Introduction to Design and Analysis of Stream Ciphers

Willi Meier

University of Applied Sciences Northwestern Switzerland
School of Engineering

Albena, June 30 - July 5, 2013

Overview

- Stream Ciphers: A short Introduction
- Cryptanalysis principles
- Time/Memory/Data tradeoffs
- Berlekamp-Massey algorithm
- LFSR-based stream ciphers
- Combiners with Memory
- Correlation attacks
- Linear (distinguishing) attacks
- Algebraic attacks
- The European NoE eSTREAM Project
- NLFSR-based stream ciphers: Trivium and Grain

Introduction

Why stream ciphers?

Applied in:

Environments with high throughput requirements. Stream ciphers can be up to 5 times faster than, e.g., AES.

Devices with restricted resources, e.g., in RFIDs (lightweight crypto).

Introduction

Stream Cipher:
Encrypts sequence of plaintext symbols, e.g., from a binary alphabet $\{0, 1\}$, or from 32-bit words.

Synchronous stream cipher:
The output of a pseudorandom generator, the keystream, is used together with the plaintext to produce the ciphertext.

Additive stream cipher:
Ciphertext symbols c_i obtained from plaintext symbols m_i and keystream symbols b_i by xor addition.

Introduction

A synchronous stream cipher:

Takes as input a κ-bit secret key k and a n-bit public initial vector v (or IV).

Initialization mixes input to generate a random looking initial state.

Thereafter, keystream is output and state is continuously updated.

Introduction

Formally:

Initialization function $F : \{0,1\}^{\kappa} \times \{0,1\}^{n} \mapsto \{0,1\}^{m}$.

State update function $G : \{0,1\}^{m} \mapsto \{0,1\}^{m}$

Output function $H : \{0,1\}^{m} \mapsto \{0,1\}$.

s_t: state at time instant t.

$$s_{t+1} = G(s_t, k),\ z_t = H(s_t, k).$$

Introduction

As in every symmetric crypto system, sender and receiver have to be in possession of the key k (e.g. of 128 bits).

Message split into small packets. Each of them encrypted using a fresh IV as input.

Introduction

Prototype stream cipher: One-Time-Pad (F. Miller 1882, G. Vernam, 1917)

Keystream: A random binary string

OTP has perfect security (Shannon, 1945).

In a deterministic stream cipher, random string of OTP replaced by pseudo random string.

Only secret key k needs to be securely transmitted.

Provable security lost.

Introduction

Examples of stream ciphers

- RC4, used, e.g., in eBanking
- E0, used in the Bluetooth protocol
- A5/1, used in GSM cellphones

A variety of cryptanalytic results known on these ciphers.

Introduction

Stream ciphers can have very simple structure, e.g., RC4 only needs a few lines for its description:

ℓ-byte key k is expanded into N-byte array $K[0...(N-1)]$, $N = 256$:

$K[y] = k[y \mod \ell]$ for any y, $0 \leq y \leq N-1$.

Algorithm 1 KSA

---

for $i = 0$ to $N-1$ in steps of 1 **do**
  $S[i] = i$
end for
$j = 0$
for $i = 0$ to $N-1$ in steps of 1 **do**
  $j = (j + S[i] + K[i])$
  Swap($S[i]$,$S[j]$)
end for

Algorithm 2 PRGA

$i = j = 0$

Key Stream Generation Loop:

$i = i + 1$

$j = j + S[i]$

Swap($S[i]$,$S[j]$)

$t = S[i] + S[j]$

return $z = S[t]$

Introduction

State-of-the-art stream ciphers include:

- SNOW 2.0, software oriented, ISO/IEC standard
- SNOW 3G, 3GPP in UEA2 and UIA2
- ZUC (core of new Long Term Evolution algorithms)
- eSTREAM finalists, e.g., Salsa20, Rabbit for software, and Grain and Trivium for hardware implementation.

