

University of Salzburg, February 2, 2007

Design and Recent Analysis of eSTREAM Candidates

Willi Meier



University of Applied Sciences Northwestern Switzerland
School of Engineering

Overview

- Stream Ciphers
- The eSTREAM Project
- Construction Principles
- Some eSTREAM Candidates
- Cryptanalysis Methods
- Analysis of Salsa
- Analysis of Trivium
- Conclusions

Stream Ciphers

Stream cipher:

Encrypts sequence of plaintext characters, e.g., from binary alphabet $\{0,1\}$.

Synchronous stream cipher:

The output of a pseudorandom generator, the *key-stream*, is used together with plaintext to produce ciphertext.

Additive stream cipher:

Ciphertext symbols c_i are obtained from plaintext symbols m_i and keystream symbols b_i by addition.

Addition is often bitwise XOR (i.e., addition mod 2):

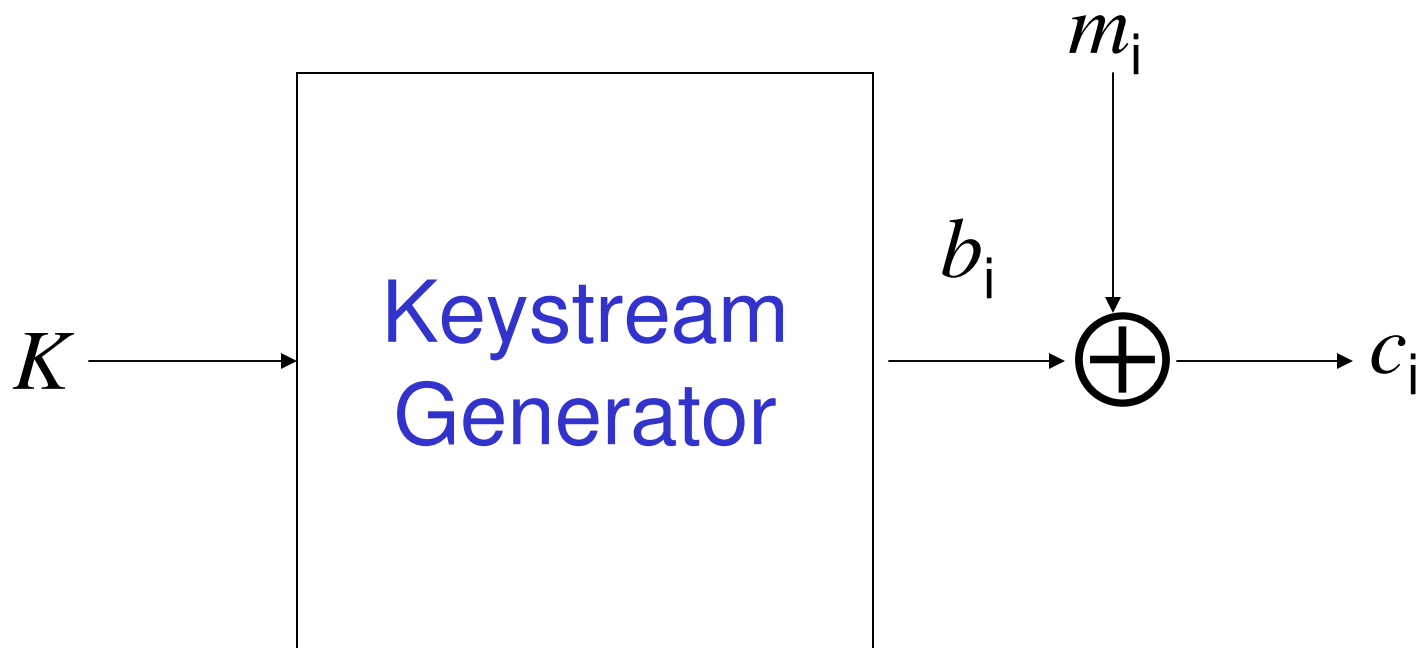
$$c_i = m_i + b_i \pmod{2}.$$

Before transmission, secret key K has to be transmitted in *secure way* to receiver.

