) \in \{(1,1,0), (0,0,1), (1,0,1), (1,1,1)\}$$

- Then

$$\begin{aligned} f(x) &= x_1x_2(1+x_3) + (1+x_1)(1+x_2)x_3 + x_1(1+x_2)x_3 + \\ & \quad x_1x_2x_3 = \ldots = x_1x_2 + x_2x_3 + x_3. \end{aligned}$$

Affine and nonlinear functions

- ANF is also recovered through

$$a_u = \sum_{\alpha \in \mathbb{F}_2^n | \alpha \preceq u} f(\alpha) \pmod 2.$$

Definition: The set of all Boolean functions in n variables denoted $\mathcal{B}_n$.

Definition: Functions of degree at most one are called *affine*.

$$\mathcal{A}_n = \{a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n; a_i \in \mathbb{F}_2, 0 \leqslant i \leqslant n\}.$$

An affine function with $a_0 = 0$ is said to be *linear*. The set of all n variable linear functions is denoted by $\mathcal{L}_n$.

Linear complexity versus period

- In our example $f(x) = x_3 + x_1x_2 + x_2x_3$ so function is balanced ; equal number of 0s and 1s, that is $\#f(x) = 1 = \#f(x) = 0$.
- Combining 3 LFSRs of lengths L_1, L_2, L_3 we get
  - Period: $T = (2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$.
  - Lin. Complexity: $LC = L_3 + L_1L_2 + L_2L_3$
- To increase the linear complexity we may choose $f(x) = x_1x_2x_3$,
  - Period: $T = (2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$.
  - Lin. Complexity: $LC = L_1L_2L_3$
- Output sequence is nonbalanced.

Highest algebraic degree and balancedness

- Easy to show that function containing degree n term is not balanced:

$$f(x) = \sum_{\alpha | f(\alpha)=1} \prod_{i=1}^{n} (1 + x_i + \alpha_i), \quad \alpha \in \mathbb{F}_2^n.$$

- If f contains $x_1x_2 \cdots x_n$ in its ANF then it has an odd number of 1s in its truth table, i.e. f not balanced !
- Now f is not balanced, f is zero unless $x = (111)$; output sequence is nonbalanced as well. Let $f(x_1, x_2) = x_1x_2$ with $L_1 = 2$, $L_2 = 3$

$$\begin{aligned} s_1 = x_1 &= 101101101101101101101101; & s_{t+2} = s_{t+1} + s_t \\ s_2 = x_2 &= 010011101001110100111010; & s_{t+3} = s_{t+2} + s_t \\ z_t = x_1x_2 &= 000001101001100100100|000 & \end{aligned}$$

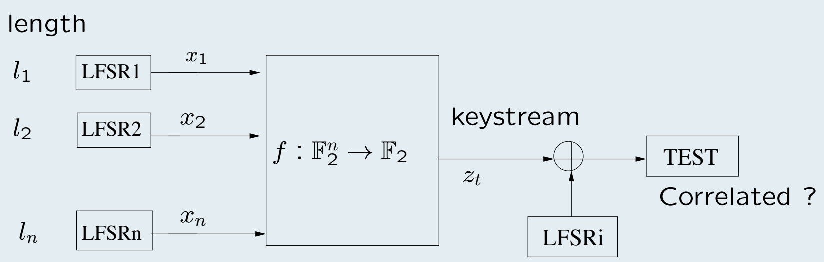
Introduction to correlation immunity

- Balanced Boolean functions in n variables are of degree $\leqslant n-1$.
- We might be interested in computing $Pb(f(x) = x_i)$! Consider the same $f(x) = x_3 + x_1x_2 + x_2x_3$ as before.

x_3	x_2	x_1	$f(x)$	$f(x) + x_1$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	0
1	1	1	1	0

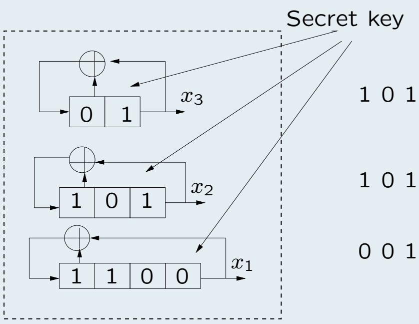
- Same situation, unbalancedness, for $f(x) + x_2$ and $f(x) + x_3$.

Correlation attacks



- Attack is performes by checking all states of LFSR1:
  - Guess not correct : We get a random sequence
  - Guess correct: Then $z_t \oplus x_1$ is biased, more zeros than ones .
- In previous example $Pb\{f(x) = x_i\} = 3/4$, thus possible to run test. The complexity of attack drops from $\prod_{i=1}^{3}(2^{L_i} - 1)$ to $\sum_{i=1}^{3}(2^{L_i} - 1)$

Example of correlation attacks



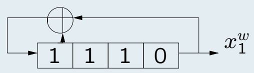
1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0

1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0

0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1

$z^t = f(x^t) = x_1 \oplus x_2x_3$ 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1 1

$f(x) \oplus x_1$ 1 0 1 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 $\leftarrow$ 16 zeros



0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 1 1 0 1 0 1 1 0

wrong key

$f(x) \oplus x_1^w$ 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 0 1 0 0 1

Protecting against correlation attacks

- If $Pb(f(x) = x_i) \neq 0.5$ then attack is performed in $\prod_{i=1}^{n}(2^{L_i} - 1)$.
- **Protection:** Design f such that $Pb(f(x) = x_i) = 0.5$ for all i.
- Then we might consider pairs of LFSRs and find correlation

$$Pb(f(x) = x_i + x_j) \neq 0.5$$

- There are techniques to construct correlation immune (nonlinear) functions of arbitrary order $1 \leqslant t \leqslant n - 3$.
- Only linear functions $f(x) = x_1 + x_2 + \cdots + x_n$ has maximum order of resiliency $(n - 1)$.

Trading-off linear complexity to correlation immunity

Definition: $f \in \mathcal{B}_n$ is *correlation immune* (CI) of order t if fixing any subset of input variables $x_{i_1}, \ldots, x_{i_r}$, $1 \leqslant r \leqslant t$ we have

$$Prob(f(x) = 0 \mid (x_{i_1}, \ldots, x_{i_r})) = Prob(f(x) = 1 \mid (x_{i_1}, \ldots, x_{i_r}))$$

If f is balanced and CI of order t then we say f is t-resilient

Trade-off Algebraic degree d of t-resilient function on $\mathbb{F}_2^n$ satisfies

$$d \leqslant n - t - 1.$$

- This means that protection from correlation attacks implies vulnerability to BM linear complexity synthesis and algebraic attacks.

Hypothesis testing

- For correlation attacks we used $Pb(f(x) = x_i) = p \neq \frac{1}{2}$. The length of the keystream N depends on p, if $p = \frac{1}{2}$ then $N \to \infty$.
- Define a random varible $\beta = N - \#\{f(x^{(t)}) \neq x_i^{(t)}\}_{t=1,\ldots,N}$.
- Then β is binomially distributed with mean value $m_{\beta|H_i}$ and deviation $\sigma^2_{\beta|H_i}$:

$$m_{\beta|H_1} = Np, \qquad \sigma^2_{\beta|H_1} = Np(1-p)$$
$$m_{\beta|H_0} = N/2, \qquad \sigma^2_{\beta|H_0} = N/4,$$

where H_1 and H_0 are the hypothesis of correct respectively wrong guess.

Some examples of resilient functions

- $f(x_1,\ldots,x_4) = x_1+x_2+x_3+x_4$ is 3-resilient function but linear, i.e. $deg(f) = 1$.
- How do we find nonlinear resilient functions ?
- For instance, $f(x_1,\ldots,x_4) = x_4(x_1+x_2)+(1+x_4)(x_2+x_3) = x_2+x_3+x_1x_4+x_3x_4$ is 1-resilient and of degree 2.
- To verify this one can check that
$$d_H(f,x_i) = \#\{x | f(x) \neq x_i\} = 2^{n-1} = 8$$
equivalent to $Pb(f(x) = x_i) = 1/2$ for any i.

Concatenating f and $1+f$

Theorem: Let $f \in \mathcal{B}_n$ be t-resilient of degree d. Then $\hat{f} = f||(1+f)$ on $\mathcal{B}_{n+1}$ is a $(t+1)$-resilient function of degree d.

Proof: Assume $Pb\{f+\sum_{j=1}^{t+1} x_{i_j} = 0\} \neq Pb\{f+\sum_{j=1}^{t+1} x_{i_j} = 1\}$, equivalently

$$\#\{f+\sum_{j=1}^{t+1} x_{i_j} = 0\} = 2^{n-1}+c \quad \neq \quad \#\{f+\sum_{j=1}^{t+1} x_{i_j} = 1\} = 2^{n-1}-c;$$

We compute,

$$\begin{aligned}\#\{\hat{f}+\sum_{j=1}^{t+1} x_{i_j} = 0\} &= \#\{f+\sum_{j=1}^{t+1} x_{i_j} = 0\} + \#\{1+f+\sum_{j=1}^{t+1} x_{i_j} = 0\} = \\ &= 2^{n-1}+c+2^{n-1}-c = 2^n.\end{aligned}$$

Nonlinearity of Boolean functions

- Sufficiently small correlation (deviation of p from $\frac{1}{2}$) makes requirement on keystream length N infeasible.

Definition: **Nonlinearity** of $f \in \mathcal{B}_n$ is defined as a minimum Hamming distance from the set of all affine functions, i.e.

$$\mathcal{N}_f = \min_{a \in \mathcal{A}_n} d_H(f, a),$$

where Hamming distance $d_H(f, a) = \{x | f(x) \neq a(x)\}$.

- Expressing $Pb(f(x) = x_i) = p = \frac{1}{2} \pm \epsilon$, the correlation coefficient ϵ is given as,

$$\epsilon = \frac{1}{2} - \frac{d_H(f, x_i)}{2^n}$$

Calculating nonlinearity -Example

x_3	x_2	x_1	$f(x)$	$x_1 + x_2$	$f(x) + x_1 + x_2$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	1	0	1

- Function f is at distance 6 from $x_1 + x_2$, then $d_H(f, 1 + x_1 + x_2) = 2$. Nonlinearity always less than 2^{n-1}.
- Procede for all linear functions and find minimum distance.

Walsh transform - a usefull tool

Definition: *Walsh transform* of $f \in \mathcal{B}_n$ in point $\alpha \in \mathbb{F}_2^n$ is defined by

$$\alpha \in \mathbb{F}_2^n \longmapsto \mathcal{F}(f+\varphi_\alpha) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\varphi_\alpha(x)} , \tag{1}$$

where $\varphi_\alpha(x) = \alpha \cdot x = \alpha_1 x_1 + \cdots \alpha_n x_n$.

- Then for $g(x) = \alpha \cdot x + b$ ($b \in \mathbb{F}_2$),

$$d_H(f,g) = 2^{n-1} - \frac{(-1)^b \mathcal{F}(f+\varphi_\alpha)}{2}. \tag{2}$$

- The nonlinearity of $f(x)$ is obtained via Walsh transform as,

$$\mathcal{N}_f = 2^{n-1} - \frac{1}{2} \max_{\alpha \in \mathbb{F}_2^n} |\mathcal{F}(f+\varphi_\alpha)|. \tag{3}$$

Representation of Walsh transform

- Computing

$$\{\mathcal{F}(f+\varphi_\alpha) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} (-1)^{\varphi_\alpha(x)} : \alpha \in \mathbb{F}_2^n\},$$

can be seen as a matrix multiplication.

$$\mathcal{F}(f+\varphi_\alpha) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} (-1)^{f(000)} \\ (-1)^{f(001)} \\ (-1)^{f(010)} \\ (-1)^{f(011)} \\ (-1)^{f(100)} \\ (-1)^{f(101)} \\ (-1)^{f(110)} \\ (-1)^{f(111)} \end{pmatrix}$$

Fast Walsh transform

- Straightforward implementation to compute the Walsh spectra

$$\{\mathcal{F}(f+\varphi_\alpha) : \alpha \in \mathbb{F}_2^n\}$$

requires 2^{2n} operations. Problem already for $n > 20$

- The $\{-1,1\}$ matrix H_n of size $2^n \times 2^n$ is called Silvester-Hadamard matrix - computed recursively. Start with,

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \cdots H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix}$$

- Enables a fast Walsh transform , only takes $n2^n$ operations.

FWT - example

- Function f on $\mathcal{B}_3$ with truth table $f = [00011101]$

For each pair (a,b) compute (a+b,a-b)

function	1	1	1	-1	-1	-1	1	-1
first	2	0	0	2	-2	0	0	2
second	2	2	2	-2	-2	2	-2	-2
final	0	4	0	-4	4	0	4	0

Interpretation of Walsh transform

- What is the meaning of $\mathcal{F}(f + x_1 + x_2) = 0$ for instance ?

$$\sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+x_1+x_2} = 0$$

• Means that $f(x) + x_1 + x_2$ is balanced, i.e. $Pb(f + x_1 + x_2 = 0) = Pb(f + x_1 + x_2 = 1)$.

• In other words no correlation attack by considering the sum of outputs generated by L_1 and L_2.

• Can we design functions having this property for any subset of input variables ?

Properties of Walsh transform

- Denote $\mathcal{F}(f + \varphi_\alpha) = \mathcal{F}(\alpha)$. Parseval's equality, valid for any $f \in \mathcal{B}_n$, states

$$\sum_{\alpha \in \mathbb{F}_2^n} \mathcal{F}(\alpha)^2 = 2^{2n}$$

Proof:

$$\begin{aligned}\sum_{\alpha \in \mathbb{F}_2^n} \mathcal{F}(\alpha)^2 &= \sum_{\alpha \in \mathbb{F}_2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)+\alpha \cdot x} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)+\alpha \cdot y} = \\ &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \sum_{y \in \mathbb{F}_2^n} (-1)^{f(y)} \underbrace{\sum_{\alpha \in \mathbb{F}_2^n} (-1)^{\alpha \cdot (x+y)}}_{0 \text{ for } x+y \neq 0} = 2^{2n}\end{aligned}$$

- Sum of squares constant, means that nonlinearity is maximized if $|\mathcal{F}(\alpha)| = 2^{\frac{n}{2}}$ for all α; bent functions

Trade-offs for cryptographic criteria

We would like to use Boolean functions satisfying:

- High algebraic degree (correlation immunity is traded-off)
- High order of resiliency (cannot have high resiliency order and nonlinearity)
- High nonlinearity (cannot achieve high resiliency)
- Low complexity of implementation (hard to achieve high degree)
- High algebraic immunity (unclear how it influences other parameters in general)
- Resistance to algebraic attacks !

Implementation complexity

- LFSR-based stream ciphers are especially suitable for hardware implementations.
- LFSR is efficiently implemented in hardware but what about Boolean functions ?
- $\mathbf{C}_\Omega(\mathbf{f})$– smallest number of gates of a circuit computing f, whose gates belong to Ω.
- Usually, $\Omega = \mathcal{B}_2$, set of Boolean functions in 2 variables.
- For Programmable Logic Arrays, $\Omega = (\wedge, \vee, \neg)$

Implementation complexity - example

Example: Consider f in 5 varibles:

- $x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_2x_5 +$
  $+x_3x_4 + x_3x_5 + x_4x_5$ 19 gates

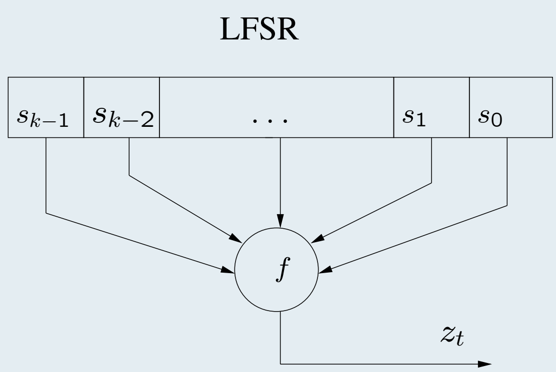
- $[(z + x_4)(z + x_5) + z] + [y(x_1 + x_2) + x_1]$
  with $z = y + x_3$ and $y = x_1 + x_2$ 10 gates

Shannon effect

For all $n \geqslant 9$ "almost all" Boolean functions in n variables have complexity $\mathbf{C}_{\mathcal{B}_2}(\mathbf{f})$ greater than, $2^n/n$.

Nonlinear filtering generator

Alternative design to destroy the linearity of LFSR.



Nonlinear filtering generator

Nonlinear filtering generator - properties

- Nonlinear filtering generator produces maximum-length sequence for:
  1. Primitive connection polynomial
  2. Balanced Boolean function
- Linear complexity is upper bounded by:

$$L_m = \sum_{i=1}^{m} \binom{L}{i},$$

where m is nonlinear order (degree) of function f.