Introduction

Stream cipher modes of operation of block ciphers (e.g., Triple DES or AES):

- Cipher feedback
- Output feedback
- Counter mode

Introduction

A dedicated stream cipher with provable security:

QUAD (Berbain-Gilbert-Patarin, 2006)

Based on difficulty of solving systems of multivariate quadratic equations mod 2.

Introduction

Difference between block ciphers and synchronous stream ciphers?

Block cipher needs several rounds until it outputs a block. Resulting output dependent on plaintext.

Dedicated stream cipher produces output after each update (round). Resulting output independent on plaintext (but on present state).

Cryptanalysis principles

In cryptanalysis of stream ciphers: Assume either that

- Some part of plaintext is known (known-plaintext attack), or
- Plaintext has redundancy (e.g., has ASCII format).

For additive stream ciphers, a known part of plaintext is equivalent to a known part of keystream.

Cryptanalysis principles

Distinction between passive and active attacks.

In passive attacks:

Exploit either output mode or initialization (resynchronization) mode.

Key recovery: Attempt to recover secret key k out of observed key stream.
Distinguishing attack: Try to distinguish observed key stream from being a purely random sequence. Distinguishing attacks may sometimes be turned into key recovery attacks.

Side-channel attack: Measures radiation or power consumption during execution of encryption.

Cryptanalysis principles

In active attacks:

- Adversary inserts, deletes or replays ciphertext digits. Causes loss of synchronization: Data intgrity check and data origin authentication necessary.
- Fault attack: Adversary actively induces faults in state (e.g., by ionizing radiation).

Berlekamp-Massey algorithm

Efficient method to deliver shortest LFSR, together with initial state that can generate a given sequence.

LFSR of length L:

State vector $(x_L, ..., x_1)$. In one step, each bit is shifted one position to the right, except the rightmost bit x_1 which is output.

On the left, a new bit is shifted in, by a linear recursion

$$x_j = (c_1 x_{j-1} + c_2 x_{j-2} + ... + c_L x_{j-L}) \bmod 2,$$

for $j > L$.

Berlekamp-Massey algorithm

Linear complexity of a binary sequence:

Length of shortest LFSR that can produce the given sequence.

Complexity of Berlekamp-Massey algorithm: Quadratic in length of LFSR.

Consequence: Linear complexity and period of stream cipher need to be large.

Time/Memory/Data tradeoffs

General type of attack. Introduced for block ciphers by Hellman (1980).

For stream ciphers introduced by Babbage (1995), Golić (1997). General treatment by Biryukov-Shamir (2000).

N: size of search space
M: amount of random access memory
T: time required by realtime phase of attack
D: amount of realtime data available to attacker

Time/Memory/Data tradeoffs

Statement of basic version of attack: $TM = N$.

Example: $T = M$; Hence $T = M = N^{1/2}$.

Attack associates to each of N possible states of generator a string of the first $\log(N)$ bits of output produced from that state.

Time/Memory/Data tradeoffs

Mapping $f(x) = y$ from states x to output prefixes y:

Easy to evaluate but hard to invert.

Preprocessing phase: Pick M random states x_i, compute y_i, and store all (x_i, y_i) in a sorted table.

Realtime phase: Given $D + \log(N) - 1$ output bits, derive all possible D windows $y_1, ..., y_D$ of $\log(N)$ consecutive bits (with overlaps). Look up each y_i in table. If one y_i is found, can determine corresponding x_i.

Time/Memory/Data tradeoffs

Threshold of success: Birthday paradox.

Two random subsets of space with N points each are likely to intersect when product of their sizes exceeds N.

Hence $DM = N$, where preprocessing time $P = M$, attack time $T = D$, i.e., $TM = N$.

Consequence: Size N of state space of stream cipher should be at least twice the size of secret key.

LFSR-based stream ciphers

LFSRs easy to implement in hardware.