Decryption:

Achieved simply by subtracting keystream symbols from ciphertext symbols: $m_i = c_i + b_i \pmod{2}$

Model of a binary additive stream cipher:



Some popular stream ciphers:

- **RC4**, used in Netscape's Secure Socket Layer (SSL) protocol.
- **A5**, in the Global System for Mobile Communication (GSM).
- **Bluetooth** stream cipher, standard for wireless short-range connectivity, specified by the Bluetooth Special Interest Group.

Stream ciphers

- Are **generally faster** than other symmetric encryption systems like block ciphers, and are much faster than any public key cryptosystem.
- Are more appropriate, when **buffering is limited**, or when characters must be individually processed.
- Have **no error propagation**.

Prototype stream cipher: **One-time-pad**

Keystream is randomly chosen binary string of same length as plaintext, and is never used again.

One-time-pad is „unconditionally secure“.

If keystream is **reused**, the one-time-pad (like every stream cipher) is **insecure**.

Drawback of one-time-pad: **Key as long as plaintext**; makes distribution of secret key difficult in practice.

In practical applications:

Random keystream is replaced by output of an efficient deterministic pseudorandom generator.

Initial state is **short random string** K of binary digits (e.g. of 128 bits).

Only secret key K needs to be securely transmitted.

Thereby **provable security is lost**.

For cryptographic applications, generated key-stream should pass a whole battery of statistical tests

Computational effort to predict the keystream for unknown initial state:

Should be far **beyond the capabilities** of an adversary.

Many pseudorandom generators used, e.g., for **computer simulations**, wouldn't satisfy this requirement.

Several stream ciphers proposed have been broken.

The eSTREAM Project

eSTREAM is a project to identify "new stream ciphers that might become suitable for widespread adoption" .

Organised by the EU ECRYPT network.

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

Call for primitives issued in November 2004.

Project due to complete in January 2008.

Divided into separate phases.

Project goal: Find algorithms suitable for different application profiles.

Profiles of submissions to eSTREAM:

Profile 1: Stream ciphers for software applications where high throughput is required (with higher performance than AES block cipher 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!

Phase 1:

General analysis of all submissions with the purpose of selecting a subset of the submitted designs for further investigation.

Designs were scrutinized based on criteria of security, performance (with respect to AES, and the other candidates), simplicity and flexibility, supporting analysis, clarity and completeness of the documentation.

Phase 1 activities included much analysis and presentations of analysis results as well as discussion.

Project developed a framework for testing performance of candidates. Used to benchmark candidates on a wide variety of systems.

SASC'2006 – Stream Ciphers Revisited (February 2006, Leuven)

End of Phase 1 in March 2006.

Phase 2

Started on 1st August 2006.

For each of the profiles a number of algorithms has been selected to be Focus Phase 2 algorithms.

Designs that eSTREAM finds of particular interest.

More cryptanalysis and performance evaluation on these algorithms encouraged.

In addition, a number of algorithms for each profile are accepted as Phase 2 algorithms (still valid as eSTREAM candidates).

Focus 2 candidates to be re-classified every six months.

Selected as focus candidates:

Profile 1: 7 candidates

Profile 2: 4 candidates

Remaining part of Timetable:

SASC'2007 (Bochum, 31st Jan. – 1st Feb. 2007)

Sept. 2007: End of evaluation phase of eSTREAM

Jan. 2008: Final report of eSTREAM

Construction principles

General form of (dedicated) stream cipher:

State with update function, and

Output function defined on state.

State is updated linearly or nonlinearly, in key-dependent or key-independent way.

Output function nonlinear (may also be key-dependent)
After each update, word or single bit is output.

State with update function is often a linear feedback shift register (LFSR).

Easily implementable in hardware.