- Fractions of Boolean functions of degree m which achieve L_m is,

$$P_m > e^{-1/L}.$$

Stream cipher applications

- A practical application of filtering generator or nonlinear combiner is as follows:
  - Choose a random secret key K, say 128 bits
  - Take a random nonused value of IV, say 128 bits.
  - Initialize the LFSR(s) with IV and K and run the cipher in NONOUTPUT MODE to mix IV and K bits.
  - After this has been done the initial state is acheived, $|S| \geqslant 256$ bits.

FCSR (Feedback Carry Shift Registers

- Introduced by Goresky & Klapper in 1993.
- Similar to LFSR but with propagation of carry bits. Computing the quotient of 2 integers as a 2-adic integer.
- A 2-adic integer can be viewed as a formal series with 2 as "variable"

$$\sum_{i=0}^{\infty} s_i 2^i, \quad s_i \in \{0,1\}.$$

.

- Addition and multiplication performed by taking carries to higher order terms, i.e. $2^n + 2^n = 2^{n+1}$.

Algebraic structure of 2-adic integers

- Somewhat weird properties like,

$$-1 = 1 + 2^1 + 2^2 + 2^3 + \cdots,$$

verified by adding 1 to both sides (with carry).

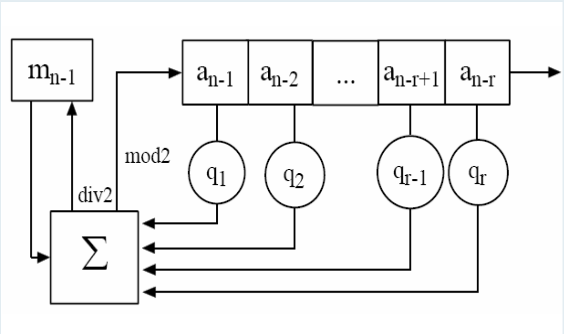
- Any negative integer belongs to the set of 2-adic numbers

$$-q = (-1)\sum_{i=0}^{k} q_i 2^i = \sum_{i=0}^{\infty} 2^i \sum_{i=0}^{k} q_i 2^i.$$

- Also any odd integer α has a unique inverse $\alpha\alpha^{-1} = 1$.
- We actually get a ring and in addition p/q is well-defined if q is odd.

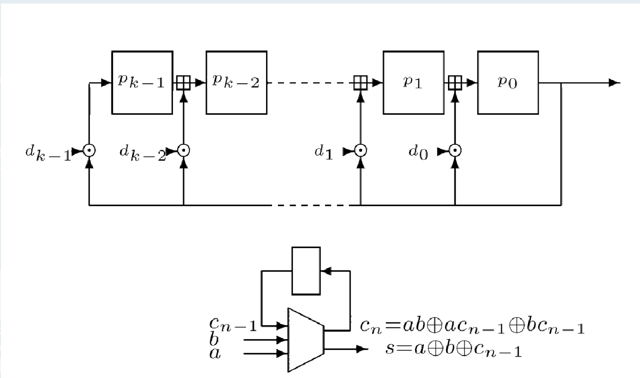
Gallois model of FCSR

1. Form the integer sum $\sigma_n = \sum_{i=1}^{r} q_i a_{n-i} + m_{n-1}$; a_i, q_i binary.
2. Shift the contents one step to the right, output a_{n-r} .
3. Put $a_n = \sigma_n$ mod 2 into the leftmost cell of the shift register.
4. Replace the memory integer m_{n-1} with $m_n = (\sigma - a_n)/2$



Fibonacci model of FCSR

- **Choice:** $q < 0 \leqslant p < |q|$, where $p = \sum_{i=0}^{k} p_i 2^i$, $q = 1 - 2\sum_{i=0}^{k-1} d_i 2^i$.



- Use a key p to initialize the main register, one more register for d.
- The circuit computes the 2-adic expansion of p/q.

Properties of 2-adic expansion

Theorem [Periodicity]: Let a $= (a_i)$ be binary sequence and $\alpha = \sum_{i=0}^{\infty} a_i 2^i$ associated 2-adic number. a is strictly periodic iff,

$$\alpha = p/q, \quad q \text{ is odd}, \quad \alpha \leqslant 0 \text{ and } |\alpha| \leqslant 1.$$

Theorem: If p and q are relatively prime integers with q odd, then the period of 2-adic expansion of p/q is the order of 2 modulo q , i.e. the least positive integer T such that,

$$2^T \equiv 1 \pmod{q}.$$

- Proving properties of 2-adic numbers is tedious, we only show sufficiency for periodicity.

Periodicity theorem

Proof: Let a $= (a_0, a_1, a_2, \ldots)$ be a strictly periodic sequence of period T. Set $\alpha = \sum_{i=0}^{\infty} a_i 2^i$. Then,

$$\begin{aligned} 2^T\alpha &= \sum_{i=0}^{\infty} a_i 2^{i+T} = \sum_{i=0}^{\infty} a_{i+T} 2^{i+T} = \\ &= \sum_{i=T}^{\infty} a_i 2^i = \alpha - \sum_{i=0}^{T-1} a_i 2^i \\ &\Downarrow \\ \alpha &= -\frac{\left(\sum_{i=0}^{T-1} a_i 2^i\right)}{(2^T - 1)} \end{aligned}$$

- α is a negative rational number. Writing $\alpha = p/q$, and taking q positive, $|\alpha| < 1$, q is odd.

Statistical properties of the FCSR sequence

Heuristic assumption: the sequence S has no statistical property which can be used to discriminate it from a random sequence, except its low 2-adic complexity.

- Experiments support this assumption (e.g. NIST Statistical test suite)
- A similar assumption is usual for LFSRs.

- If we choose a negative retroaction prime q such that

$$2^k < |q| < 2^{k+1} \text{ and } \mathrm{ord}_q(2) = T = |q| - 1,$$

then with each nonzero initialization, we get a sequence of period T in which any word of k bits is a subsequence.

2-adic complexity

- 2-adic complexity Λ(similar to linear complexity)- smallest number of cells of FCSR that generates S.
- If p and q co-prime then complexity $\Lambda = \max(wt(|p|), wt(|q|))$.
- Extended Euclidean Algorithm applied to integers $2^{2\Lambda+1}$ and $S_n = \sum_{i=0}^{2\Lambda} s_i 2^i$ recovers p, q using only $2\Lambda + 1$ bits of $S = p/q$.
- Difference to LFSR: Structure of FCSR need not to be destroyed by nonlinear filtering (FCSR is itself nonlinear).
- Enough to take linear Boolean function

Comparison of FCSR to LFSR

Property	LFSR	FCSR
Implem. Architecture	Galois and Fibonacci	Galois and Fibonacci
Max. period sequence	m-sequence	l-sequence
Initial state alg.	Berlekamp-Massey	Rational Approximate
Correlation	Periodic correlation	Arithmetic correlation
Complexity	Linear span	2-adic span
Algebraic expression	Finite field	p-adic numbers
Mathematical tool	Finite fields	p-adic theory
Security	Known	Unclear

- Increase of the length of output l-sequence, repetition is observed (DFT test) before full period is formed, which deteriorates pseudo-random property.
- So the available length of l-sequence is much less than the real period.

Statistical weaknesses of FCSR

- Some choices of q gives weak statistical properties, does not pass DFT test.

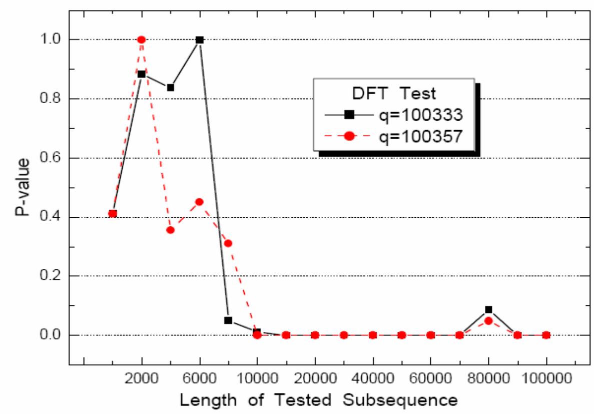
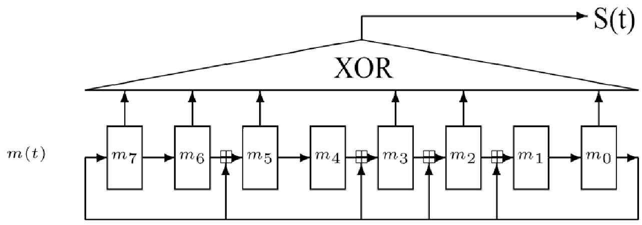


Figure 7 Result of DFT test on the 2nd group q

Filtering FCSR - an example

$q = -347, \qquad d = \frac{|q|+1}{2} = 174 = (10101110)_2,$

Filter: $F = (11101101)_2$



- As 2-adic and linear structures are unrelated we may use simple XOR filtering.

F-FCSR - an eSTREAM candidate

- Interesting dual proposal of FCSR to eSTREAM, both SW and HW.
- The original design was flawed due to weak initialization scheme.
- Initial state of the carry register (derived from IV and key) only 68 bits. Two different carry gives the same carry state with $Pb = 2^{-68}$. Birthday paradox finds coallision after randomly chosen 2^{34} IV's.

$\boxed{\text{Need } 2^{34} \times 68 \text{ bits to perform distinguishing attack}}$.

- Internal state was supposed to have $128 + 68$ bits but due to few initial clockings; real entropy only 128 bits.

$\boxed{\text{This allows TMD trade-off attack with complexity } 2^{64}.}$

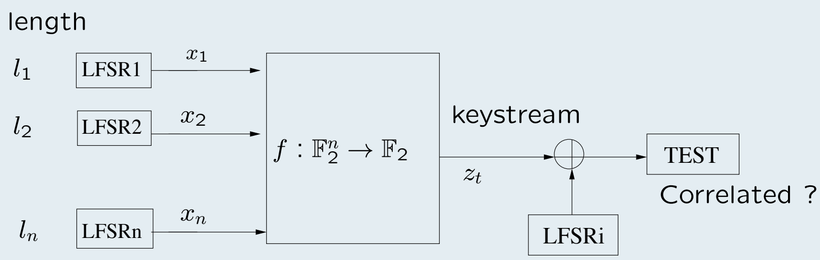
Chapter 7

LFSR-based Stream Ciphers

Content of this chapter:

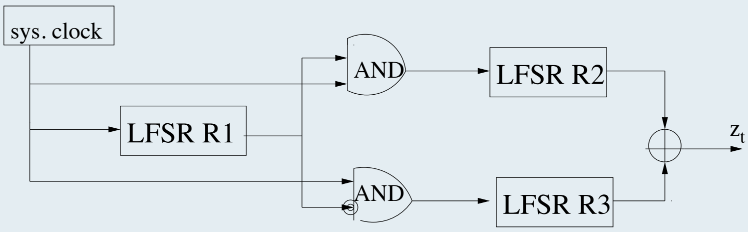
- LFSR-based stream ciphers - alternating step generator, shrinking and summation generator.
- Bluetooth and GSM encryption algorithms A5/x, E_0.
- Use of NFSR in modern stream ciphers.
- Low hardware implementations of stream ciphers - eSTREAM candidates Trivium, Grain-128.

Avoiding correlation attacks



- Attack is avoided through good design of Boolean function (hard)
- Alternatively, destroy linearity of LFSR by other means !

Clock controlled generators - alternating step generator



- Clock register R_1:
  - If the output of R_1 is 1 then R_2 clocked, R_3 not clocked but its last output bit is repeated;
  - If the output of R_1 is 0 then R_3 clocked, R_2 not clocked but its last output bit is repeated;
  - Output the XOR of R_2 and R_3.

Properties of alternating generator

Idea: Introduce the nonlinearity by irregular clocking of LFSRs; irregular decimation of sequence.

- Assume R_1 produces a deBrujin sequence of period 2^{L_1} (add a 0 at the end of maximum length sequence); L_2 and L_3 produces maximum-length sequence:
  - The period of sequence is $2^{L_1}(2^{L_2}-1)(2^{L_3}-1)$
  - The linear complexity satisfies:

$$(L_2+L_3)2^{L_1-1} < LC \leqslant (L_2+L_3)2^{L_1}$$

  - Good statistics; distribution of patterns is almost uniform

Security of alternating generator

Assumptions:

• The length of shift registers are pairwise
relatively prime; approximately of same length $L_1 \approx L_2 \approx L_3$.
• Maximum length LFSRs (primitive connection polynomials)

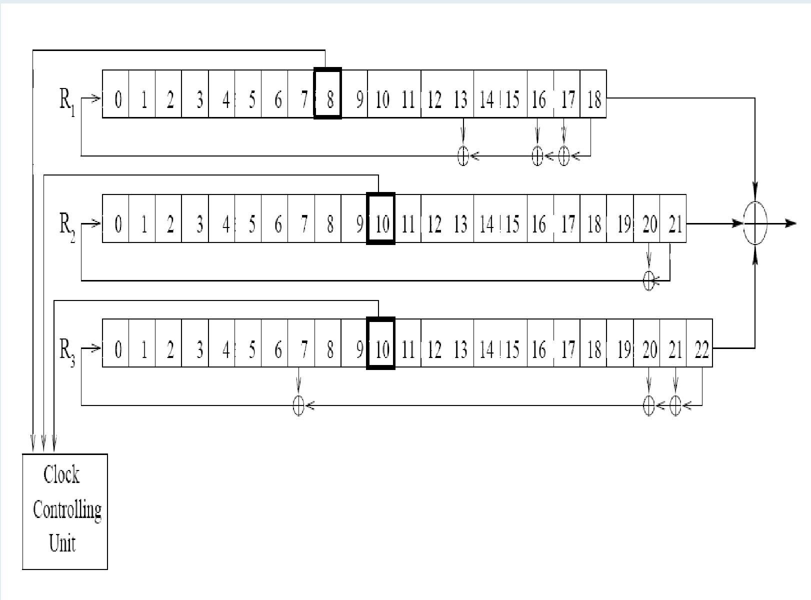
- The best known attack is guess-and-determine attack on R_1.
- Thus, taking $L_1 \approx 128$ the generator is secure against all presently known attacks.

A slight disadvantage of the LFSR's lengths is relatively large state (initialization).

History of A5/x

- A5/1 is an irregularly clocked stream cipher developed in 1987 as the GSM standard.
- Design was kept secret, the general design was leaked in 1994, full specification in 1999
- In 2000, around 130 million of GSM customers relied on A5/1.
- A5/2 is a deliberate weakening for certain export regions (US)
- Though the key is of length 54 bits (in GSM implementation, designed for 64 bits); weak cipher even ciphertext-only attack in few seconds on PC !

A5/1 stream cipher



Description of A5/1

- The registers R_1, R_2, R_3 are clocked depending on the majority of bits in positions $(8, 10, 10)$ respectively.
- For instance if $(8, 10, 10) = (0, 1, 1)$ then R_2 and R_3 are clocked- more 1s than zeros.
- Either two or three registers are clocked each time.
- GSM implementation uses only 54 key bits and IV frame number of 22 bits.

Initialization of A5/1

1. The registers are first set to zero.

2 For 64 cycles the ith key bit is added to the least significant bit of R_1, R_2, R_3 and each register is clocked

$$R_j[0] = R_j[0] \oplus K[i]; \quad 0 \leqslant i < 64; \quad j \in [1, 3].$$

3 Similarly, the 22 bits of the frame number are added in 22 cycles.

4. Then the cipher is run for 100 cycles in NOOUTPUT MODE.

Attacks on A5

- Several known-plaintext attacks such as Golic's (linear equations attack) from 1997.

- In 2000, time-memory-data trade-off attack recovers the key:
  - In 1 second using 2 min. of conversation
  - Several minutes from 2 seconds of known plaintext.

In 2006, Barkan, Biham and Keller present ciphertext-only attack :

> "We present a very practical ciphertext-only cryptanalysis of GSM encrypted communication, and various active attacks on the GSM protocols. These attacks can even break into GSM networks that use "unbreakable" ciphers. We first describe a ciphertext-only attack on A5/2 that requires a few dozen milliseconds of encrypted off-the-air cellular conversation and finds the correct key in less than a second on a personal computer.We extend this attack to a (more complex) ciphertext-only attack on A5/1 ..."