Depending on linear recursion, LFSRs have desirable properties:

- Output sequence has large period (e.g. maximum period $2^L - 1$).
- Good statistical properties.
- Easy to analyse algebraically.

LFSR-based stream ciphers

Drawback for cryptography: LFSRs easy to predict.

Solve a system of linear equations for unkonwn state bits and recursion coefficients, or use Berlekamp-Massey algorithm.

Destroy linearity by

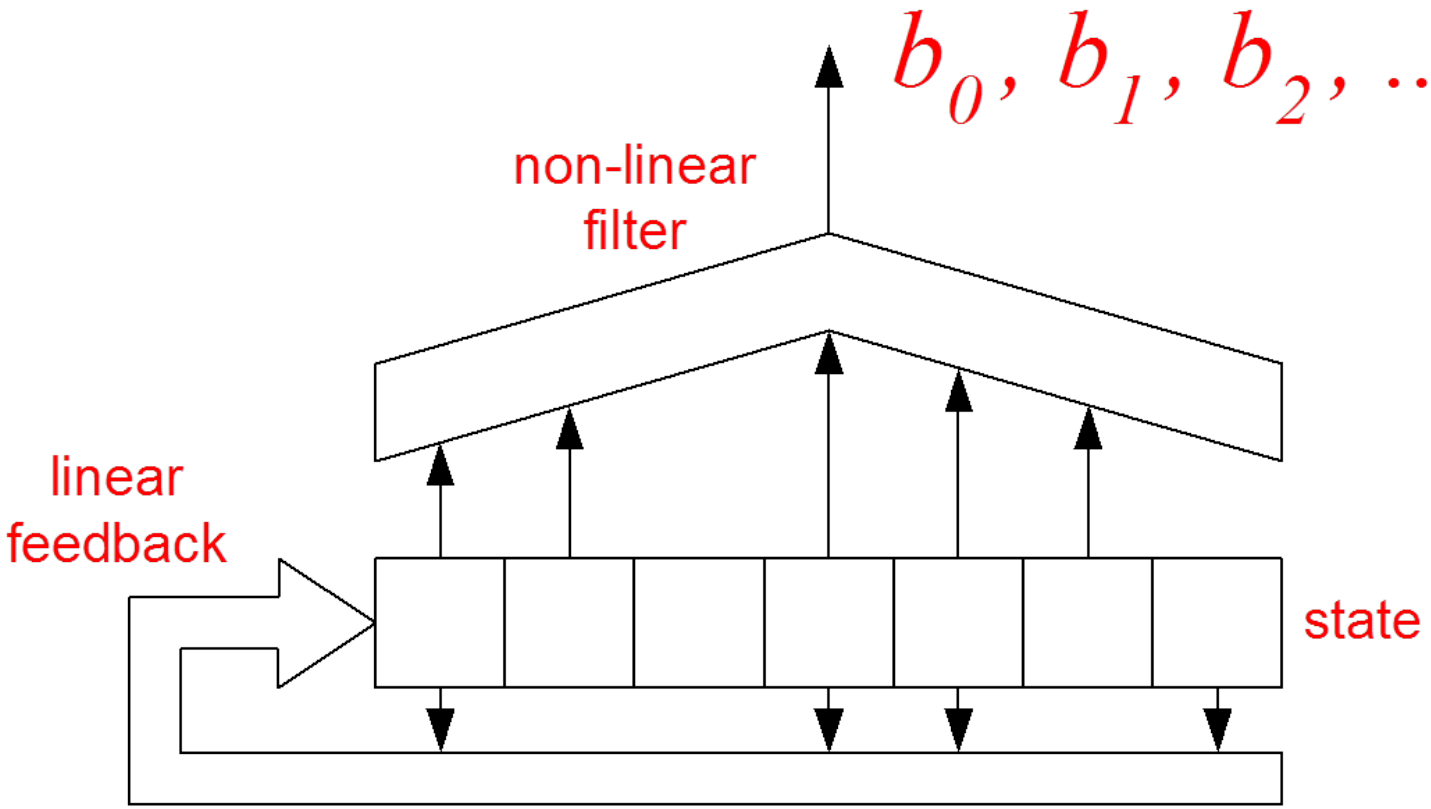
- Nonlinear filter/combining functions on outputs of one or several LFSRs.
- Use of output of one/several LFSRs to control the clock of one/more other LFSRs.