A LFSR of length L :

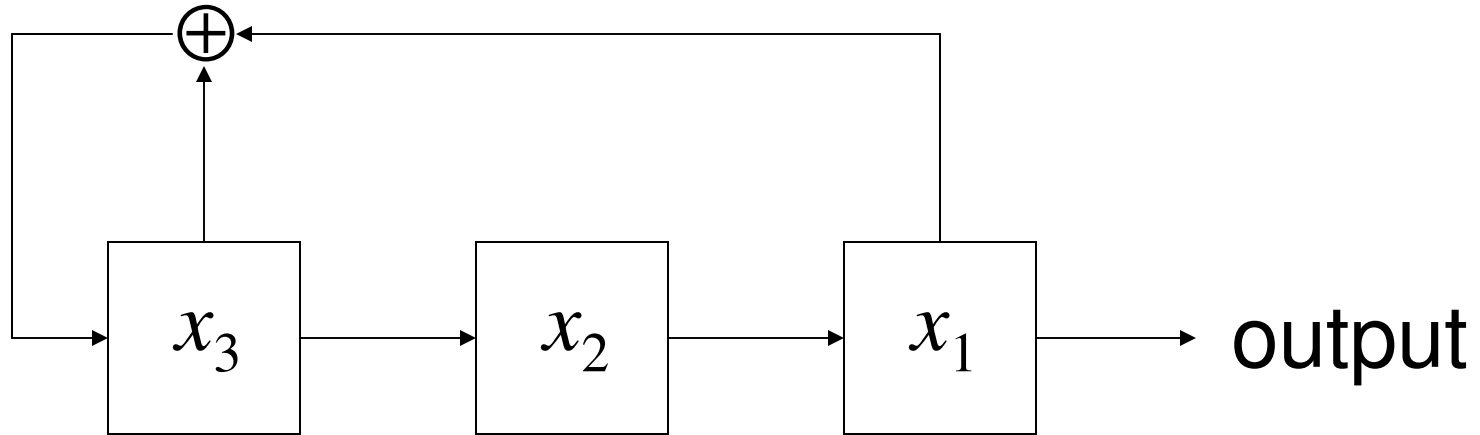
Consists of a bit vector (x_L, \dots, x_1) . In one step, each bit is shifted one position to the right, except the right-most bit x_1 which is output.

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

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

for $j \geq L$

Example: $L = 3$



$$x_j = x_{j-1} + x_{j-3} \pmod{2}$$

Depending on the chosen linear recursion, LFSR's have desirable properties:

- Produce output sequences of large period (e.g. maximum period $2^L - 1$)
- Produce sequences with good statistical properties
- Can be readily analyzed using algebraic techniques

Serious drawback of LFSR's for cryptography:

Output is **easily predictable**, even for **unknown initial state** of bit vector (x_L, \dots, x_1) , and **unknown recursion**:

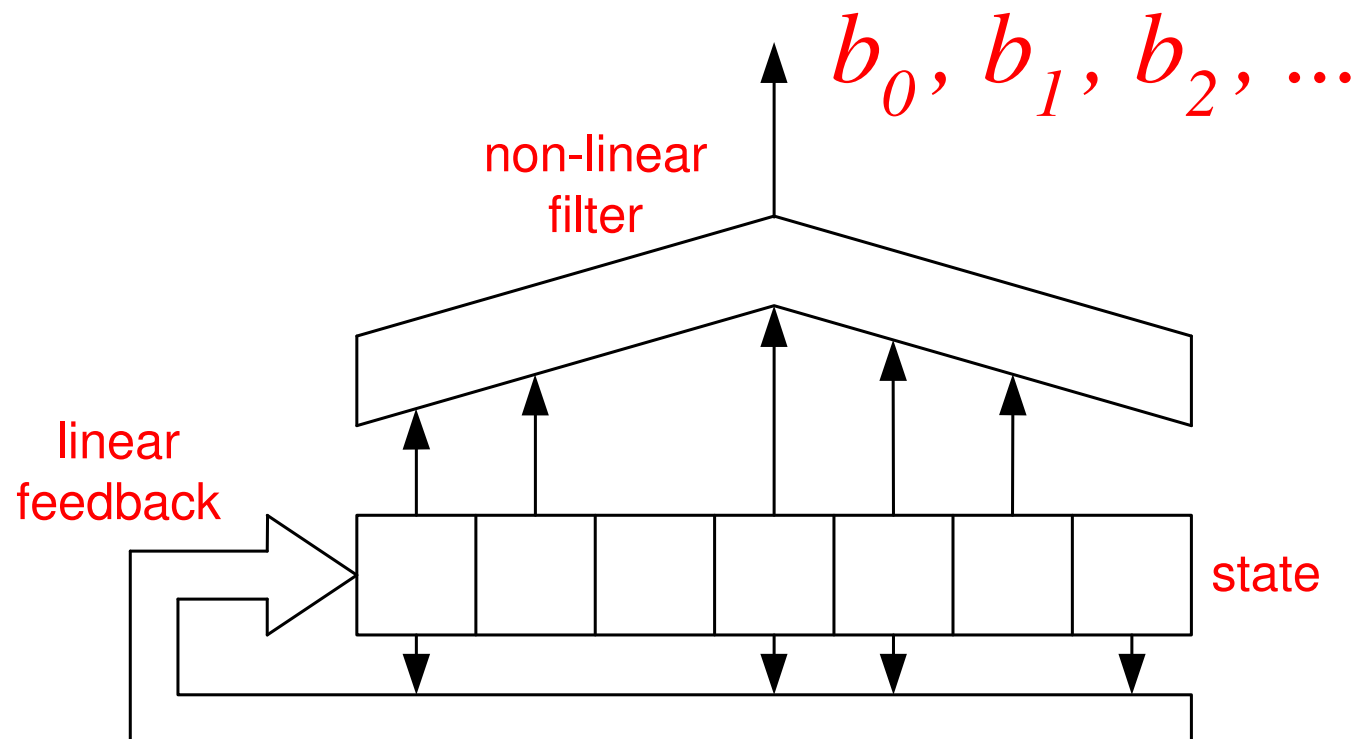
Solve a **system of linear equations** in unknown state bits.

Common **methods for destroying linearity** properties of LFSR's:

- Use **nonlinear filter/combining function** on outputs of one/several LFSR's
- Use output of one/more LFSR's to **control** the **clock** of one/more other LFSR's.

Classical construction: Nonlinear filter generator.

Generate keystream bits b_0, b_1, b_2, \dots , as some nonlinear function f of the stages of a single LFSR.



Extensions/Developments:

For software applications: Word oriented instead of bit-oriented registers, where a word is a 32 bit entity.

Due to algebraic attacks on LFSR-based stream ciphers:

Several eSTREAM candidates use **nonlinear** update functions, but otherwise similar to filter generator.

Some eSTREAM candidates use entirely different construction principles:

Based on design of block ciphers.

Some eSTREAM Candidates

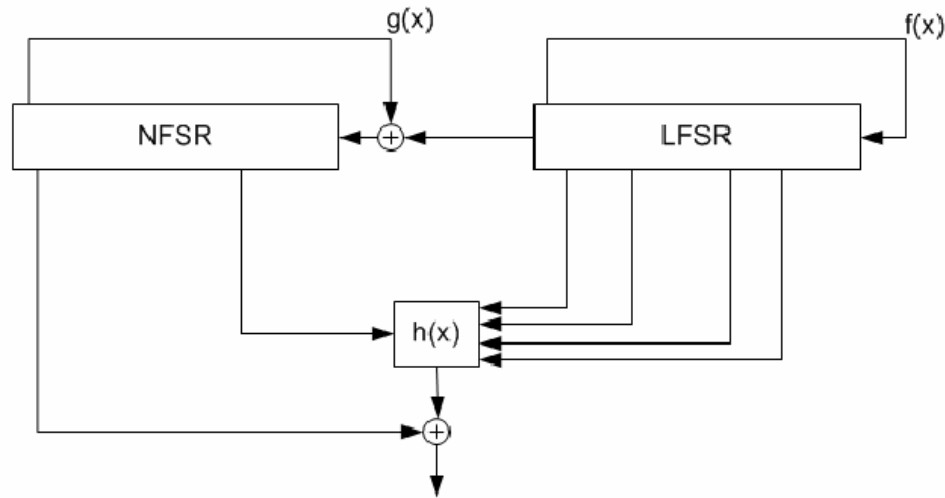
Choose 3 focus candidates (out of 11):