Replacement A5/3

- Due to serious weaknesses of A5/1 (and especially A5/2) in the 3GPP mobile standard A5/1 was replaced by A5/3.
- A5/3 is actually a 128 bit block cipher named KASUMI, which is an optimized modification of MISTY1 (designed in 1995) for hardware applications.
- Is 3GPP more secure than GSM ?
- YES and NO; KASUMI was broken in 2005, 2^{55} chosen plaintexts , and time complexity 2^{76}.

Not a practical attack but the security of 3GPP is compromised.

Shrinking generator

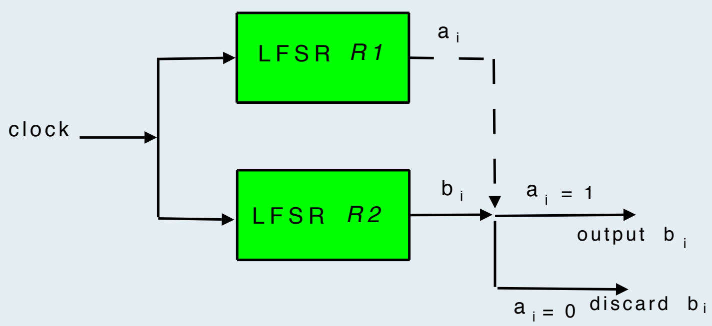
- Proposed at Crypto 93' as an alternative method of LFSR-based ciphers.
- The cipher can be viewed as a variable clock control generator.
- The analysis of statistical properties is relatively easy
  - Very fast implementation in hardware
  - Output rate is not regular; need for buffering

Shrinking generator - description

1. Registers R_1 and R_2 are clocked.

2 Output of R_1 is 1, the output of R_2 forms part of the keystream.

3 Output of R_1 is 0, the output bit of R_2 is discarded.



Shrinking generator - example

Suppose:

- Registers of length $L_1 = 3$ and $L_2 = 5$;
- Connection polynomials $C_1(x) = 1+x+x^3$ and $C_2(x) = 1+x^3+x^5$;
- Let initial states be $[1,0,0]$ and $[0,0,1,0,1]$.

Then:

$$\begin{aligned} a^7 &= 0,0,1,1,1,0,1,0,0,1,1,1,0,1,0,0,1,1,1,0,1,0,0,1,1,1,0,1,0,0,1 \\ b^{31} &= 1,0,1,0,0,0,0,1,0,0,1,0,1,1,0,0,1,1,1,1,1,0,0,0,1,1,0,1,1,1,0 \\ s &= 1,0,0,0,0,1,0,1,1,1,1,1,0,1,1,1,0,\ldots \end{aligned}$$

Security of shrinking generator

Security of shrinking generator depends heavily on the knowledge of connection polynomials.

- If the connection polynomials are known the best known key recovering attack takes $\mathcal{O}(2^{L_1}L_2^3)$ operations.
- Keeping secret connection polynomials gives $\mathcal{O}(2^{2L_1}L_1L_2)$.
- For practical applications taking $L_1 \approx L_2 \approx 64$ implies attack complexity 2^{128}.

Implementing benefits of shrinking generator

- Probably the most efficient hardware structure that exists: only two LFSRs, simple logic and buffer.
- Ideally suited for low hardware overhead such as RFID (Radio Frequency Identification) application.

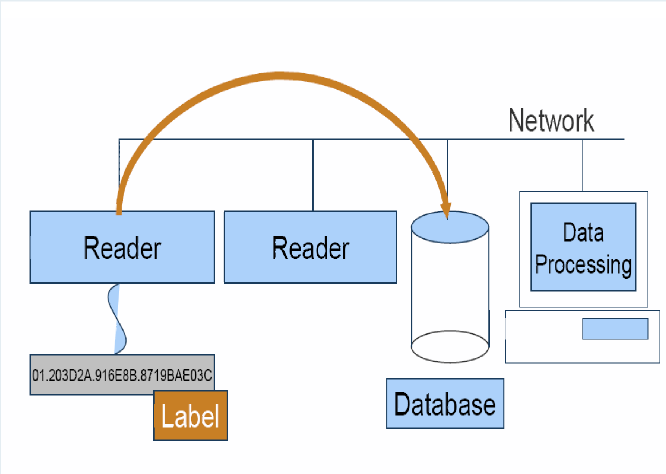
Still there are two problems:

- Keeping secret connection polynomials is not a good secrecy policy (even for hardware applications)
- Irregular rate and need for buffering.

RFID application of stream ciphers

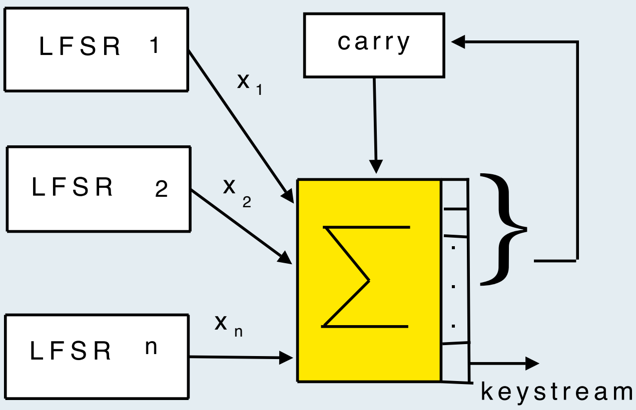
- Very interesting application; inevitable use of low hardware complexity stream ciphers
- Requirement for RFID:
  - Essentially the next generation of barcodes - EPC (Electronic Product Code)
  - Tagging 20 million items with 5 cent tag costs $\$1,000,000$.
  - 2000-4000 gates available for security (cost limitation)
  - AES implementation requires 20,000-40,000 gates (some says), 20 cents extra per tag.

RFID - system interface



Summation generator

- Idea is to use integer addition rather than addition (mod 2). Bits from n sequences (LFSR of max. length) are added as integers together with carry.



Summation generator - operation mode

- Secret key consists of initial states of LFSRs, and an initial carry C_0.

1. At time $t \geqslant 1$ the LFSRs are clocked giving outputs $x_1, \dots, x_n$, and the *integer sum* is computed:
$$S_t = \sum_{i=1}^{n} x_i + C_{t-1}.$$
2. The keystream bit is $z_t = S_t \pmod 2$ (the least significant bit of S_t).
3. The new carry C_t is computed as $\lfloor S_t/2 \rfloor$ (remaining bits of S_t).

Summation generator - an important example

- Consider $n = 4$, then $0 \leqslant \sum_{i=1}^{4} x_i \leqslant 4$. Therefore using 2 bits for carry the integer sum
$$S_t = \sum_{i=1}^{n} x_i + C_{t-1},$$
is well-represented as $S_t \leqslant 7$ so that
$$C_t = \lfloor S_t/2 \rfloor \Rightarrow wt_H(C_t) = 2.$$

Example: Assume $C_0 = (0, 1)$ and $(x_1^1, x_2^1, x_3^1, x_4^1) = (1, 1, 0, 1)$ (LSB is the rightmost bit). Then,

$$\begin{aligned} S_1 &= 3 + 1 = 4 = (1, 0, 0) \\ C_1 &= \lfloor 4/2 \rfloor = 2 = (1, 0) \end{aligned}$$

Balancedness of summation generator

- Computing the keystream and carry bits is given below:

S_t dec.	S_t bin.	C_t dec.	C_t bin	Nmb. possib. for (x, C_{t-1})
0	(0,0,0)	0	(0,0)	1
1	(0,0,1)	0	(0,0)	5
2	(0,1,0)	1	(0,1)	11
3	(0,1,1)	1	(0,1)	15
4	(1,0,0)	2	(1,0)	15
5	(1,0,1)	2	(1,0)	11
6	(1,1,0)	3	(1,1)	5
7	(1,1,1)	3	(1,1)	1

- For instance, $S_t = 1$ gives:
  - $\sum_{i=1}^{4} x_i = 0$ and $C_{t-1} = 1$ (only if $(x_1, x_2, x_3, x_4) = (0, 0, 0, 0)$)
  - $\sum_{i=1}^{4} x_i = 1$ and $C_{t-1} = 0$ (four cases $(1, 0, 0, 0), \ldots, (0, 0, 0, 1)$).

Introducing nonlinearity through carry bits

- Bitwise XOR is a linear operation over $\mathbb{F}_2$. That is, given $a = (a_0, \ldots, a_{n-1})$ and $b = (b_0, \ldots, b_{n-1})$ we compute $a \oplus b = d$ as,

$$d_i = a_i \oplus b_i; \; i = 0, \ldots, n-1.$$

- Using bit representation $d = a + b \pmod{2^n}$ is computed as,

$$d_i = a_i \oplus b_i \oplus c_i; \; i = 0, \ldots, n-1,$$

  where c_i is a carry bit computed as (adopting $c_0 = 0$),

$$c_{i+1} = a_i b_i \oplus c_i(a_i \oplus b_i), \;\; i = 0, \ldots, n-2.$$

- In each step of iteration the degree is increased by one so that d_{n-1} is a function of degree n in input bits , $a_0, \ldots, a_{n-1}$ and $b = b_0, \ldots, b_{n-1}$.

Small example of modular addition

- Consider the XOR addition of $a = (0, 1, 1, 1)$ and $b = (1, 1, 0, 1)$.

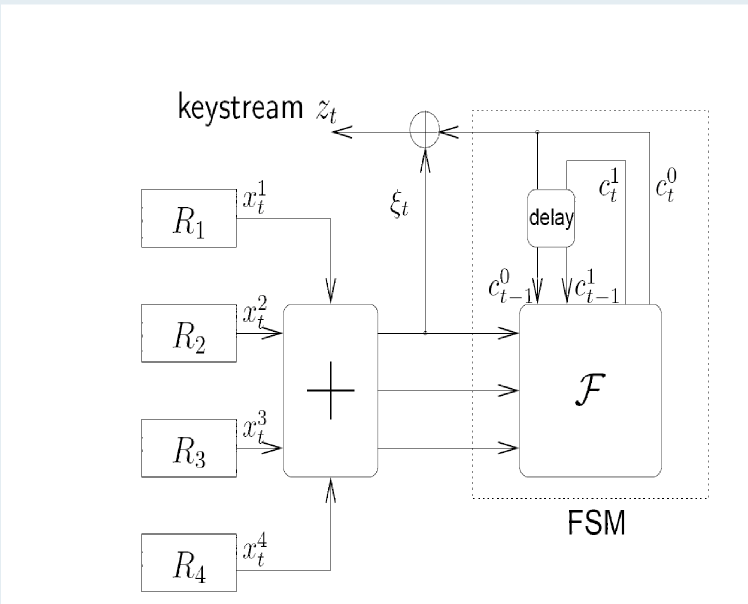
$$\begin{array}{rccccc} & 0 & 1 & 1 & 1 & a \\ \oplus & 1 & 1 & 0 & 1 & b \\ \hline & 1 & 1 & 1 & 0 & d \end{array}$$

- Addition $\pmod{16}$ gives,

$$\begin{array}{rccccc} & 0 & 1 & 1 & 1 & a \\ \pmod{16} & 1 & 1 & 0 & 1 & b \\ \hline & 0 & 1 & 0 & 0 & d \end{array}$$

- Note that the carry bits are given by: $c_0 = 0$, $c_1 = a_0 b_0$, $c_2 = a_1 b_1 \oplus a_0 b_0 (a_1 \oplus b_1)$, $c_3 = a_2 b_2 \oplus c_2 (a_2 \oplus b_2)$.

E0 stream cipher



E0 stream cipher

- E0 stream cipher is a summation generator with four bits of memory $(c_{t-1}^1, c_{t-1}^0, c_t^1, c_t^0)$
- The four registers $R_1, \ldots, R_4$ are of length 25, 31, 33, and 39 respectively. Thus, 128 key bits are stored in LFSRs .
- The keystream bits are computed as:

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0$$

- Using temporary bits $s_{t+1} = (s_{t+1}^1, s_{t+1}^0)$ state is updated via:

$$\begin{aligned} s_{t+1} &= \left\lfloor \frac{\sum_i x_t^i + 2c_t^1 + c_t^0}{2} \right\rfloor \\ c_{t+1}^1 &= s_{t+1}^1 + c_t^1 + c_{t-1}^0 \\ c_{t+1}^0 &= s_{t+1}^0 + c_t^0 + c_{t-1}^1 + c_{t-1}^0 \end{aligned}$$

E0 - state transition

- Denoting current state by $\sigma_t = (c_{t-1}, c_t)$, the next state $\sigma_{t+1} = (c_t, c_{t+1})$ can be computed:

Table 3.4: State transition of σ_{t+1} given $w(x_t)$ and σ_t

		σ_t															
		00	01	02	03	10	11	12	13	20	21	22	23	30	31	32	33
$w(x_t)$	0	00	11	23	32	03	12	20	31	01	10	22	33	02	13	21	30
	1	00	10	23	31	03	13	20	32	01	11	22	30	02	12	21	33
	2	01	10	20	31	02	13	23	32	00	11	21	30	03	12	22	33
	3	01	13	20	30	02	10	23	33	00	12	21	31	03	11	22	32
	4	02	13	21	30	01	10	22	33	03	12	20	31	00	11	23	32

E0 - basic properties

- All nonlinearity of the keystream collected in the carry bit c_t^0.
- The carry c_t^0 depends on the initial state σ_0 and all the previous inputs $x_{t-1}, x_{t-2}, \ldots, x_0$.
- There is no correlation to the subset of inputs $(x_t^1, x_t^2, x_t^3, x_t^4)$. E.g. considering c_t^0 as independent balanced variable then,

$$Pb(z_t = x_t^1 \oplus x_t^2) = Pb(x_t^3 \oplus x_t^4 \oplus c_t^0 = 0) = \frac{1}{2}$$

- Assume $\sum_i x_t^i = 2$ holds for t_0, t_0+1, t_0+2, t_0+3 then,

$$c_{t_0}^0 + c_{t_0+1}^0 + c_{t_0+2}^0 + c_{t_0+3}^0 + c_{t_0+4}^0 = 1,$$

which can be used to mount a correlation attack.

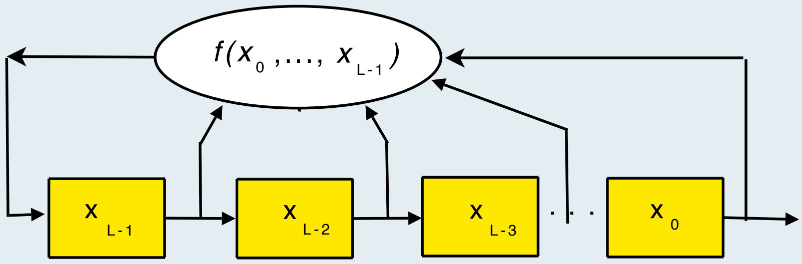
E0 - statistical properties

- The cryptographic properties of summation generator are summarized by:
  - The period of the keystream $T = \prod_{i=1}^4 (2^{L_i} - 1)$.
  - Linear complexity is close to the period.
  - Maximum correlation immunity in the common sense
- Summation generator is vulnerable to certain (conditional) correlation attacks.
- There are attacks that "breaks" E0 in academic sense, but none of these is applicable in practice.

NFSR (Nonlinear Feedback Shift Registers)

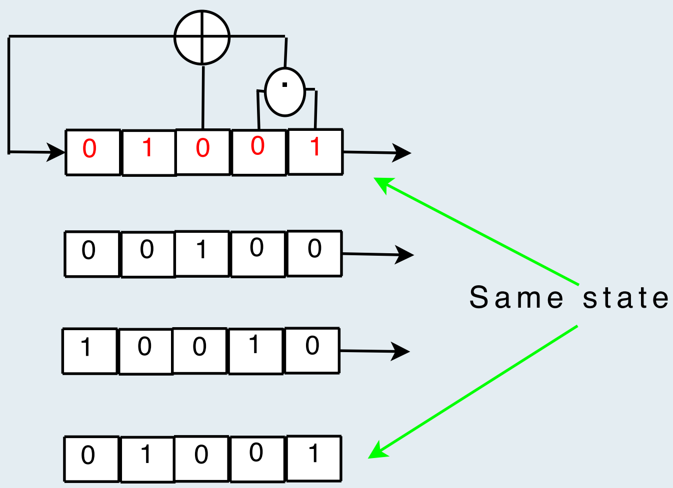
- The current tendency for low hardware complexity is to use NFSR with (or without) additional LFSRs.
- NFSR introduces the nonlinearity directly into keystream. As Jim Massey expressed it:

  "The linearity is curse of the cryptographer."



NFSR example

- Only certain combining functions results in sequences with maximum period.