LFSR-based stream ciphers can have some provable properties, like large period or linear complexity.

LFSR-based stream ciphers

Nonlinear filter generator:

Generate key stream bits $b_0, b_1, b_2, \ldots$, as some nonlinear function f of the stages of a single LFSR.



LFSR-based stream ciphers

Many (classical) stream ciphers are LFSR-driven, e.g.,

- $A5/1$
- Shrinking and self-shrinking generator

Combiners with Memory

A (k, m)-combiner with k inputs and m memory bits is a finite state machine (FSM), defined by an output function

$$f : \{0,1\}^m \times \{0,1\}^k \to \{0,1\}$$

and a memory update function

$$\varphi : \{0,1\}^m \times \{0,1\}^k \to \{0,1\}^m.$$

For given stream of inputs $(X_1, X_2, ...)$, $X_i \in \{0,1\}^k$, and initial assignment Q_1 in $\{0,1\}^m$ to memory bits, the output bit stream is defined as:

$$z_t = f(Q_t, X_t),$$

and

$$Q_{t+1} = \varphi(Q_t, X_t),$$

all $t > 0$.

Often, driving devices for generating input streams are LFSRs. Initial states determined by the secret key.

Example: Summation generator

Let $k = 2$ inputs. Write X_t as $X_t = (a_t, b_t)$. Number of memory bits: $m = 1$, given by carry of integer addition in binary representation.

Functions f and φ defined as

$$z_t = f(Q_t, a_t, b_t) = a_t \oplus b_t \oplus Q_t$$

and

$$Q_{t+1} = \varphi(Q_t, a_t, b_t) = a_t b_t \oplus a_t Q_t \oplus b_t Q_t.$$

Important stream ciphers using a combiner with memory: E_0, SNOW 2.0, SOSEMANUK.

Combiners with Memory

SNOW 2.0 (Ekdahl-Johansson, 2002)

Key size 128 bits.

Overall structure: Word-oriented filter generator. At each cycle a 32-bit word is output.

A length 16 LFSR over the finite field $GF(2^{32})$ feeds a finite state machine.

FSM represents nonlinear part and consists of two 32-bit registers. $m = 64$ bit memory.

Nonlinearity achieved through integer addition as well as 32-bit permutation using S-box and MixColumn of AES.

Correlation Attacks

Correlation attack illustrated by Combination Generator

The outputs a_m of s LFSRs are used as input of a Boolean function f to produce key stream,

$$f(a_{1m}, ..., a_{sm}) = b_m.$$

Correlation: Prob$(b_m = a_{im}) = p, \;\; p \neq 0.5$.

Example: $s = 3$.

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$$

$p = 0.75$.

Correlation Attacks

Statistical model: Assume a binary asymmetric source z_m with $\text{Prob}(z_m = 0) = p > 0.5$. Let

$$b_m = a_m + z_m \bmod 2.$$

Decoding problem: Given N digits of $\underline{b}$ (and the structure of the LFSR, of length L).

Find correct output sequence $\underline{a}$ of the LFSR.

Correlation Attacks

Known solution: By exhaustive search over all initial states of LFSR find $\underline{a}$ such that

$$T = \#\{j | b_j = a_j, 0 \leq j \leq N\}$$

is maximum. Complexity $O(2^L)$.

Feasible for L up to about 50.

Search can be accelerated by Fast Correlation Attacks.

Correlation Attacks

Fast correlation attack: Significantly faster than exhaustive search over all initial states of target LFSR. Based on using

parity check equations created from feedback polynomial of LFSR (R. Gallager, Low-density parity-check codes 1963, MS 1988, CJM 2003,...).