Profile 1: Salsa (D. Bernstein)

Profile 2: Grain (M. Hell, Th. Johansson)

Trivium (Ch. De Cannière, B. Preneel)

Grain



3 main parts:

80 bit LFSR, 80 bit NFSR, nonlinear filter h .

Input to NFSR masked with a LFSR bit.

Output bit masked with xor of 7 NFSR bits.

Grain-128

Variant with key length 128 bit and two 128 bit registers.

Possibility for increased speed at cost of extra hardware.

Throughput in simplest version: 200 Mbit/s.

Trivium

State: 288 bits

nonlinear update

linear output function

80-bit key

State consists of 3 registers,

$$R_1 = (x_1, \dots, x_{93}), \quad R_2 = (x_{94}, \dots, x_{177}), \quad R_3 = (x_{178}, \dots, x_{288}).$$

Construction influenced by design of block ciphers.

Update and output in Trivium

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

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

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

$$z_i \leftarrow t_1 + t_2 + t_3$$

$$t_1 \leftarrow t_1 + x_{91}x_{92} + x_{171}$$

$$t_2 \leftarrow t_2 + x_{175}x_{176} + x_{264}$$

$$t_3 \leftarrow t_3 + x_{286}x_{287} + x_{69}$$

$$(x_1, \dots, x_{93}) \leftarrow (t_3, x_1, \dots, x_{92})$$

$$(x_{94}, \dots, x_{177}) \leftarrow (t_1, x_{94}, \dots, x_{176})$$

$$(x_{178}, \dots, x_{288}) \leftarrow (t_2, x_{178}, \dots, x_{287})$$

Salsa

State: matrix of 16 words of 32 bits

Update: Increment of a counter

Output function: Kind of hash function, achieved through iteration of simple operation, called **quarterround**:

Input $y = (y_0, y_1, y_2, y_3)$, Output $z = (z_0, z_1, z_2, z_3)$

$$z_1 = y_1 \oplus ((y_0 + y_3) \lll 7)$$

$$z_2 = y_2 \oplus ((z_1 + y_0) \lll 9)$$

$$z_3 = y_3 \oplus ((z_2 + z_1) \lll 13)$$

$$z_0 = y_0 \oplus ((z_3 + z_2) \lll 18)$$

State as a matrix:

$$\begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{pmatrix}$$

Update below diagonal words first.

Repeat for all word in columns, then in rows.

10 rounds columns, 10 rounds rows.

Output the keystream $X^0 + X^{20}$.

Cryptanalysis Methods

In **cryptanalysis of stream ciphers**: Common to assume either that

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

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

- **Key recovery attack**: Attempt to recover secret key K out of observed keystream.
- **Distinguishing attack**: Try to distinguish observed keystream from being a purely random sequence.

Distinguishing attacks often weaker than key recovery attacks.

May still be threat, if they allow to deduce information on unknown plaintext out of known part of plaintext, e.g. if period of keystream sequence is small.

Consequence: **Period needs to be large.**

Distinguishing attacks

Goal of key-recovery attacks: Find secret key faster than by exhaustive search.

Distinguishing attacks:

Allow for **distinguishing** observed keystream **from random**, or

make **prediction** about **future** portions of keystream out of known keystream segment.

General statistical framework:

Hypothesis testing

Need to distinguish probability distribution generated by output of a stream cipher from truly random distribution.

Specific distinguishing attacks: [Linear attacks](#) (Golić, Coppersmith-Halevi-Jutla).

Linear attack: Concentrate on non-linear output function to look for characteristic that can be distinguished from random, e.g. linear approximation that has noticeable bias.