Conditions on feedback function

- Feedback function is a Boolean function in L variables,

  $f(x_0, \dots, x_{L-1}) = c_1 1 \oplus c_{x_0} x_0 \oplus \dots \oplus c_{x_{L-1}} x_{L-1} \oplus \dots \oplus c_{x_0 \cdots x_{L-1}} x_0 \cdots x_{L-1}$

- Only certain combining functions results in sequences with maximum period (necessary conditions):

  1. c_1 must be equal to one
  2. The number of terms is even
  3. $f(x_0, x_1, \dots, x_{L-1}) = x_0 \oplus g(x_1, \dots, x_{L-1})$
  4. There is a symmetry between the x_i and the x_{L-i} variables
  5. At least there is one $c_{x_i} = 0$.

Conditions on feedback function II

• Most of the necessary conditions for maximum period are easily proved:

Proof: 1) The state $(1, 0, 0, \dots, 0)$ must follow $(0, 0, 0, \dots, 0)$ state,

$$f(0, 0, \dots, 0) = c_1 \cdot 1 = 1 \Longrightarrow c_1 = 1.$$

2) Similarly $(0, 1, 1, \dots, 1)$ must follow $(1, 1, 1, \dots, 1)$ state,

$$f(1, 1, \dots, 1) = \sum_{i=0}^{2^L - 1} c_i = 0 \Longrightarrow \text{Even number of } c_i = 1.$$

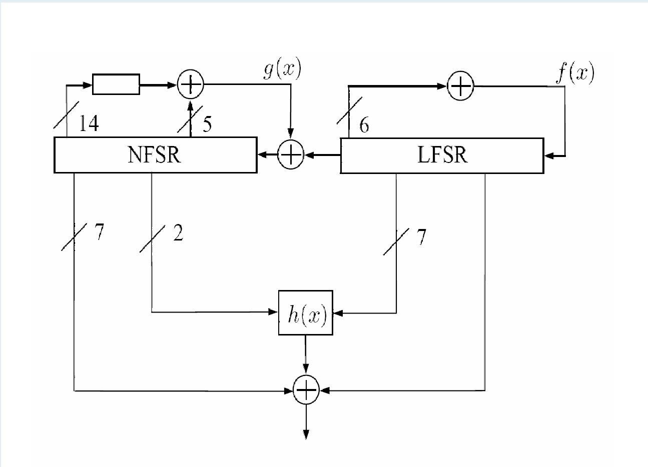
- Several eStream candidates utilize NFSR in the design, two efficient design approaches are GRAIN-128 and TRIVIUM.

- For better statistical properties LFSRs can be combined.

Grain-128 - description

- Modern stream cipher designed for low hardware complexity.
- Grain Version 0 was first submitted to the eSTREAM project, but successfully cryptanalyzed !
- A tweaked version reffered to as Grain-128 have reached final third phase of eSTREAM project.
- Grain-128 is a nonlinear filtering generator - the filter (Boolean function) takes the input from one NFSR and one LFSR.

Grain-128



The numbers along lines depict the number of input bits.

Grain-128 - design rationales

- The nonlinear feedback polynomial is a sum of one linear and one bent function (maximal nonlinearity):

$$\begin{aligned} g(x) &= 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + \\ &+ x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + \\ &+ x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}. \end{aligned}$$

- The choice of taps is probably quite arbitrary. The filtering function is simple and of low degree:

$$h(y) = y_0y_1 + y_2y_3 + y_4y_5 + y_6y_7 + y_0y_4y_8.$$

  - Bits y_0 and y_4 come from NFSR; the remaining 7 bits come from LFSR.

Grain-128 - key initialization

- The output is fed back during initialization, clocked 256 times.

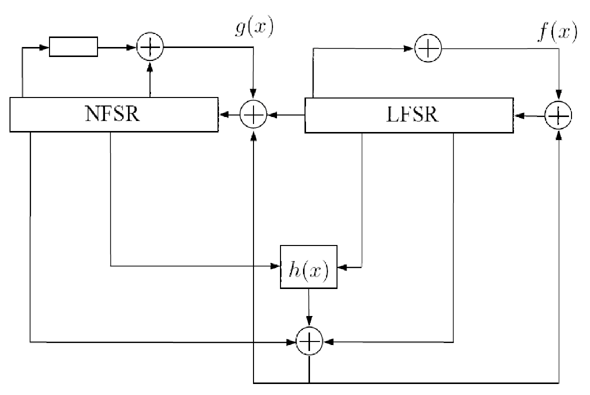


Fig. 2. The key initialization.

Grain-128 - design rationales II

- Internal state 256 bits; protects from time-memory-data attack
- Speed acceleration :Functions f, g and h can be implemented several times - producing several bits at the time (up to 32 bits) !
- LFSR for long period; NFSR for high confusion (alg. degree)

- Motivations for choices of f, g, h:

- Function f is primitive connection polynomial (not sparse !).
- Function g is easy to implement - linear part adds resiliency and quadratic terms nonlinearity.
- h similarly designed as g - mixed inputs from LFSR and NFSR.

Grain-128 - hardware complexity

- 2 input NAND gate "defined" to have gate count 1.

Gate Count	*Speed Increase*					
Building Block	1x	2x	4x	8x	16x	32x
LFSR	1024	1024	1024	1024	1024	1024
NFSR	1024	1024	1024	1024	1024	1024
$f(\cdot)$	12.5	25	50	100	200	400
$g(\cdot)$	37	74	148	296	592	1184
Output func	35.5	71	142	284	568	1136
Total	2133	2218	2388	2728	3408	4768

Trivium - another eStream candidate

- A hardware oriented stream cipher
  - Extreme simplicity, elegance, speed - still no efficient attacks.
- Design based on the mutual update of 3 NFSRs
- Possibility for increased speed as for Grain-128
- Period is hard to determine, hopefully the probability of short cycles is infitensimaly small !

Trivium - run mode

- The state consists of 288 bits $(s_1, s_2, \ldots, s_{288})$.

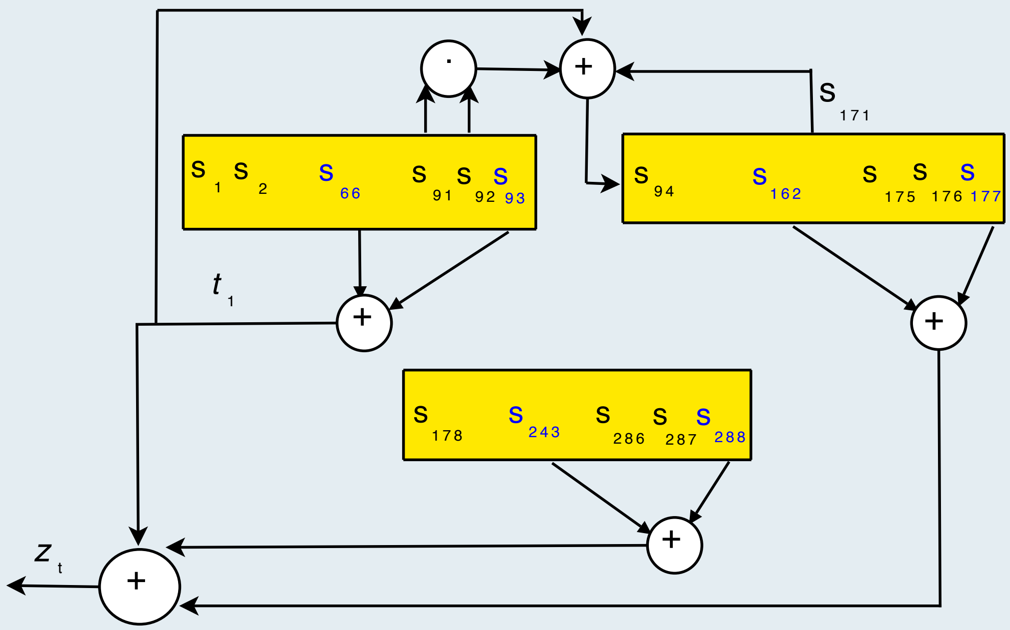
$$
\begin{aligned}
\text{for } i &= 1 \text{ to } N \text{ do} \\
t_1 &\leftarrow s_{66} + s_{93} \\
t_2 &\leftarrow s_{162} + s_{177} \\
t_3 &\leftarrow s_{243} + s_{288} \\
z_i &\leftarrow t_1 + t_2 + t_3 \\
t_1 &\leftarrow t_1 + s_{91} \cdot s_{92} + s_{171} \\
t_2 &\leftarrow t_2 + s_{175} \cdot s_{176} + s_{264} \\
t_3 &\leftarrow t_3 + s_{286} \cdot s_{287} + s_{69} \\
(s_1, s_2, \ldots, s_{93}) &\leftarrow (t_3, s_1, \ldots, s_{92}) \\
(s_{94}, s_{95}, \ldots, s_{177}) &\leftarrow (t_1, s_{94}, \ldots, s_{176}) \\
(s_{178}, s_{179}, \ldots, s_{288}) &\leftarrow (t_2, s_{178}, \ldots, s_{287})
\end{aligned}
$$

Trivium - key and IV setup

- The cipher is initialized by :
  - Load 80-bit key and 80-bit IV into the 288-bit initial state,
  - Set all remaining bits to 0, except for s_{286}, s_{287}, and s_{288}.
  - Run the cipher 4 full cycles (as above) in NOOUTPUT mode

$$
\begin{aligned}
(s_1, s_2, \ldots, s_{93}) &\leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0) \\
(s_{94}, s_{95}, \ldots, s_{177}) &\leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0) \\
(s_{178}, s_{179}, \ldots, s_{288}) &\leftarrow (0, \ldots, 0, 1, 1, 1) \\
\text{for } i &= 1 \text{ to } 4 \cdot 288 \text{ do} \\
\text{Compute} & \quad t_1, t_2, t_3 \text{ as above} \\
(s_1, s_2, \ldots, s_{93}) &\leftarrow (t_3, s_1, \ldots, s_{92}) \\
(s_{94}, s_{95}, \ldots, s_{177}) &\leftarrow (t_1, s_{94}, \ldots, s_{176}) \\
(s_{178}, s_{179}, \ldots, s_{288}) &\leftarrow (t_2, s_{178}, \ldots, s_{287})
\end{aligned}
$$

Trivium scheme (update of second NFSR only)



Trivium - hardware complexity

- Authors count 12 NAND gates per stage of NFSR (for Grain-128 authors used 8 NAND gates)

Components	1-bit	8-bit	16-bit	32-bit	64-bit
Flip-flops:	288	288	288	288	288
AND gates:	3	24	48	96	192
XOR gates:	11	88	176	352	704
NAND gate count:	3488	3712	3968	4480	5504

Chapter 8

Other Primitives

Content of this chapter:

- Software oriented stream ciphers using LFSR, SNOW 2.0.
- Basic mathematical background.
- S-boxes and functions over finite fields.
- Vectorial Boolean functions.
- T-functions and stream ciphers based on T-functions.

Design rationales for software applications

- Basic recommendations for software-oriented design are

▶ Use "software-friendly" operations,

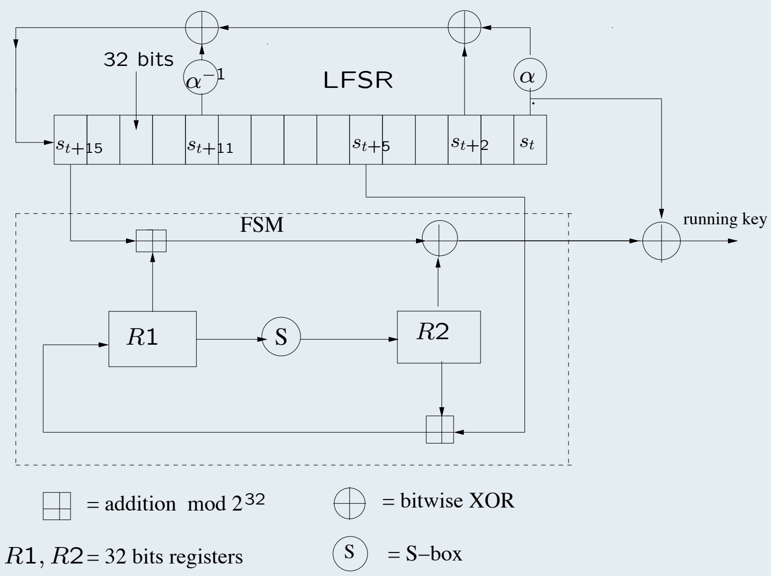
 ▷ Design fast stream cipher - at least 3x faster than AES

 ▷ Do not compromize security by making the cipher too fast

 ▷ Be careful about proper initialization procedure, should be fast but secure as well

▶ Suitable operations for this purpose are logical operations, modular addition and multiplication . . . but even LFSR over extension fields

Software oriented ciphers based on LFSR - SNOW 2.0



Snow 2.0 - design approach

- SNOW 2.0 is designed for dedicated software applications on 32-bit processors.
- Almost all fundamental design strategies included:
  - LFSR for long period (however the period might be shorter due to FSM)
  - Mixing different operations XOR, addition modulo 2^{32} and S-box.
  - S-box for additional confusion, high algebraic degree.
  - Simplicity and effectiveness

Mixing operations - loosing algebraic structure

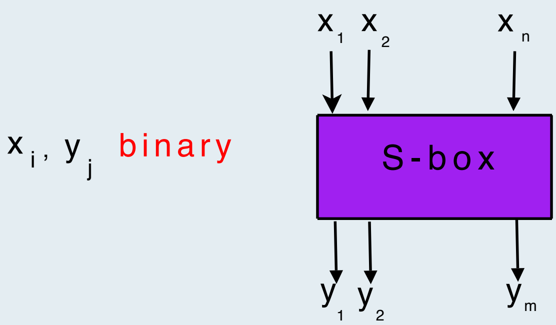
- SNOW 2.0 mixes XOR and modulo 2^{32} addition - no associativity $(a \oplus b) \boxplus c \neq a \oplus (b \boxplus c)$.

$$\begin{array}{ccc ccc}
 & a & 1001 & & c & 0001 \\
\oplus & b & 0111 & \boxplus & b & 0111 \\
\hline
 & & 1110 & & & 1000 \\
\boxplus & c & 0001 & \oplus & a & 1001 \\
\hline
 & & 1111 & & & 0001
\end{array}$$

- Hard to express dependency between input and output.
- For efficient implementation 4 S-boxes (8×8) are used. 4 lookup tables of size 256 bytes.

S-boxes

- Substitution box (S-box) is a basic nonlinearity component in block ciphers.
- Used in software oriented stream ciphers to induce nonlinearity.



• Correspond to a mapping $S : GF(2)^n \to GF(2)^m$. If $n = m$ symmetric S-box , a mapping over a field $GF(2^n)$!

Do we need both S-box and $+ \pmod{2^{32}}$ in SNOW 2.0

• YES WE DO !!

- Assume we exclude S-box in SNOW 2.0. Recall that using bit representation $d = a + b \pmod{2^n}$,

$$d_i = a_i \oplus b_i \oplus c_i; \;\; i = 0, \ldots, n-1,$$

c_i is a carry bit computed as (adopting $c_0 = 0$),

$$c_{i+1} = a_i b_i \oplus c_i(a_i \oplus b_i), \;\; i = 0, \ldots, n-2.$$

- The LSB $d_0 = a_0 + b_0$ is linear function of inputs. One input comes directly from LFSR.

The LSB of the keystream word linear function of secret state bits !!

Basic mathematical background - Groups

- *Group* is a set G together with an operation "$\circ$" satisfying:

1. $\forall a, b \in G : a \circ b \in G$ Algebraic closure
2. $\forall a, b, c \in G : a \circ (b \circ c) = (a \circ b) \circ c$ Associativity
3. $\exists ! e \in G : \forall a \in G : a \circ e = e \circ a = a$ e is identity element
4. $\forall a \in G, \exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$ Inverse element

Theorem: The order of an element (least integer t such that $a^t = e$) of a finite group divides the order of the group.

Examples of Groups

- $(\mathbb{Z}, \cdot)$ is not a group as,

$$3^{-1} = ? \text{ i.e. } 3 \cdot x = 1 \text{ has no solution in } \mathbb{Z}$$

- Define $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$. Then $(\mathbb{Z}_5, + \pmod 5)$ is a group under usual integer addition. We check,

$$\forall a \in \mathbb{Z}_5, \ a + 0 = a; \ a + (-a) = 0, \ -a = 5 - a$$

- Also, $(\mathbb{Z}_5^*, \cdot \pmod 5)$ is a group since,

$$1^{-1} = 1; \ 2^{-1} = 3; \ 3^{-1} = 2; \ 4^{-1} = 4;$$

- Thus $(\mathbb{Z}_5, +, \cdot)$ is a field (division is well defined). **But** we would like to work in finite structures of size 2^n !