Correlation Attacks

Correlation attacks can be successful if cipher allows for good approximations of the output function by linear functions in state bits of LFSRs involved (Linear attack).

In design of stream ciphers, Boolean functions f should

- be correlation immune
- have large Hamming distance to affine functions
- have large algebraic degree (to counter Berlekamp-Massey synthesis)

Correlation attacks

Correlation immunity:

Let $X_1, X_2, ..., X_n$ be independent binary variables, which are balanced (i.e. each takes values 0 and 1 with probability 1/2.

A Boolean function $f(x_1, x_2, ..., x_n)$ is m-th order correlation immune if for each subset of m random variables $X_{i1},, X_{i2}, ..., X_{im}$ the random variable $Z = f(X_1, X_2, ..., x_n)$ is statistically independent of the random vector $(X_{i1}, X_{i2}, ..., X_{im})$.

Tradeoff between order m of correlation immunity and degree of Boolean function: For balanced f, degree of f is at most $n - m - 1$ for $1 \leq m \leq n - 2$. Tradeoff can be avoided by using memory.

Example: The function f in the summation generator with $k = 2$ inputs is second order correlation immune,

$$f(Q_t, a_t, b_t) = a_t \oplus b_t \oplus Q_t.$$

Linear Attacks

Linear attacks seek for correlations between

1. linear functions of selected keystream bits, or
2. between linear functions of selected keystream bits and linear functions in state bits.

Correlations can be exploited either for a distinguisher or even for key recovery in second case, if there are many more linear relations than unknowns.

Linear Attacks

Correlations in combiner with M-bit memory:

Consider block of m consecutive outputs $Z_t = (z_t, z_{t-1}, ..., z_{t-m+1})$ as a function of corresponding block of input vectors $X_t = (X_t, X_{t-1}, ..., X_{t-m+1})$ at time t and the preceeding M-bit memory vector C_{t-m+1} at time $t - m + 1$. Assume X_t and C_{t-m+1} balanced and mutually independent.

Then, if $m \geq M$, there must exist linear correlations between the output and the input bits (Golić), but they may also exist if $m < M$.

Linear attacks have been devised against various stream ciphers, including SNOW 1.0 (Coppersmith-Halevi-Jutla, 2002) and SNOW 2.0 (Watanabe-Biryukov-De Cannière, Nyberg-Wallén,...).

Algebraic attacks

Algebraic attacks: Solve systems of algebraic equations (CM, 2003).

Type of equations: System of multivariate polynomial equations over finite field, e.g. $GF(2)$.

$$\begin{array}{ccccccc} x_1 & + & x_0x_1 & + & x_0x_2 & + \cdots & = 1 \\ x_1x_2 & + & x_0x_3 & + & x_7 & + \cdots & = 0 \\ \ldots & + & \ldots & + & \ldots & + \cdots & = \ldots \end{array}$$

Breaking a good cipher should require:

" ... as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type " [Shannon, 1949, Communication theory of secrecy systems].

Common experience: Large systems of equations become intractable soon with increasing number of unknowns (is NP-hard problem).

Algebraic Attacks

However:

Systems that are

- Overdefined, i.e., have more equations than unknowns, or
- Sparse

are easier to solve than random systems, e.g., by

- Linearization
- Gröbner bases
- SAT-solvers

Algebraic Attacks

Direct algebraic approach:

Derive equations in key/state bits

$$\begin{aligned} f(k_0, k_1, ..., k_{n-1}) &= b_0 \\ f(L(k_0, k_1, ..., k_{n-1})) &= b_1 \\ f(L^2(k_0, k_1, ..., k_{n-1})) &= b_2 \\ ..... &= ... \end{aligned}$$

$L()$: Linear recursion.

Algebraic Attacks

Solve this system of equations.

In context of stream cipher analysis: System overdefined depending on amount of known key stream.