Linear attacks have been applied e.g. to RC4.

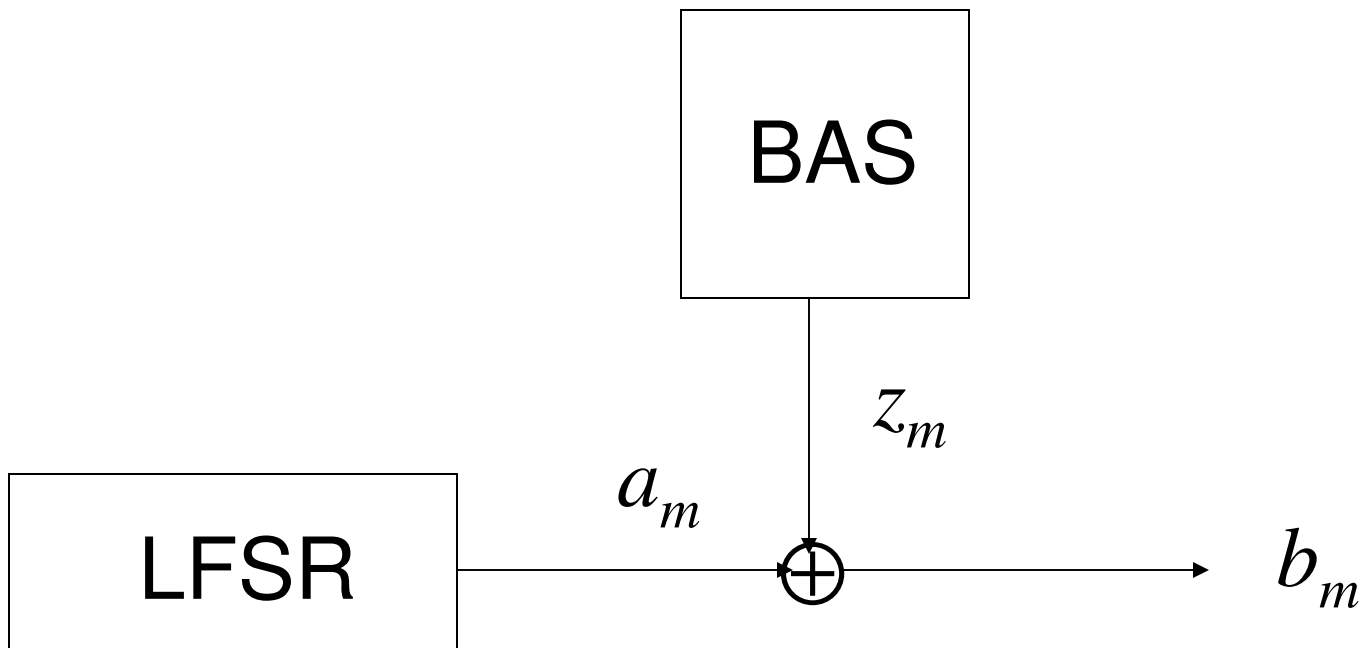
Debate about applicability of distinguishing attacks.

It has been argued that some distinguishing attacks against stream ciphers are unrelated to their security in practical use:

Amount of data required to perform distinguishing attack is huge compared to actual lifetime of secret key used.

Correlation Attack

Statistical Model:



BAS: Binary asymmetric source,
 $Prob(z_m = 0) = p > 0.5$

Problem: Given N digits of \underline{b} (and the structure of the LFSR, of length n).

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

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

$$T = \# \{ j \mid b_j = a_j, 1 \leq j \leq N \}$$

is maximum. Complexity: $O(2^n)$

Feasible for n up to about 50.

Advanced method: Fast correlation attack.

Solve this system of equations. **Very overdefined**, even for moderate quantity of keystream, e.g., 20 Kbytes.