More complex structures - Rings

- We need two algebraic operations "+" and "·" on a set.

Definition: A set R together with "+" and "·" is a ring if,

1. $(R, +)$ is abelian group with $\mathbf{0}$ as "additive" identity.
2. R is closed under "·" and $\mathbf{1} \neq \mathbf{0} \in R$, $\mathbf{1}$ is multiplicative identity
3. For all $a, b, c \in R$ we have $a \cdot (b + c) = a \cdot b + a \cdot c$. (Distributivity)

Definition: Let $(R, +)$ be abelian group. If $(R \setminus \mathbf{0}, \cdot)$ is a group then $(R, +, \cdot)$ is called a *field*.

Examples of Rings

- An important example of a ring is a polynomial ring over $\mathbb{Z}_2 = \{0, 1\}$
- Its elements are formal polynomials of the form,

$$f(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n; \quad a_i \in \mathbb{Z}_2$$

- This concept can be generalized to several indeterminates $x_1, \ldots, x_n$
- Then we get a ring $\mathbb{Z}_2[x_1, \ldots, x_n]$-important for the study of Boolean functions. In a ring there are zero divisors, i.e. $fg = 0$ for nonzero $f, g \in \mathbb{Z}_2[x_1, \ldots, x_n]$ (algebraic attacks).

Construction of finite fields of nonprime order

- The goal is to construct a structure (set) having 2^n elements, say $\mathbb{F}_{2^n}$, where two operations "+" and "·" are well defined. In addition, $(\mathbb{F}_{2^n}, +)$ should be abelian group and $(\mathbb{F}_{2^n} \setminus 0, \cdot)$ a group.

Definition: A polynomial $f(x) \in \mathbb{F}_2[x]$ is said to be irreducible if $f(x) = g(x)h(x)$ implies that either g or h is a constant polynomial.

- E.g. $f(x) = x^3 + x + 1$ is irreducible polynomial over $\mathbb{F}_2$ whereas,

$$r(x) = x^3 + x^2 + x + 1 = (x^2 + 1)(x + 1)$$

is reducible.

Construction of finite fields of nonprime order cont.

- The degree of $f(x) = \sum_{i=0}^{n} a_i x^i$ is the largest i for which $a_i \neq 0$.

- Then one can prove that using an irreducible f of degree n we construct a field of 2^n elements,

$$\mathbb{F}_2[x]/ \pmod{f(x)} = \{ \text{ all polynomials of degree less than } n\},$$

- Operations on polynomials $p(x), q(x) \in \mathbb{F}_2[x]/ \pmod{f(x)}$ are:

$$\begin{aligned} p(x) + q(x) &= p(x) + q(x) \pmod{f(x)} \\ p(x) \cdot q(x) &= p(x) \cdot q(x) \pmod{f(x)} \end{aligned}$$

Examples of construction

- Construction of $\mathbb{F}_{2^3}$ using $f(x) = x^3 + x + 1$, f primitive that is x is generator of multiplicative group.

$$\begin{aligned}
x^0 &= 1 \\
x^1 &= x \\
x^2 &= x^2 \\
x^3 &= x + 1 \pmod{x^3 + x + 1} \\
x^4 &= x^2 + x \pmod{x^3 + x + 1} \\
x^5 &= x^2 + x + 1 \pmod{x^3 + x + 1} \\
x^6 &= x^2 + 1 \pmod{x^3 + x + 1} \\
x^7 &= 1 \pmod{x^3 + x + 1}
\end{aligned}$$

- Can verify that for instance $x^5 = (x^2 + 1)(x^3 + x + 1) + x^2 + x + 1$.

Polynomials over finite fields

- A polynomial over finite field is a formal expression ,
$$F(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n,$$
where $a_i \in \mathbb{F}_q$ and x is indeterminate.

- A function over finite field $\mathbb{F}_{2^n}$ is the evaluation of polynomial,
$$F(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{2^n-1}x^{2^n-1}, \; a_i \in \mathbb{F}_{2^n}.$$

Example: Most commonly used primitive is symmetric S-box, mapping n to n binary bits. Inverse S-box in AES,
$$F(x) = x^{-1} = x^{2^n-2}, \quad x \in \mathbb{F}_{2^n}$$

Functions over finite fields

- Fixing the basis of the field,say polynomial $(1, \alpha, \ldots, \alpha^{n-1})$, we get *isomorphic representation of vector space and finite field*. That is, any element $x \in \mathbb{F}_{2^n}$ can be written as $x = \sum_{i=0}^{n-1} x_i \alpha^i$, $x_i \in \mathbb{F}_2$.

$$x \in \mathbb{F}_{2^n} \overset{F}{\mapsto} F(x) \in \mathbb{F}_{2^n},$$

$$(x_0, \ldots, x_{n-1}) \in \mathbb{F}_2^n \overset{F}{\mapsto} (f_1(x_0, \ldots, x_{n-1}), \ldots, f_n(x_0, \ldots, x_{n-1})) \in \mathbb{F}_2^n.$$

$$F = (f_1, f_2, \ldots, f_n)$$

- We can consider a collection of n Boolean functions instead of F.

The main cryptographic criteria for functions over finite fields

- Nonlinearity, resistance to linear cryptanalysis
- Differential properties, resistence to differential cryptanalysis.
- High algebraic degree and permutation property

• **Fundamental result:** Algebraic degree of $F(x) = \sum_{i=0}^{2^n-1} a_i x^i$ is given by the highest Hamming weight of i for which $a_i \neq 0$.

• Thus, $F(x) = x^{-1} = x^{2^n-2}$ over $\mathbb{F}_{2^n}$ is of algebraic degree $n-1$ as $2^n - 2 = \underbrace{(1, 1, 1, \ldots, 1, 0)}_{n}$.

Linear combinations of component functions $f_1, \ldots, f_n$ of degree $n-1$.

Example of representation

- Let $F(x) = x^3$ over $\mathbb{F}_{2^3}$ defined by primitive polynomial $p(x) = x^3 + x + 1$ over $\mathbb{F}_2$. Let α be primitive element of $\mathbb{F}_{2^3}$, i.e. $\alpha^3 = \alpha + 1$.
- Then the component functions are derived as,

$$\begin{aligned}
F(x) &= x^3 = (x_0 + \alpha x_1 + \alpha^2 x_2)^3 = \\
&= (x_0 + \alpha x_1 + \alpha^2 x_2)(x_0 + \alpha x_1 + \alpha^2 x_2)^2 = \\
&= (x_0 + \alpha x_1 + \alpha^2 x_2)(x_0 + \alpha^2 x_1 + \alpha^4 x_2) \overset{\alpha^3=\alpha+1}{=} \dots \\
&= (x_0 + x_1 + x_2 + x_1 x_2) + \alpha(x_1 + x_0 x_1 + x_0 x_2) + \alpha^2(x_2 + x_0 x_1) \\
&= 1 \cdot f_1(x_0, x_1, x_2) + \alpha f_2(x_0, x_1, x_2) + \alpha^2 f_3(x_0, x_1, x_2).
\end{aligned}$$

Trace function

Definition: For $\alpha \in \mathbb{F} = \mathbb{F}_{q^m}$ and $\mathbb{K} = \mathbb{F}_q$, the *trace* $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)$ of α over $\mathbb{K}$ is defined by,

$$\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha) = \alpha + \alpha^q + \cdots + \alpha^{q^{m-1}}.$$

- It can be proved that $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)$ is always an element of $\mathbb{K} = \mathbb{F}_q$.

  Proof: $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha) \in \mathbb{K} \Leftrightarrow \mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)^q = \mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)$. But,

  $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)^q = (\alpha + \alpha^q + \cdots + \alpha^{q^{m-1}})^q = \alpha^q + \cdots + \alpha^{q^{m-1}} + \alpha = \mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha)$, since $\alpha^{q^m} = \alpha$. ∎

- Trace is a linear operator $\mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha + \beta) = \mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\alpha) + \mathrm{Tr}_{\mathbb{F}/\mathbb{K}}(\beta)$

• Need Trace to derive good Boolean functions.

Trace of the function x^3 - example

- Any element $a \in \mathbb{F}_{2^3}$ can be written as,

$$a = a_0 1 + a_1 x + a_2 x^2; \quad a_i \in \mathbb{F}_2.$$

- By properties of trace function

$$\mathrm{Tr}(a_0 1 + a_1 x + a_2 x^2) = a_0 \mathrm{Tr}(1) + a_1 \mathrm{Tr}(x) + a_2 \mathrm{Tr}(x^2).$$

Then,

$$\begin{aligned}
\mathrm{Tr}(1) &= 1 + 1 + 1 = 1 \\
\mathrm{Tr}(x) &= x + x^2 + x^4 = x + x^2 + x(x+1) = 0 \\
\mathrm{Tr}(x^2) &= \mathrm{Tr}(x) = 0
\end{aligned}$$

Representation of the mapping $\mathrm{Tr}(x^3)$

x	(x_0, x_1, x_2)	x^3	$x^3 \pmod{x^3 + x + 1}$	(y_0, y_1, y_2)	$\mathrm{Tr}(x^3)$
0	(0,0,0)	0	0	$(0,0,0)$	0
1	(1,0,0)	1	1	$(1,0,0)$	1
x	(0,1,0)	x^3	$x+1$	$(1,1,0)$	1
x^2	(0,0,1)	x^6	x^2+1	$(1,0,1)$	1
x^3	(1,1,0)	x^9	x^2	$(0,0,1)$	0
x^4	(0,1,1)	x^{12}	x^2+x+1	$(1,1,1)$	1
x^5	(1,1,1)	x^{15}	x	$(0,1,0)$	0
x^6	(1,0,1)	x^{18}	x^2+x	$(0,1,1)$	0
x^7	(1,0,0)	x^{21}	1	$(1,0,0)$	1

Walsh spectra of some trace mappings

- Good functions but problem is the hardware complexity.

Walsh spectra of $x \mapsto Tr(x^d)$ over $\mathbb{F}_{2^8}$.

$d/\#\{\alpha : \mathcal{F}_\alpha = c\}$	c											
	96	64	48	32	28	24	20	16	12	8	4	0
7		1		30				120				105
11		1	4	18				132				101
19			8	8				152				88
23		2		20				144				90
31				40				96				120
43	2	1		8				136				109
254 (-1)				5	16	36	24	34	40	36	48	17

- Note that $\mathcal{N}_f(x^{-1}) = \mathcal{N}_f(x^{31})$. But degrees are 7 respectively 5.

Suitable power mappings in cryptography

- If you consider previous table the best confusion (nonlinearity) is achieved by $F(x) = x^{-1}$ and $F(x) = x^{31}$.

- Though algebraic degree is 7 respectively 5 if $n = 8$,

$$\begin{aligned} y &= F(x) = x^{-1} / \cdot x^2 \quad \Rightarrow yx^2 = x \\ y &= F(x) = x^{31} / \cdot x \quad \Rightarrow yx = x^{32} \end{aligned}$$

$\boxed{\text{Result is quadratic equations that relate input and output bits}}$

- The internal structure of SNOW 2.0 allows description of the cipher as a system of 3706 quadratic equations with 1598 variables.

SNOW 3G a 3GPP confidentiality algorithm

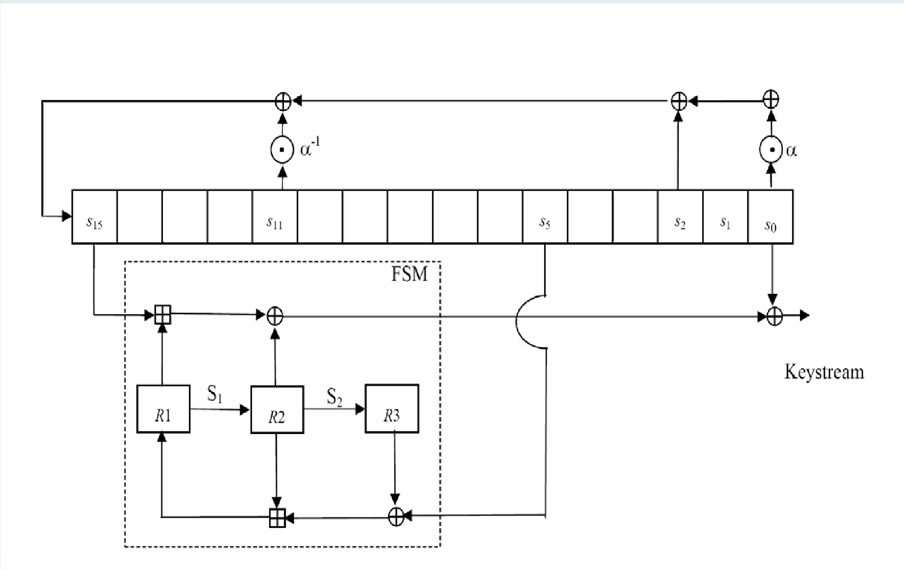
- Since A5/x algorithm does not give confidence, in 3GPP designers decided to have two algorithms based on different design principles. The choice was to use :
  - KASUMI f8 block cipher encryption scheme
  - Strenghten version of SNOW 2.0, named SNOW 3G

Once when XL, XLS or Grobner basis break AES, SNOW 2.0 . . . ?!

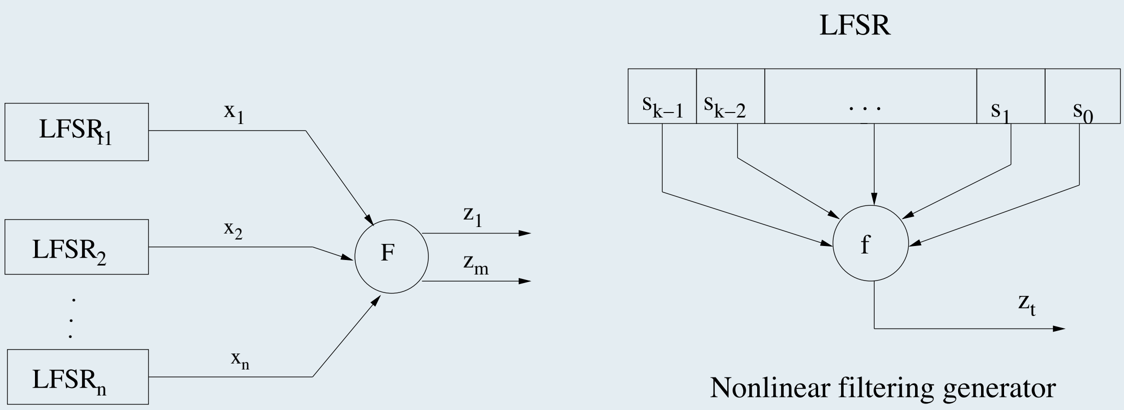
- Prof. Kaisa Nyberg asked me for a suitable polynomial function with no quadratic I/O equations.
- I/O size fixed $n = 8$ - always cubic I/O relations !
- The result was a Dickson permutation polynomial

$$P(x) = x + x^9 + x^{13} + x^{15} + x^{33} + x^{41} + x^{45} + x^{47} + x^{49}$$

Modifying SNOW 2.0 - SNOW 3G



Vectorial functions $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ in stream ciphers



- Increase the throughput by generating several keystream bits at the time $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ (similar to e.g. Blum-Blum-Shub)

Estimating the size of the output

- For these schemes clearly $m \ll n$.
- For instance taking permutation $n = m$ each output directly reveals secret state bits.

How far we can go in decompression ?

- For standard choice IV=K=128 bits, m slightly larger than $n/2$ leads to attacks of complexity less than 2^{128}.
- **Idea:** Consider the reduced preimage space,

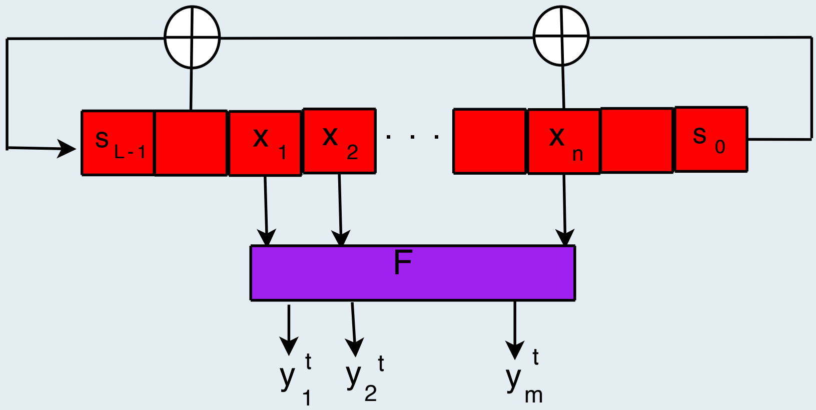
$$|S_y| = |\{x : F(x) = y\}| = 2^{n-m}.$$

- Observe output blocks $y^{t_1}, \ldots, y^{t_c}$ so that $c \times n > L$, using $L = 2K$.
- Solve $2^{(n-m)c}$ linear systems in time $2^{(n-m)c} L^3 = 2^K L^3$.