Linearization:

Assumption: f is of low algebraic degree d. Then the key is found given about $D = \sum_{i=1}^{d} \binom{n}{d}$ key stream bits and within D^{ω} computations, where ω is the exponent of Gaussian reduction ($\omega < 3$).

Linearization: One new variable for each monomial. Solve linear system.

Algebraic Attacks

Scenarios for high-degree f:

Suppose $f = g \cdot h$. Assume furthermore

- $f \cdot g = 0$, where the degree of g is low, or
- $f \cdot g = h$, where both, degrees of g and h are low.

If output bit $b_i = 1$, the first case gives $g(s) = 0$ for state s.

If output bit $b_i = 0$, get equation $h(s) = 0$.

Algebraic attacks

Idea of algebraic attack:

Instead of $f(s) = b_t$ with $s = L^t(k)$ and secret key k,

solve the equations

$$f(s) \cdot g(s) = b_t \cdot g(s)$$

with well-chosen function g.

Question: Do favorable functions g of low degree exist?

Algebraic Attacks

Under some condition, such functions g do always exist.

Theorem (Low-degree relations)
Let f be any Boolean function in k variables. Then there is a nonzero Boolean function g of degree at most $k/2$ such that $f(x) \cdot g(x)$ is of degree at most $k/2$.

(Take ceilings of $k/2$ if k is odd.)

This result has been motivated by cryptanalysis of multivariate digital signature schemes as well as by cryptanalysis of AES block cipher.

Algebraic Attacks

Consequence:

Algebraic attack breaks any stream cipher with linear feedback and Boolean output function with a small number k of state bits as input, in polynomial complexity, if k is considered as a small constant.

Complexity only approx. square root of known attack.

Algebraic Attacks

Attack works for more general LFSR-based stream ciphers, e.g., for combiners with memory.

Fast algebraic attack (Courtois 2003).

No multivariate equations of low degree should exist that relate state bits and one or more output bits.

Algebraic attack on filter generator by Helleseth-Rønjom (2007):

Needs $O(D)$ keystream bits with complexity $O(D)$, after precomputation with complexity $O(D(\log_2 D)^3)$.

Does not take advantage of low-degree polynomial multiples of filter function.

The eSTREAM Project

eSTREAM: Project to identify ”new stream ciphers that might become suitable for widespread adoption”.

Organized by the EU NoE network ECRYPT.

Set up as a result of failure of predecessor project NESSIE.

Started in November 2004 and ended in May 2008.

Project goal: Find algorithms suitable for different profiles.

No standardization (as opposed to AES or SHA-3 competitions).

The eSTREAM Project

Profile 1: Stream ciphers for software applications where high throughput is required (with higher performance than AES in counter mode).

Profile 2: Stream ciphers for hardware applications with restricted resources, e.g., limited storage, gate count, or power consumption.

Both profiles contain a subcategory with ciphers that also provide authentication in addition to encryption.

In reaction to Call for Primitives: 34 proposals were submitted!

The eSTREAM Project

Four finalists in each category:

Profile 1 (Software):
HC-128
Rabbit
Salsa20/12
SOSEMANUK

Profile 2: (Hardware):
Grain v1
MICKEY 2.0
Trivium
(F-FCSR)

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

NLFSR-based stream ciphers: Trivium and Grain

Nonlinear feedback shift register (NLFSRs): Building blocks of several lightweight primitives.

Facilitate efficient hardware.

Classical LFSR-based stream ciphers: Update function is implemented by one or several LFSRs.

Burden to create nonlinearity of construction carried entirely by output function.

NLFSR-based stream ciphers: Trivium and Grain

NLFSR-based constructions: Nonlinearity may be shared between update and output function.

Can prevent algebraic attacks.

NLFSRs much less understood than LFSRs (e.g., period?)

Only few tools available to assess security of NLFSR-based cryptosystems.