An obvious linearization attack:

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

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

Scenarios

Degree of output function f large, $f=g*h$

- $f*g=0$, degree of g low
- $f*g=h$, degrees of g and h low

If output bit $b_i=1$, take $g(s)=0$, else take equation $h(s)=0$

Instead of $f(s) = b_t$ with $s = L^t(K)$, $K = \text{key}$:

Solve the equations

$$f(s) * g(s) = b_t * g(s)$$

with well chosen function g .

In some cases, such $g(s)$ 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 $f(x) * g(x)$ is of degree at most $k/2$.

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

Time/Memory/Data tradeoffs for Stream Ciphers

General type of attack.

For block ciphers introduced 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

Statement of basic version of attack: $T M = N$

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

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

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

Easy to evaluate but hard to invert.

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

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

Threshold of success: Birthday paradox

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

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

Consequence: Size N of state space should preferably be twice the size of secret key.

Other methods

In stream cipher cryptanalysis, attack methods often dedicated and one of a kind.

Besides correlation attacks, algebraic attacks and time memory tradeoffs only few general methods.

Method known from analysis of block ciphers:
Differential cryptanalysis.

Analysis of Salsa

Indocrypt'2006 (Fischer, Berbain, Biasse, Robshaw)

Initialization of Salsa:

Fill state with $(key, counter, nonce)$, $counter = 0$

Initial vector (known): $IV = (counter, nonce)$

$$\begin{pmatrix} const & key & key & key \\ key & const & nonce & nonce \\ counter & counter & const & key \\ key & key & key & const \end{pmatrix}$$

Differential attack on r round function F (previous work by Crowley):

$$(K, IV) \rightarrow F(K, IV)$$

$$(K', IV') \rightarrow F(K', IV')$$

$$\Delta \rightarrow \Delta^r$$

Framework:

Many inputs (K, IV) with fixed Δ

Bias in Δ^r

Measure bias with χ^2 test

Find optimal input difference Δ , and optimal inputs (K, IV)

Determine optimal mask in Δ^r

Step 1: Optimal input difference Δ

Consider linear version of Salsa, depends only on Δ .
 Good approximation if active words have low weight.

$$\Delta_2 = 0x00000100, \Delta_6 = 0x00001000, \Delta_{14} = 0x80080000$$

$$\begin{array}{c}
 \begin{pmatrix} 0 & 0 & \underline{1} & 0 \\ 0 & 0 & \underline{1} & 0 \\ 0 & 0 & \underline{0} & 0 \\ 0 & 0 & \underline{2} & 0 \end{pmatrix} \xrightarrow{\text{col}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \underline{1} & 0 \end{pmatrix} \xrightarrow{\text{row}} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 3 & 4 \end{pmatrix} \xrightarrow{\text{col}} \\
 \\
 \begin{pmatrix} 4 & 1 & 3 & 4 \\ 1 & 2 & 4 & 8 \\ 1 & 0 & 7 & 10 \\ 3 & 1 & 3 & 14 \end{pmatrix} \xrightarrow{\text{row}} \begin{pmatrix} 13 & \underline{1} & 6 & 7 \\ 11 & \underline{14} & 5 & 7 \\ 7 & \underline{4} & 14 & 5 \\ 14 & 21 & 18 & 17 \end{pmatrix} \xrightarrow{\text{col}} \begin{pmatrix} 13 & 16 & 17 & 17 \\ 6 & 16 & 19 & 23 \\ 14 & \underline{13} & 18 & 15 \\ 18 & 16 & 15 & 15 \end{pmatrix}
 \end{array}$$

Step 2: Optimal inputs (K , IV)

Integer addition in true Salsa: Difference Δ^r depends on input.

Conditions on input, such that Salsa is close to LinSalsa.

Previous differences plus simple condition on IV : first round linear.

Step 3: Optimal mask in Δ^r

Word 9 of Δ^5 is expected to be biased.

Examine each bit with χ^2 test.

Experimental results:

Randomly fix key, weak nonce, increment counter, fixed Δ

With $N = 2^{24}$ samples, average $\chi^2 = 383$, instead of $\chi^2 = 1$

Statistical weakness after 5 rounds; can be detected 2 rounds later:

Observe $X + X^7$, guess 5 key words and compute word 9 in X^5 .