Deriving linear systems for filtering generator

• Given y^{t_i} we know $x \in S_{y^{t_i}}$. Guessing $x = (x_1^*, \ldots, x_n^*) \in S_{y^{t_i}}$ one gets n linear equations,

$$x_i = \sum_{j=0}^{L-1} a_j s_j; \quad i = 1, \ldots, n$$



Cryptographic criteria for $\mathbb{F}_{2^n} \to \mathbb{F}_{2^m}$

• For Boolean functions degree defined as the largest length of monomials,

$$f(x_1, \ldots, x_4) = x_1x_2x_3 + x_2x_4 + x_1 \quad deg(f) = 3$$

• Assume

$$\begin{aligned} y_1 = f_1(x_1, \ldots, x_4) &= x_1x_2x_3 + x_2x_4 + x_1 \\ y_2 = f_2(x_1, \ldots, x_4) &= x_1x_2x_3 + x_2x_4 + x_2 \end{aligned}$$

• Then consider $y_1 + y_2 = x_1 + x_2$ and easily break the scheme.

Nonlinearity, resiliency and degree defined w.r.t. $\sum_{i=1}^{m} a_i f_i$, $a \neq 0$.

An example of construction

- Assume that we want to construct 1-resilient function $F: \mathbb{F}_2^6 \to \mathbb{F}_2^3$ by concatenating linear functions on $\mathbb{F}_2^4$.

f_1	f_2	f_3	$\sum f_i$
x_1+x_2	x_2+x_3	x_1+x_3	0
x_1+x_3	x_1+x_4	x_2+x_3	x_2+x_4
x_2+x_3	$x_2+x_3+x_4$	x_3+x_4	x_3
$x_1+x_3+x_4$	x_1+x_4	x_2+x_4	$x_2+x_3+x_4$

- E.g. the ANF of function f_1 is,

$$\begin{aligned} f_1(y,x) &= (y_1+1)(y_2+1)[x_1+x_2]+y_1(y_2+1)[x_1+x_3]+ \\ &+ (y_1+1)y_2[x_2+x_3]+y_1y_2[x_1+x_3+x_4]. \end{aligned}$$

Binary linear codes

- f_1, f_2, f_3 are all 1-resilient functions of degree $d=3$. On the other hand, $f_1+f_2+f_3$ is of degree 2 and it is not even balanced.

- Solution is to use codewords of a binary linear code.

Definition: A binary linear $[k,m,t]$ code, C, is an m-dimensional subspace of $\mathbb{F}_2^k$ s.t. for all $c \in C \setminus \{\mathbf{0}\}$

$$wt(c) \geqslant t+1.$$

- In other words, there is a basis of C, say $C = < c_1, \ldots, c_m >$, $c_i \in \mathbb{F}_2^k$, s.t. any $c \in C$ can be written as,

$$c = \sum_{i=1}^{m} a_i c_i, \; a_i \in \mathbb{F}_2.$$

Example of a binary linear code

- Let us consider 3 binary vectors in $\mathbb{F}_2^4$, $c_1 = (1,1,0,0)$, $c_2 = (1,0,1,0)$, $c_3 = (0,1,0,1)$. Then $C =< c_1, c_2, c_3 >$ is a set of 8 vectors,

$$\begin{aligned}
\mathbf{0} &= (0,0,0,0)\\
c_1 &= (1,1,0,0)\\
c_2 &= (1,0,1,0)\\
c_3 &= (0,1,0,1)\\
c_1 + c_2 &= (0,1,1,0)\\
c_1 + c_3 &= (1,0,0,1)\\
c_2 + c_3 &= (1,1,1,1)\\
c_1 + c_2 + c_3 &= (0,0,1,1)
\end{aligned}$$

Using binary linear codes to construct F

- Let $< c_1, \ldots, c_m >$ be a basis of a binary linear $[k, m, t+1]$ code C. Then the component functions $f_1, \ldots, f_m \in {}_k$ may be defined on the same k-dimensional subspace τ as,

$$f_i^{\tau}(x_1, \ldots, x_k) = c_i \cdot x = c_{i,1}x_1 + \cdots + c_{i,k}x_k,\ i = 1, \ldots, m;\quad c_i \in \mathbb{F}_2^k$$

$$\boxed{\text{If } F : \mathbb{F}_2^n \to \mathbb{F}_2^m \text{ - need to define } 2^{n-k} \text{ linear functions for each } f_i \text{ !}}$$

- Then $\sum_{i=1}^m a_i f_i^{\tau}(x)$ is a linear function of weight $\geqslant t+1$ for any nonzero choice of a_i's,

$$\sum_{i=1}^m a_i f_i^{\tau}(x) = \sum_{i=1}^m a_i c_i \cdot x = \underbrace{\left(\sum_{i=1}^m a_i c_i\right)}_{c \in C \setminus \{0\}} \cdot x.$$

Using binary linear codes to construct F cont.

- Assume that component functions of F are concatenations of **distinct** linear functions on $\mathbb{F}_2^k$. Thus we need 2^{n-k} such functions.
- Due to definition of resiliency if F is to be t-resilient then all linear combinations of linear subfunctions are t resilient on these 2^{n-k} subspaces.
- Do we need a set of 2^{n-k} *disjoint* linear codes to construct F?
- Disjoint means that for $\mathcal{C} = \{C_1, \ldots, C_{2^{n-k}}\}$,

$$C_i \cap C_j = \{0\}, \quad 1 \leqslant i < j \leqslant 2^{n-k}$$

How to use all the codewords of C

Lemma [Johansson-Pasalic '00]: Let $c_0, \ldots, c_{m-1}$ be a basis of a binary $[k, m, t+1]$ linear code C. Let β be a primitive element in $\mathbb{F}_{2^m}$ and $(1, \beta, \ldots, \beta^{m-1})$ be a polynomial basis of $\mathbb{F}_{2^m}$. Define a bijection $\phi : \mathbb{F}_{2^m} \mapsto C$ by

$$\phi(a_0 + a_1\beta + \cdots a_{m-1}\beta^{m-1}) = a_0c_0 + a_1c_1 + \cdots a_{m-1}c_{m-1}.$$

Consider the matrix

$$A^* = \begin{pmatrix} \phi(1) & \phi(\beta) & \ldots & \phi(\beta^{m-1}) \\ \phi(\beta) & \phi(\beta^2) & \ldots & \phi(\beta^m) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(\beta^{2^m-2}) & \phi(1) & \ldots & \phi(\beta^{m-2}) \end{pmatrix}.$$

For any linear combination of columns (not all zero) of the matrix A^*, each nonzero codeword of C will appear exactly once.

Proof of the Lemma

Proof: Since ϕ is a bijection, it is enough to show that the matrix

$$\begin{pmatrix} 1 & \beta & \dots & \beta^{m-1} \\ \beta & \beta^2 & \dots & \beta^m \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{2^m-2} & 1 & \dots & \beta^{m-2} \end{pmatrix}$$

has the property that each element in $\mathbb{F}^*_{2^m}$ will appear once in any nonzero linear combination of columns of the above matrix.

Any nonzero linear combination of columns can be written as

$$(c_0 + c_1\beta + \cdots + c_{m-1}\beta^{m-1})\begin{pmatrix} 1 \\ \beta \\ \vdots \\ \beta^{2^m-2} \end{pmatrix}, \quad c_0, \dots, c_{m-1} \in \mathbb{F}_2$$

Application of the Lemma

- Let β be be a primitive element in $\mathbb{F}_{2^3}$ and $(1, \beta, \beta^2)$ be a polynomial basis of $\mathbb{F}_{2^m}$. For $c_0 = (1,1,0,0)$, $c_1 = (1,0,1,0)$, $c_2 = (0,1,0,1)$, and $\beta^3 + \beta + 1 = 0$ we compute A^*,

$$A^* = \begin{pmatrix} c_0 & c_1 & c_2 \\ c_1 & c_2 & c_0+c_1 \\ c_2 & c_0+c_1 & c_1+c_2 \\ c_0+c_1 & c_1+c_2 & c_0+c_1+c_2 \\ c_1+c_2 & c_0+c_1+c_2 & c_0+c_2 \\ c_0+c_1+c_2 & c_0+c_2 & c_0 \\ c_0+c_2 & c_0 & c_1 \end{pmatrix}; \overbrace{\begin{pmatrix} c_0+c_1 \\ c_1+c_2 \\ c_2+c_0+c_1 \\ c_0+c_2 \\ c_0 \\ c_1 \\ c_2 \end{pmatrix}}^{A_1^*+A_2^*}.$$

Application of the Lemma

- Let us construct 1-resilient $F : \mathbb{F}_2^6 \to \mathbb{F}_2^3$ concatenating linear functions in 4 variables. We may take any 4 rows of A^* (here we take first 4 rows) and define F (using $c_0 = (1,1,0,0)$, $c_1 = (1,0,1,0)$, $c_2 = (0,1,0,1)$) as,

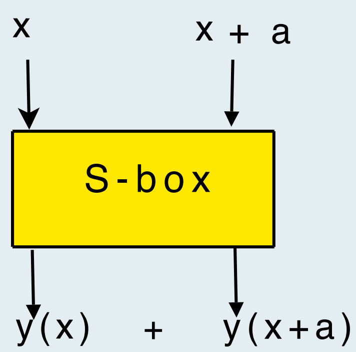
$$F = \begin{pmatrix} c_0 \cdot x & c_1 \cdot x & c_2 \cdot x \\ c_1 \cdot x & c_2 \cdot x & (c_0 + c_1) \cdot x \\ c_2 \cdot x & (c_0 + c_1) \cdot x & (c_1 + c_2) \cdot x \\ (c_0 + c_1) \cdot x & (c_1 + c_2) \cdot x & (c_0 + c_1 + c_2) \cdot x \end{pmatrix}.$$

$$F = \begin{pmatrix} x_1 + x_2 & x_1 + x_3 & x_2 + x_4 \\ x_1 + x_3 & x_2 + x_4 & x_2 + x_3 \\ x_2 + x_4 & x_2 + x_3 & x_1 + x_2 + x_3 + x_4 \\ x_2 + x_3 & x_1 + x_2 + x_3 + x_4 & x_3 + x_4 \end{pmatrix}.$$

Differential cryptanalysis

- Particularly important for block ciphers

• **Idea:** Compute the distribution tables for S-boxes, i.e. certain input differences can result in highly nonuniform distribution of the output XOR of an S-box.



Differential properties of S-boxes

- We use a polynomial representation, i.e. $F(x) = \sum_{i=0}^{2^n-1} a_i x,\ a_i \in \mathbb{F}_{2^n}$.
- Differential properties of F counts the number of solutions to

$$F(x+a) + F(x) = b \quad a \in \mathbb{F}_{2^n}^*, b \in \mathbb{F}_{2^n}. \tag{1}$$

  F is called *almost perfect nonlinear*(APN) if each equation (1) has at most two solutions in $\mathbb{F}_{2^n}$. Highest resistance to differential cryptanalysis.

Example: Find the differential properties of $F(x) = x^3$ in $GF(2^n)$.

$$F(x+a) + F(x) \;=\; (x+a)^3 + x^3 = (\cancel{x}^3 + ax^2 + a^2x + a^3) + \cancel{x}^3 = b.$$

Quadratic equation, either two or no solutions in $GF(2^n)$. x^3 is APN !

Kloosterman sums and inverse function

- If $\mathbb{F}_q$ is a finite extension over $\mathbb{F}_2$ then *Kloosterman sum* is defined,

$$K(a,b) = \sum_{x \in \mathbb{F}_q} (-1)^{\mathrm{Tr}(ax+bx^{-1})}; \quad a,b \in \mathbb{F}_q,$$

  where we take $0^{-1} = 0$.

- This is exactly the Walsh spectra of x^{-1}. The elements a chooses linear functions and b different linear combinations of the output functions of x^{-1}.

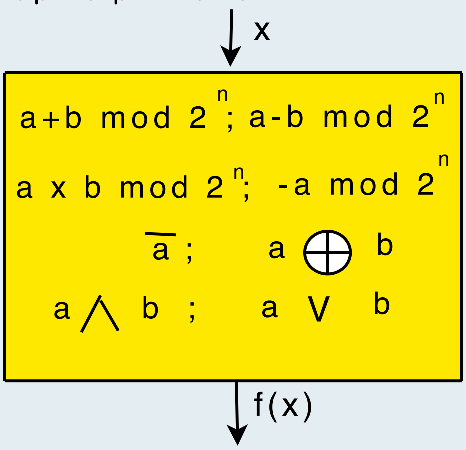
- Hard to prove anything for similar sums, except some bounds:

$$|K(a,b)| \leqslant 2q^{\frac{1}{2}},$$

  for any $a, b \in \mathbb{F}_q$ and $b \neq 0$. So, $Nl(x^{-1}) \geqslant 2^{n-1} - 2^{\frac{n}{2}}$.

T-functions

- Introduced in 2003 by A. Klimov and A. Shamir
- The pupose is combine primitive machine instructions (operations) to obtain cryptographic primitive.



Basic properties of T-functions

- All processors support the 8 operations - efficient software implementation.
- Common feature for all operations - no propagation (triangular function) from left to right (LSB is zero bit, MSB is $(n-1)$-th bit). That is, i-th bit of output $[f(x)]_i$ depends only on input bits $x_0, \ldots, x_i$

Example: Recall again that $d = a + b \pmod{2^n}$ using $c_0 = 0$,

$$d_i = a_i \oplus b_i \oplus c_i; \; i = 0, \ldots, n-1,$$

Thus

$$d_0 = a_0 \oplus b_0, d_1 = a_1 + b_1 + c_1(a_0, b_0), d_2 = a_2 + b_2 + c_2(a_0, b_0, a_1, b_1) \ldots$$

Invertibility of T-functions

- Often it is desirable to use invertible building blocks; T-functions are used in block ciphers as well (RC6);
- Polynomials over the ring of integers modulo 2^n are easier to characterize than polynomials over fields!

Theorem [Rivest, 1999]: Let $P(x) = a_0 + a_1x + \ldots + a_dx^d$ be a polynomial with integral coefficients. Then $P(x)$ is a permutation polynomial modulo 2^n, $n > 2$ IF AND ONLY IF

$$a_1 \text{ is odd} \; ; \; (a_2 + a_4 + \ldots) \text{ is even} \; ; \; (a_3 + a_5 + \ldots) \text{ is even} \; .$$

For instance $x \mapsto x + 2x^2 \pmod{2^n}$ is permutation, but the theorem does not cover,

$$x \mapsto x \oplus (x^2 \vee 1)$$

Cycle length of T-functions

- Not all invertible T-functions has good cycle structure.

Example: Consider $x \mapsto x + 2x^2 \pmod{16}$,

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	10	5	4	7	14	9	8	11	2	13	12	15	6	1

A permutation but several cycles, including fixed points in red, e.g. $4 \mapsto 4 \mapsto 4 \cdots$

Cannot construct PRNG similar to LCG $x_{i+1} = f(x_i)$!

- Important but (nonconstructive) result states
  - T-function is invertible IFF it can be represented as $c + x + 2v(x)$
  - A single cycle IFF T-function written as $1 + x + 2(v(x+1) - v(x))$

where $v(x)$ is some T-function.

Single cycle T-functions

- Idea similar as for LFSR-based stream ciphers:
  - LFSR has two cycles, a long period $2^L - 1$, and all-zero
  - T-function can be constructed to have a single 2^n cycle.
- Protect from accidental derivation of all-zero initial state for LFSR.

How do we construct efficient single cycle T-functions ?

- Selecting $v(x) = x^2$ still gives 7 operations (inefficient) using

$$T(x) = 1 + x + 2(v(x+1) - v(x))$$

- For instance $x \mapsto x + (x^2 \vee 5) \pmod{2^n}$ is invertible single cycle mapping - only 3 operations.

Application of $x \mapsto x + (x^2 \vee 5) \pmod{2^n}$ as PRNG

- **PRIMITIVE:** Use $x \mapsto x + (x^2 \vee 5) \pmod{2^n}$ as a substitute for LFSR or LCG rather than stand-alone cipher.
- MSB are much stronger in this application (depends on more bits), take $m \ll n$ MSB bits as output.

Any PRNG can be attacked with TMD attack in complexity $2^{n/2}$!