NLFSR-based stream ciphers: Trivium and Grain

Trivium is eSTREAM finalist, designed by De Cannière and Preneel in 2005.

- 80-bit secret key and 80-bit initial value IV (public)
- 3 quadratic NLFSRs, of different lenghts
- 1152 initialization rounds before output is produced
- Increased efficiency by factor up to 64: Implement Boolean functions in parallel

NLFSR-based stream ciphers: Trivium and Grain

State size is 288 bit.

Update function nonlinear, to counter algebraic attacks.

Output function is linear.

At each update, one output bit is produced.

NLFSR-based stream ciphers: Trivium and Grain

Initialization of Trivium

$(s_1, s_2, ..., s_{93}) \leftarrow (k_0, ..., k_{79}, 0, 0, ..,)$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (x_0, x_1, ..., x_{79}, 0., , , , 0)$
$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (0, 0, ..., 0, 1, 1, 1)$
for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, ..., s_{92})$

$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, ..., s_{176})$

$(s_{178}, ..., s_{288}) \leftarrow (t_2, s_{178}, ..., s_{287})$

end for

NLFSR-based stream ciphers: Trivium and Grain

Output generation of Trivium

for $i = 1$ to ℓ **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

NLFSR-based stream ciphers: Trivium and Grain

Remarks

If in iterations, state variables $s_1, ..., s_{288}$ are expressed by $k_1, ..., k_{80}$ and $v_1, ..., v_{80}$, degree of polynomials increases only slowly.

System of equations in state variables for given output sequence $z_1, ..., z_\ell$ is of low degree for $\ell = 288$, and has only few nonlinear monomials.

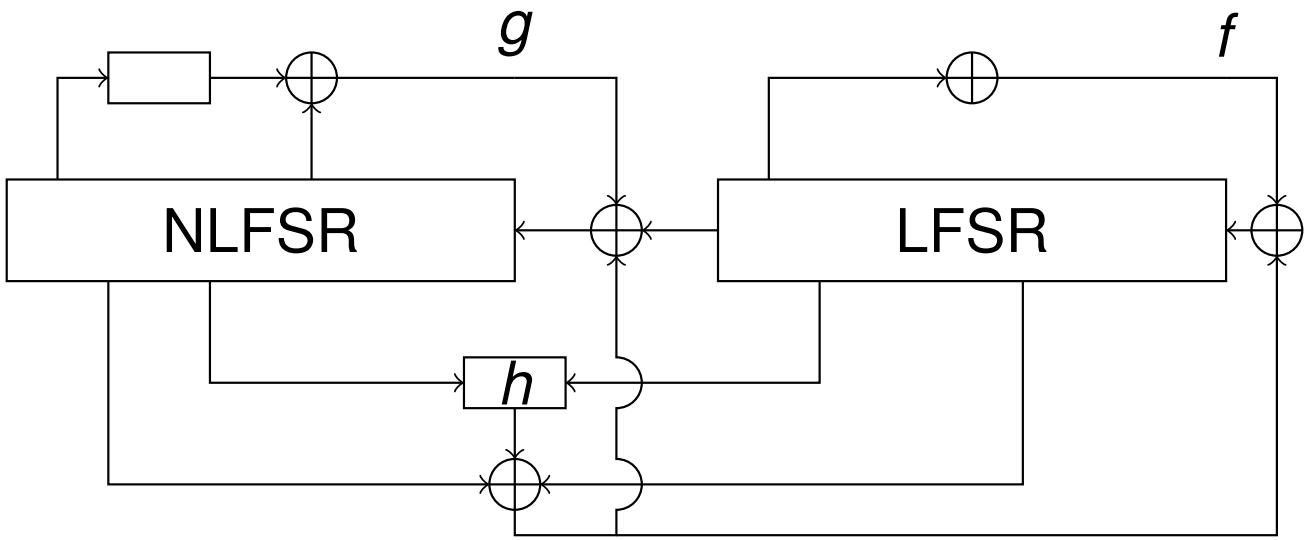
Best attack on full Trivium for given output sequence by Maximov-Biryukov.