Repeat for N pairs with input difference Δ

Right guess of subkey when χ^2 is large

Attack on 7 rounds?

Complexity about 2^{217} using 2^{24} pairs of keystream.
Input difference in key: related-key scenario / non-randomness of hash function F .

Attack on 6 rounds:

Complexity about 2^{177} using 2^{16} pairs of keystream.
Crowley's differential, input difference in IV only.

Analysis of Trivium

Consider augmented function:

Stream cipher with update function L , output function f .

Augmented function $S_m: F^n \rightarrow F^m$

defined by

$$S_m: x \rightarrow (f(x), f(L(x)), \dots, f(L^{m-1}(x)))$$

Update function L linear (e.g. in LFSR) or nonlinear (e.g. in Trivium).

Investigate algebraic properties of augmented function.

Update and output in TRIVIUM

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

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

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

$$z_i \leftarrow t_1 + t_2 + t_3$$

$$t_1 \leftarrow t_1 + x_{91}x_{92} + x_{171}$$

$$t_2 \leftarrow t_2 + x_{175}x_{176} + x_{264}$$

$$t_3 \leftarrow t_3 + x_{286}x_{287} + x_{69}$$

$$(x_1, \dots, x_{93}) \leftarrow (t_3, x_1, \dots, x_{92})$$

$$(x_{94}, \dots, x_{177}) \leftarrow (t_1, x_{94}, \dots, x_{176})$$

$$(x_{178}, \dots, x_{288}) \leftarrow (t_2, x_{178}, \dots, x_{287})$$

In first 66 clocks, each keystream bit linear in initial state bits.

General result from algebraic attacks says that always exist (additional) linear equations in initial state if m output bits of augmented function are fixed and m is large enough.

Can we find more linear equations in state bits that hold for a given m -bit output vector?

This could be useful in attack.

A linear equation is determined by 289 binary coefficients.

Need to find preimages for given output vector.

Finding preimages for $m = 66$ obvious.

Can find preimages efficiently for m up to 144 or larger:

Factors in products of update function have consecutive index.

Fix alternating bits of state.

Get linear relations in remaining variables.

Let c, l, q denote constant, linear, and quadratic dependence on initial state.

Assume all even bits of initial state to be c .

Evolution of states with partially fixed input

Initial state	After 1 update	After 84 updates
$R_1 = 1c1c1\dots$	$R_1 = 11c1c1\dots$	$R_1 = 11111\dots$
$R_2 = c1c1c\dots$	$R_2 = 1c1c1c\dots$	$R_2 = 11111\dots$
$R_3 = c1c1c\dots$	$R_3 = 1c1c1c\dots$	$R_3 = q1111\dots$

After update 83, bits 82 and 83 of R_2 both l .

Variable t_2 takes bits 82 and 83 to produce nonlinear term.

After update 84, $t_2 = x_{178}$ is q (nonlinear terms in t_1, t_3 appear later).

...

Keystream bit $66 + 84 = 150$ is q .

Hence first 149 keystream bits linear in remaining state bits.

Degree of freedom: 144

Potential attacks

For $m = 144$, are there additional linear equations beyond the 66 known ones?

Linear equation determined by $D_x = 289$ coefficients.

Need more than 289 preimages for given output of augmented function.

Large scale experiment:

Choose random output y of 144 bits.

Compute 400 preimages by random choice of 144 fixed bits of initial state x .

Set up matrix and solve.

Repeat for 30 other random choices of y .

Result:

Get always 66 linear equations.

Can go further: Determine preimages for m up to 150 (and even for $m = 160$) with partial search.

Still find (only) 66 linear equations for a 150 bit output of consecutive 0's.

Trivium seems resistant against additional (probabilistic) linear relations in augmented function.

Conclusions

- eSTREAM project has motivated a large number of designs of new stream ciphers.
- Novel construction principles as well as new analysis methods.
- Hope to see some promising proposals in future practical implementations.