1. Precompute outputs for 2^t random states of PRNG (of sufficient length to uniquely determine the state), sort the list
2. Observe 2^d actual outputs
3. If $t + d = n$ the probability of collision is 0.5. Optimal $t = d = n/2$.

Security of $x \mapsto x + (x^2 \vee 5) \pmod{2^n}$ T-function

- We can represent $x \in [0, 2^n]$ as,

$$x = 2^{2n/3} x_u + 2^{n/3} x_v + x_w, \quad x_u, x_v, x_w \in [0, 2^{n/3}]$$

Suppose that $x_w = 0$. Then

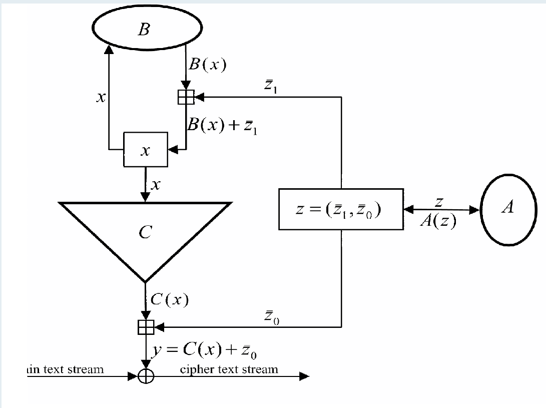
$$\begin{aligned} 2^{2n/3} x_u + 2^{n/3} x_v &\mapsto 2^{2n/3} x_u + 2^{n/3} x_v + ((2^{2n/3} x_u + 2^{n/3} x_v)^2 \vee 5) = \\ &= 2^{2n/3}(x_u + x_v^2) + 2^{n/3} x_v + 5 \pmod{2^n}. \end{aligned}$$

$\boxed{\text{Difference between MSB } x_v^2 \text{ - leads to a TMD attack in } 2^{n/3}}$.

• Most of the stream cipher designs have failed to resist cryptanalysis.
• Software implementation is not efficient as claimed (especially multiplication).

eSTREAM proposal ABC

• Broken several times, 7 different cryptanalysis - T-function design reminds a dedicated hash function design !!



• A is an LFSR of length 64; B is a single cycle T function; C is a filtering function (table look-ups, $\boxplus$, and $\ggg$).

Chapter 9

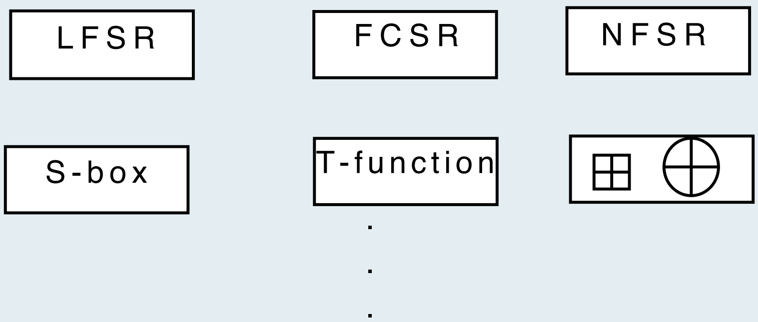
Algebraic Attacks

Content of this chapter:

- Algebraic attacks preliminary.
- Algebraic immunity of Boolean functions.
- Vectorial Boolean functions - multivariate input/output equations.
- Fast and probabilistic algebraic attacks.
- S-boxes and multivariate equations, attacks using Grobner basis.

Summary on stream cipher primitives

- Goal is to design keystream generator with:
  - Two inputs: KEY and IV
  - OUTPUT: Keystream sequence of long period and good statistical properties.
  - Using standard building blocks as given below.



Algebraic attacks

Known from Shannon theory but revisited in 2001.

– Find equations (on any cipher) with the key (state) bits as unknowns.
– Fill in the known variables and constants
– Solve the equations

Problems :

– Non-linear equations (of high degree)
– Finding the equations highly dependent on the cipher
– Finite field algebras

Setting up equations - Shannon's attack

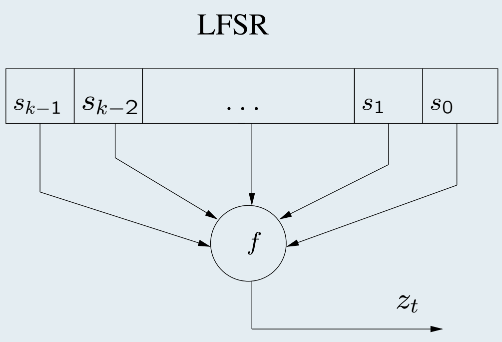
- For any cipher encyphering can be seen as $E = f(K, M)$.
- Given $M = m_1, m_2, \ldots, m_s$ and $E = c_1, c_2, \ldots, c_s$ cryptanalyst can set up equations for different key elements $k_1, k_2, \ldots, k_r$

$$\begin{aligned} c_1 &= f_1(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r) \\ c_2 &= f_2(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r) \\ &\vdots \\ c_s &= f_s(m_1, m_2, \ldots, m_s; k_1, k_2, \ldots, k_r). \end{aligned}$$

- Each equation must be complex in k_i and involve many of them.

Algebraic attacks on LFSR based stream siphers

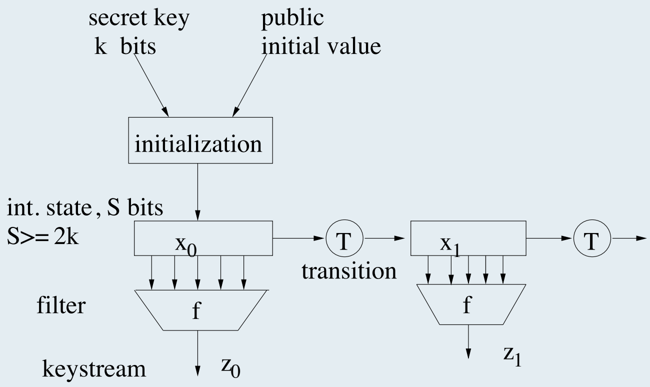
- For these ciphers $c_i = m_i \oplus z_i$ (additive ciphers), z_i keystream.



Nonlinear filtering generator

- Known plaintext attack means that keystream z_t is known, in addition $z_t = f(s_0, \ldots, s_{k-1})$.

LFSR filters - example



- Initial state $x_0 = s_0, \ldots, s_{S-1}$. Then $x_t = L^t(s_0, \ldots, s_{S-1})$.
- **Problem.** Recover $s_0, \ldots, s_{S-1}$ from $z_0, \ldots, z_{N-1}$,

$$z_t = f(x_t) = f \circ L^t(s_0, \ldots, s_{S-1}),\ 0 \leq t \leq N-1.$$

Shannon's attack for linear transition ciphers

- Set up the enciphering equations:

$$\begin{aligned} z_0 &= f(s_0, s_1, \ldots, s_{K-1}) \\ z_1 &= f \circ L(s_0, s_1, \ldots, s_{K-1}) \\ &\vdots \\ z_t &= f \circ L^t(s_0, s_1, \ldots, s_{K-1}). \end{aligned}$$

- System of equations in K variables of **degree** $d = deg(f)$.
  The number of terms is

$$\leq \sum_{i=0}^{d} \binom{K}{i} \approx \frac{K^d}{d!}$$

- Observe more than $\frac{K^d}{d!}$ bits and solve system using **linearization** (turn nonlinear system to linear) in **complexity** $\left(\frac{K^d}{d!}\right)^3$.

Algebraic attacks: Linearization

- Basic linearization algorithm:
  - System is overdefined - more equations than monomials
  - Replace each monomial with new variable
  - Solve linear system

$$\begin{array}{l} s_0 \oplus s_1 \oplus s_1 s_2 = 0 \\ s_0 s_1 \oplus s_0 s_2 \oplus s_1 s_2 = 1 \\ s_1 \oplus s_0 s_1 = 0 \\ s_0 s_1 \oplus s_1 s_2 \oplus s_1 = 0 \\ s_0 s_1 \oplus s_0 = 0 \\ s_0 s_1 \oplus s_1 s_2 = 1 \end{array} \rightarrow \begin{array}{l} t = s_0 s_1 \\ u = s_0 s_2 \\ v = s_1 s_2 \end{array} \rightarrow \begin{array}{l} s_0 \oplus s_1 \oplus v = 0 \\ t \oplus u \oplus v = 1 \\ s_1 \oplus t = 0 \\ s_1 \oplus t \oplus v = 0 \\ t \oplus s_0 = 0 \\ t \oplus v = 1 \end{array} \rightarrow \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ t \\ u \\ v \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

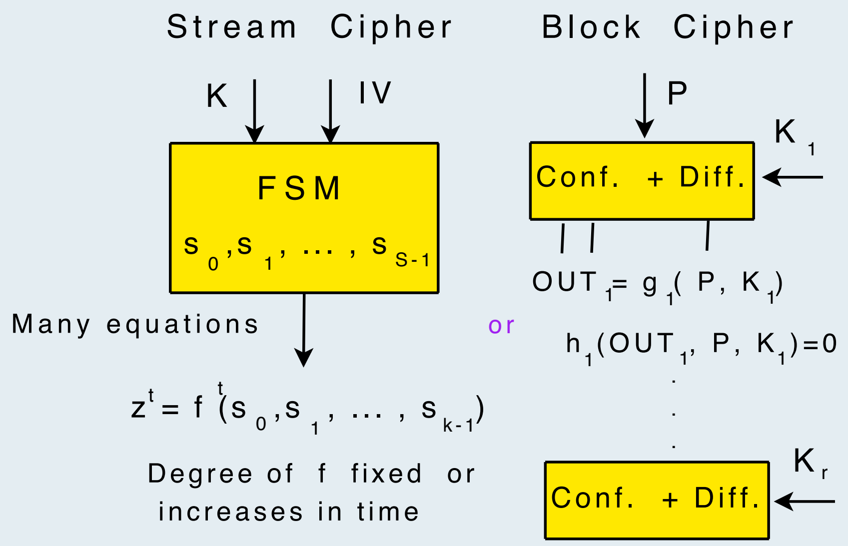
Algebraic attacks preliminaries

- Can we decrease the degree of the system ?
- If we can set up a true system of lower degree $r < d$ complexity becomes smaller,

$$\left(\frac{S^r}{r!}\right)^3 \leftarrow \left(\frac{S^d}{d!}\right)^3$$

- How do we decrease the degree of the system ?
- What ciphers are vulnerable to this attack ?

Block versus Stream ciphers



- g_i explicit functions, e.g. x^{-1} as an S-box gives degree $70 = 10 \times 7$ after 10 rounds. h_i implicit equations, output values new variables

Annihilators of Boolean function

- Let $f(x_3, x_2, x_1) = x_1x_2 + x_2x_3$.

x_3	x_2	x_1	$f(x)$	$g(x)$
0	0	0	0	$*$
0	0	1	0	$*$
0	1	0	0	$*$
0	1	1	1	0
1	0	0	0	$*$
1	0	1	0	$*$
1	1	0	1	0
1	1	1	0	$*$

- Assign "$*$" to get **annihilator** g, $f(x)g(x) = 0$, of low degree !
- For instance $g(x) = 1 + x_2$ gives

$$f(x)g(x) = [x_2(x_1 + x_3)][1 + x_2] = x_2(x_1 + x_3) + x_2(x_1 + x_3) = 0$$

An example of a bad design- Toyocrypt

- **Toyocrypt** uses LFSR of length 128 to generate $z_t = f(\mathbf{s}^t)$,

$$f(s_0, \ldots, s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42}$$
$$+ \quad s_1s_2s_9s_{12}s_{18}s_{20}s_{23}s_{25}s_{26}s_{28}s_{33}s_{41}s_{42}s_{51}s_{53}s_{59} + \prod_{i=0}^{62} s_i.$$

- Now $T \approx \binom{128}{63} \approx 2^{124}$ which gives attack $Compl = 2^{124^3} = 2^{372}$.

- But $f(\mathbf{s})(1+s_{23})$ is of degree $\mathbf{3}$! System $f(\mathbf{s}^t)(1+s_{23}) = z_t(1+s_{23})$.

  Then $T \approx \binom{128}{3} = 2^{18}$, and attack complexity $\mathbf{2^{54}}$.

Algebraic attacks- decreasing the degree of f

- Set of annihilators is exactly $Ann(f) =< 1 + f >$, ideal spanned by $1 + f$.

- **Idea of attack:** Find annihilators of degree less than $deg(f)$. Observing $z_t = 1$,

$$f(x_t) = 1 \Rightarrow \underbrace{f(x_t)g(x_t)}_{=0} = g(x_t) \Rightarrow g \circ L^t(s_0, s_1, \ldots, s_{S-1}) = 0.$$

- Similarly for $h \in Ann(1 + f)$and $z_t = 0$,

$$h(x_t)(1 + f(x_t)) = 0 \Rightarrow h \circ L^t(s_0, s_1, \ldots, s_{S-1}) = 0.$$

- Solve a system of equations of degree $deg(g) = deg(h) < deg(f)$.

Amount of keystream needed

- Denote by $g_1, \ldots, g_u$ linearly independent annihilators of f
- Similarly $h_1, \ldots, h_u$ linearly independent annihilators of $1 + f$
- Each keystream bit s_t gives rise to u linearly independent equations of degree $d = deg(g_i) = deg(h_i)$
- One annihilator enough, but keystream sequence needed reduced by factor u,

$$\text{Nmb. of keystream bits } \approx \binom{S}{d}/u.$$

Protection to algebraic attacks for linear transition ciphers

- Find a minimum degree for which complexity of attacks is larger than brute force attack.

Example: For $S = 2k = 256$ to protect against Shannon's attack is $deg(g) \geq 7$ as

$$\left(\sum_{i=0}^{7} \binom{S}{i}\right)^3 \approx 2^{129}.$$

Meaning no annihilators of degree < 7 for filtering function !

Annihilator g of $deg(g) = 3$ decreases complexity to 2^{63}.

Bounds on degree of annihilators

$$\begin{array}{c|c|} & 1\ x_1\ \cdots x_n\ x_1x_2\ \cdots\ x_{n-d+1}\cdots x_n \\ \hline x : f(x) = 1 & \\ wt(f) = 2^{n-1} & RM^f(d,n) \\ & \\ \hline \end{array}$$

all monomials of degree $\leq d$

- (Courtois & Meier EC 03)There exists $g \neq 0$, with $deg(g) \leq d$ if,

$$wt(f) < \sum_{i=0}^{d} \binom{n}{i}.$$

- More columns than rows $\Longrightarrow$ linear dependency of columns.
- Consequently, always annihilators of degree $\lceil \frac{n}{2} \rceil$ for $wt(f) = 2^{n-1}$.

Computing annihilators

- How to find these annihilators: Find the kernel of matrix, use Gauss in complexity

$$2^{n-1}\left(\sum_{i=0}^{d} \binom{n}{i}\right)^2.$$

- Complexity of size 2^{43} already for $d = 7$ and $n = 15$.
- Eurocrypt 2003 (Meier-Pasalic-Carlet) provides better algorithm

Memory	$\frac{1}{4}\binom{n}{d} \cdot \binom{n}{d-1}$
Complexity	$\frac{1}{8}\binom{n}{d}^3$

- Currently there are even faster algorithms, e.g. Armknecht et al. Eurocrypt 2006.

Algebraic immunity

Definition: Algebraic immunity (AI) of f defined as minimum degree of $g \in \mathcal{B}_n$ such that either $fg = 0$ or $(1+f)g = 0$.

- Previous result gives upper bound $AI(f) \leq \lceil \frac{n}{2} \rceil$ for balanced functions.
- The degree of system ≥ 7 for $S = 2k = 256$ as,
$$\binom{256}{7} \approx 2^{43} \Rightarrow Compl. = (2^{43})^3 = 2^{129}.$$
- Thus function f on at least $n = 14$ variables.
- Optimized algebraic immunity when $AI(f) = \lceil \frac{n}{2} \rceil$.

Relationship between annihilators of f and $1+f$

- Turns out not to be easy- want to find $\min_{deg}\{Ann(f), Ann(1+f)\}$

Theorem: When n is odd, a balanced function f has optimized $AI = (n+1)/2$ if and only if
$$deg(Ann(f)) = deg(An(1+f)) = (n+1)/2$$

- Open problem is to determine whether there is such connection for even n.

Annihilators of f and $1+f$ - example

- For unbalanced function $f(x_1, x_2, x_3) = x_1x_2x_3$ easy to check that $g = 1 + x_1$ annihilates f, that is $fg = 0$.
- On the other hand $1 + f$ is only annihilated by f !
- The truth table of $1 + f$ is

$$\begin{aligned} 1+f &= [11111110] \\ &\Downarrow \\ g &= [00000001]. \end{aligned}$$

- Thus $deg(Ann(f)) = 1$ whereas $deg(Ann(1 + f)) = 3$.