Involves guessing of certain state bits and products of state bits that reduce nonlinear system of equations to linear one.

Complexity: $c \cdot 2^{84}$ for some constant c.

NLFSR-based stream ciphers: Trivium and Grain

Initialization of Grain-128a (follow up of Grain-128)



f: Primitive feedback polynomial of the LFSR.

g: Nonlinear feedback polynomial of the NLFSR of order 4.

$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$.

NLFSR-based stream ciphers: Trivium and Grain

State size: 256 bit.

Key size: 128 bit. Loaded in NLFSR.

IV size: 96 bit. Loaded in LFSR.

Remaining 32 bits fixed to 1, except last bit, which is set to 0.

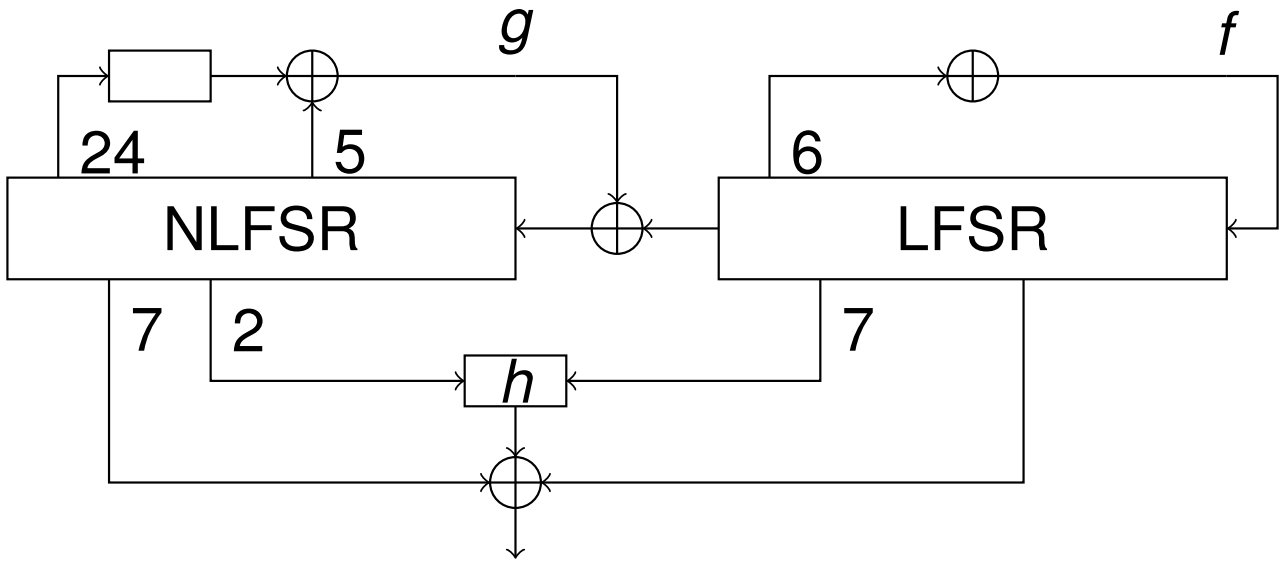
Grain-128a allows for optional authentication.

Grain-128a is update of Grain-128, which has been cryptanalyzed with complexity lower than 2^{128} operations.

Authentication based on additional LFSR using method by H. Krawczyk.

NLFSR-based stream ciphers: Trivium and Grain

Output mode of Grain-128a



In mode without authentication, all output bits used directly as keystream.

Increase of efficiency by factor up to 32 using parallel implementation of Boolean functions.

Concluding remarks

- Ratio between known and publicly known design and analysis?
- Initialization mechanism ad hoc: Better designs?
- Stream ciphers with provable properties (correlations, linear approximations)
- CAESAR competition for Authenticated Encryption