Simulation results on AI, n odd

AI of balanced functions, n odd.

		algebraic immunity		
n	nb. trials	$\frac{n-3}{2}$	$\frac{n-1}{2}$	$\frac{n+1}{2}$
7	10,000	0	0.838	0.162
9	10,000	0	0.709	0.291
11	10,000	0	0.785	0.215
13	10,000	0.010	0.913	0.077
15	500	0.002	0.988	0.0

Main concentration for $AI \in \{\frac{n-1}{2}, \frac{n+1}{2}\}$.

Simulation results on AI, n even

AI of balanced functions, n even.

		algebraic immunity		
n	nb. trials	$\frac{n}{2}-2$	$\frac{n}{2}-1$	$\frac{n}{2}$
8	10,000	0	0	1
10	10,000	0	0	1
12	10,000	0	0.084	0.916

- Most likely that $AI = \frac{n}{2}$.
- For random balanced functions AI good on average. Need results for deterministic constructions.

Good algebraic immunity from known classes

- Dropping resiliency one may consider $Tr(x^{-1})$ over $\mathbb{F}_{2^n}$.

n	degree	nonlin.	alg. immunity
6	5	24	3
7	6	54	4
8	7	112	4
9	8	234	4
10	9	480	5
14	13	8064	6

- Slightly unoptimized but good candidate.

Attacks on real ciphers

- Mostly applicable to ciphers designed before invention of algebraic attacks (few exceptions)
- The greatest success to,
  - Toyocrypt stream cipher.
  - E_0 encryption algorithm in Bluetooth.
  - LILY-128 stream cipher

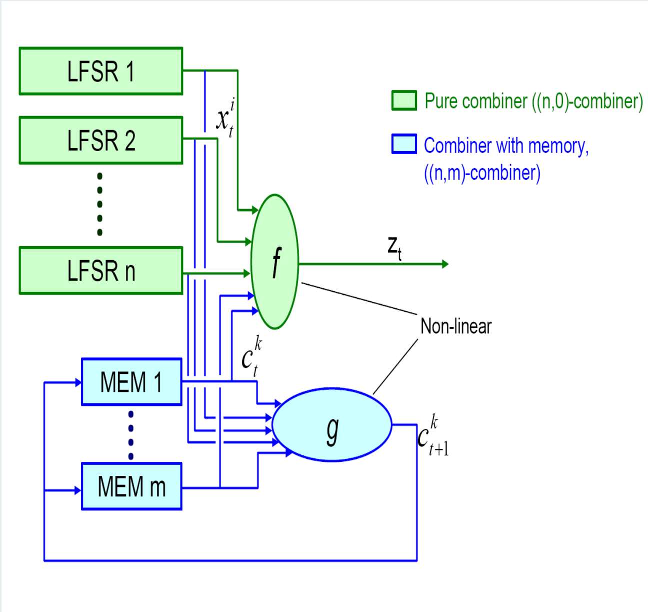
Summary for annihilators of Boolean functions

- There exist constructions of "strong" Boolean functions, unifying high degree, high nonlinearity and resiliency.

  Open problem: Propose construction which ensures optimum algebraic immunity (only annihilators of degree $\lceil \frac{n}{2} \rceil$) as well !

- To make cipher secure increase the variable space n, but trading-off against higher implementation cost (more gates-slower).

Combiners with memory



Algebraic attacks on combiners with memory

In this case $\forall t \;\; z_t = f(x_t^1, \ldots, x_t^n, c_t^1, \ldots, c_t^m)$. The LFSR inputs are still linear functions of key (state) bits,

$$z_t = f(L^t(k_1, \ldots, k_s), c_t^1, \ldots, c_t^m) = f(L^t(K), \overline{c_t}).$$

- Now we have $f(L^t(K), \overline{c_t}) \oplus z_t = 0$, but collecting keystream bits does not help- new variables $\overline{c_t}$ all the time.
- We could substitute all the c_t with a function of c_0, after all $\overline{c_{t+1}} = g(\overline{c_t})$ for all t. (c_0 can be assumed to be known to the attacker)

But: degree of equations would increase exponentially with t.

Algebraic attacks on combiners with memory II

Task: cancelling out the memory-bits from (n, m)- combiners

- Result by Armknecht and Krause in Crypto 2003:
  - There is a Boolean function H of a degree at most $\lceil\frac{n(m+1)}{2}\rceil$and an integer r strictly larger than the number of memory bits, such that:

$$\forall t : H(L^t(K), z_t, \ldots, z_{t+r-1}) = 0$$

  - There is also an algorithm for finding H.

• The problem is that even small m implies $deg(H)$ is rather high, complexity of attack is high.

Fast algebraic attacks -reducing the degree

- IDEA: Assume an system of equations $H(L^t(K), z_t, \ldots, z_{t+r-1}) = 0$ can be split into two halves,

$$H_1(L^t(K)) + H_2(L^t(K), z_t, \ldots, z_{t+r-1}) = 0$$

  where $deg(H) = deg(H_1) = d_1$ and $deg(H_2) = d_2$ with $d_1 > d_2$.

• H_1 only dependent on linear function of the secret key bits $\Longrightarrow$ after "several" clocks the system of H_1:s will be linearly dependent

$$\exists \alpha_0, \ldots, \alpha_h : \sum_{i=0}^{h} \alpha_i H_1(L^{t+i}(K)) = 0$$

• The parameter h is about $\binom{K}{d_1}$ (theory of linear recurring sequences).

- This is precomutation step - usually complexity comparable to algebraic attack.

Fast algebraic attacks - reducing the degree

We have achieved degree reduction as:

$$\forall t \qquad \sum_{i=0}^{h} \alpha_i H_1(L^{t+i}(K)) = 0$$

$$\Updownarrow$$

$$\sum_{i=0}^{h} \alpha_i H_2(L^{t+i}(K), z_t, \ldots, z_{t+r-1}) = 0$$

- Degree reduced, but number of needed consecutive keystream bits increased (dramatically).
- Assumption and efficient retrieval of coefficients α_i was proven correct for most stream ciphers by Armknecht in Oct 2004.

Fast algebraic attacks - complexity

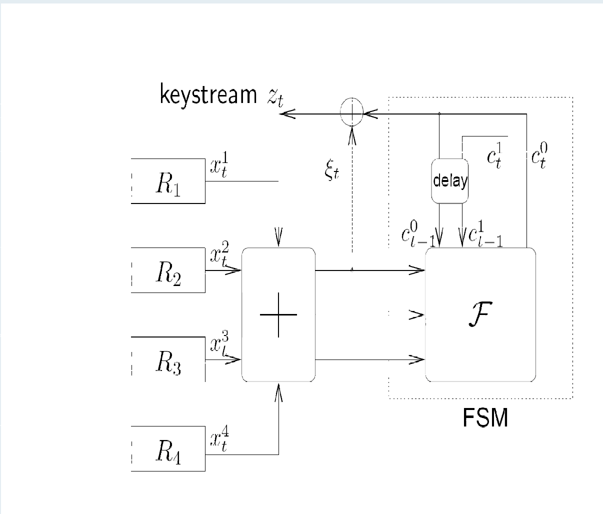
1. **Relation Search** Computing the equation[s] of type $H_1 + H_2 = 0$. At most $\binom{n}{d_1}$ steps, n number of LFSR variables.
2. **Precomputation step** Find unique coefficients $\alpha_0, \ldots, \alpha_h$ so that $\sum_{i=0}^{h} \alpha_i H_1(L^{t+i}(K), z_t) = 0$. The complexity is $h \log^2 h$, where $h = \binom{|K|}{d_1}$.
3. **Substitution step** Write $\sum_{i=0}^{h} \alpha_i H_2(L^{t+i}(K), z_t, \ldots, z_{t+r-1}) = 0$ for $E = \binom{|K|}{d_2}$ consecutive values of t, for example $t = 1, \ldots, E$. Complexity (dominating) $2Eh \log h$.
4. **Solving step** Solve these equations by linearization. It requires E^3 operations.
5. **Keystream requirement** Need $h + E \approx h$ keystream bits.

Breaking SFINKS with fast algebraic attacks

- SFINKS was a eSTREAM submission, now withdrawn. **Idea:** Take a 256 stage LFSR and nonlinear combining function $deg(f) = 15$.
- Authors were awared of existence of annihilators (algebraic immunity) of degree ≥ 6, which would give $\left(\binom{256}{6}\right)^3 = 2^{115}$, above 80 bits security level.
- But fast algebraic degree could be mounted due to existence of g, h such that $fg + h = 0$, $\quad deg(h) = 8 = d_1$; $\quad deg(g) = 2 = d_2$.

Precomputation step	2^{60}
Substitution step	2^{70}
Solving step	2^{42}
Keystream required	2^{48}

E0 summation generator



E0 is an $(n, m) = (4, 4)$ combiner; 4 memory bits.

Fast algebraic attack on E0

- **Prediction:** degree at most $10 = n(m+1)/2$, dependency of at most 5 consecutive keystream bits.

- Practice: degree 4, dependency of 4 consecutive bits

$$
\begin{aligned}
G(L^t(K), z_t, z_{t+1}, z_{t+2}, z_{t+3}) &= z_t + z_{t+1} + z_{t+2} + z_{t+3} \\
&+ \pi^2_{t+1}(z_{t+1} + z_{t+2} + z_{t+3}) \\
&+ \pi^4_{t+1} + \ldots + \pi^1_{t+2} \cdot \pi^1_{t+1} \cdot z_{t+2}(z_{t+1} + 1) \\
&+ \ldots + \pi^1_{t+3} \cdot \pi^2_{t+1} \\
&= 0.
\end{aligned}
$$

π^i_t is i-th elementary symmetric polynomial in the unknown outputs of the four LFSRs

$$\pi^i(s_1, \ldots, s_K) = \sum_{1 \le j_1 \le j_2 \le \cdots \le j_i \le K} s_{j_1} s_{j_2} \cdots s_{j_i}$$

Fast algebraic attack on E0

- Fast algebraic attack: Decomposition into G_1 and G_2, where,

  $G = G_1 \oplus G_2$; $G_1(L^{t+i}(K)) = \pi^4_{t+1+i} + \pi^2_{t+1+2} \cdot \pi^2_{t+1+i}$; $\deg(G_2) = 3$

- Armknecht's results on Boolean annihilators: the size of E0's characteristic function's "one-set" is too big to allow degree < 3.

- Described attack is of optimal order of complexity.

- Attack's complexity is therefore,

$$Compl. = \left(\binom{128}{3}\right)^3 = 2^{54}$$

• E0 is academicaly broken, but 2^{23} keystream bits are needed. Impossible as at most 2744 bits generated with same IV.

Probabilistic algebraic attacks

- Not well understood, especially solving the system of probabilistic equations.
- Idea is to decrease the degree of system by considering equations satisfied with high probability. Toyocrypt (a rare example) is well approximated by $s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i}$.
- Instead of $f(x)g(x) = 0$ for all x we may find $g'(x)$ such that $deg(g') < deg(g)$ and $f(x)g(x) = 0$ for almost all x
- In other words, find the minimum distance of the $RM^f(d', n)$.

Probabilistic algebraic attacks II

- First problem is how to find minimum distance - NP hard problem !
- How do we solve the system ?
- Basic idea: form sufficiently many systems - one of them correct.
- Treating $RM^f(d', n)$ as a random code determinstic algebraic attacks are better (nonlinear filters)!
- Seems that solving MANY small systems, MANY becomes too much.

Expanding output space

- More output bits, consider $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.
- Already indicated that security decreases with increasing m.
- Instead of annihilators multivariate equations (outputs y_i are known), $F = (y_1(x), \ldots, y_m(x))$,

$$\sum_{a,b} c_{a,b} y^a x^b; \quad x \in \mathbb{F}_2^n,\ y \in \mathbb{F}_2^m, \quad c \in \mathbb{F}_2.$$

- Important to find low degree equation in x_i since outputs y_j are known. Find equations so that $c_{a,b} = 0$ for $wt(b) > d$.

Finding multivariate equations

- Similar to Boolean case, apply Gauss on matrix.

$x \in \mathbb{F}_2^n$	
$RM(d,n)$	
$y_1 RM(d,n)$	Nmb of rows $2^m \sum_{i=0}^{d} \binom{n}{i}$
$\vdots$	
$y_1 \cdots y_m RM(d,n)$	

- The existence condition becomes,

$$2^n < 2^m \sum_{i=0}^{d} \binom{n}{i}$$

Susceptibility to algebraic attacks

- Would be convenient to output byte, $m = 8$. For $n = 24$ always equations of degree $d \leq 6$.
- The use of certain resilient $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ completely compromised.
- Resiliency important for nonlinear combiners - prevent correlation attacks.
- **Do not** increase throughput for nonlinear combiners, based on extended Maiorana-McFarland function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$.

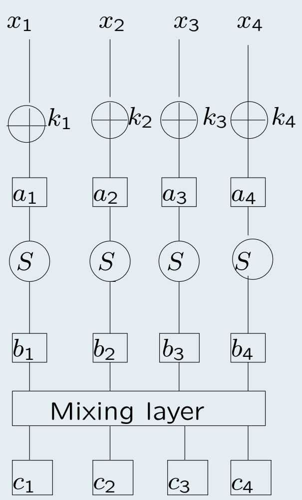
I/O (Input/Output) equations for standard S-boxes

- Function $y = x^{-1}$ allows sparse quadratic equations both over extension and prime field.
- Rewriting $yx+1 = 0$ for $x \neq 0$ is very sparse over $GF(2^n)$. Translated to $GF(2)$ one finds $5n - 1$ linearly independent sparse multivariate quadratic equations.
- In general, there exist multivariate equations of degree at most d whenever,
$$2^n < \sum_{i=0}^{n} \binom{2n}{i},$$
considering terms of degree at most d in $x_1, \ldots, x_n, y_1, \ldots, y_n$.

Algebraic attacks - stream cipher vs block ciphers

- Certain stream ciphers vulnerable to algebraic attacks.
- However no real threats for SNOW 2.0. In SOBER-128 equation of degree 14 relating input and keystream.
- What about block ciphers, especially AES ?
- Need a single (or two) plaintext-ciphertext pair(s).
- Cannot generate enough equations as in case of LFSR stream ciphers.

Algebraic system - simplified example



For $i = 1$ to 4

$a_i = x_i + k_i$

$a_i b_i = 1$

$c_i = \sum_{j \neq i} b_j$

Derived algebraic systems

Cipher	AES-128	BES (over $GF(2^8)$)	HFE
Total variables	1600	3968	n
Total equations	8000	5248	n
Total terms	16096	7808	-

Induced equations for some ciphers

- Sparse systems as number of terms much less than NmbVar^2.
- For HFE system is not random but rather pseudorandom. This is the reason why $F4$ can successfully break 80 bits challenge.

MQ algebraic systems

- System of Multivariate Quadratic equations (MQ) known to be NP-hard for a system of random equations.
- Used in some cryptosystem, best known HFE (Hidden Field Equations), [Pattarin '96], not random.
- Not always exponential time. The algorithm $F5$ breaks HFE 80 bits challenge.
- The idea is to modify Buchberger algorithm for particular instances of the problem.

Representing algebraic systems

- Each equation describing the scheme corresponds to polynomial.
- Then algebraic system is represented by a set of polynomials F in $\mathbb{F}[x_1, \ldots, x_n]$.

$$\begin{array}{rc} a_i = & x_i + k_i \\ a_i b_i = & 1 \\ c_i = & \sum_{j \neq i} b_i \end{array} \longrightarrow F = \left\{ \begin{array}{c} a_i - x_i - k_i = 0 \\ a_i b_i - 1 = 0 \\ c_i - \sum_{j \neq i} b_i = 0 \end{array} \right\},$$

- Solutions to system are common zeros of the ideal spanned by F.

Gröbner basis algorithm

- Grobner basis algorithms: bring the system to desired triangular form,

$$\{a_1 - h_1(x_n), a_2 - h_2(x_{n-1}, x_n), \ldots, a_n - h(x_1, x_2, \ldots, x_n)\}.$$

- Basic Buchberger algorithm for finding Gröbner runs in exponential time (worst case).
- Some modifications of Buchberger such as $F4$ and $F5$ might be faster but still there is no success when applied to AES or BES.

Summary of algebraic attacks

- Efficient atacks for certain schemes, especially stream ciphers based on linear transition (combiners, filters).
- Possibility of breaking block ciphers with Gröbner based algebraic attacks is small.
- The sparseness of the system not sufficient in case of many variables.