ASK 2018, ISI Kolkata
November 13, 2018

Recent Results on Stream Ciphers

Willi Meier

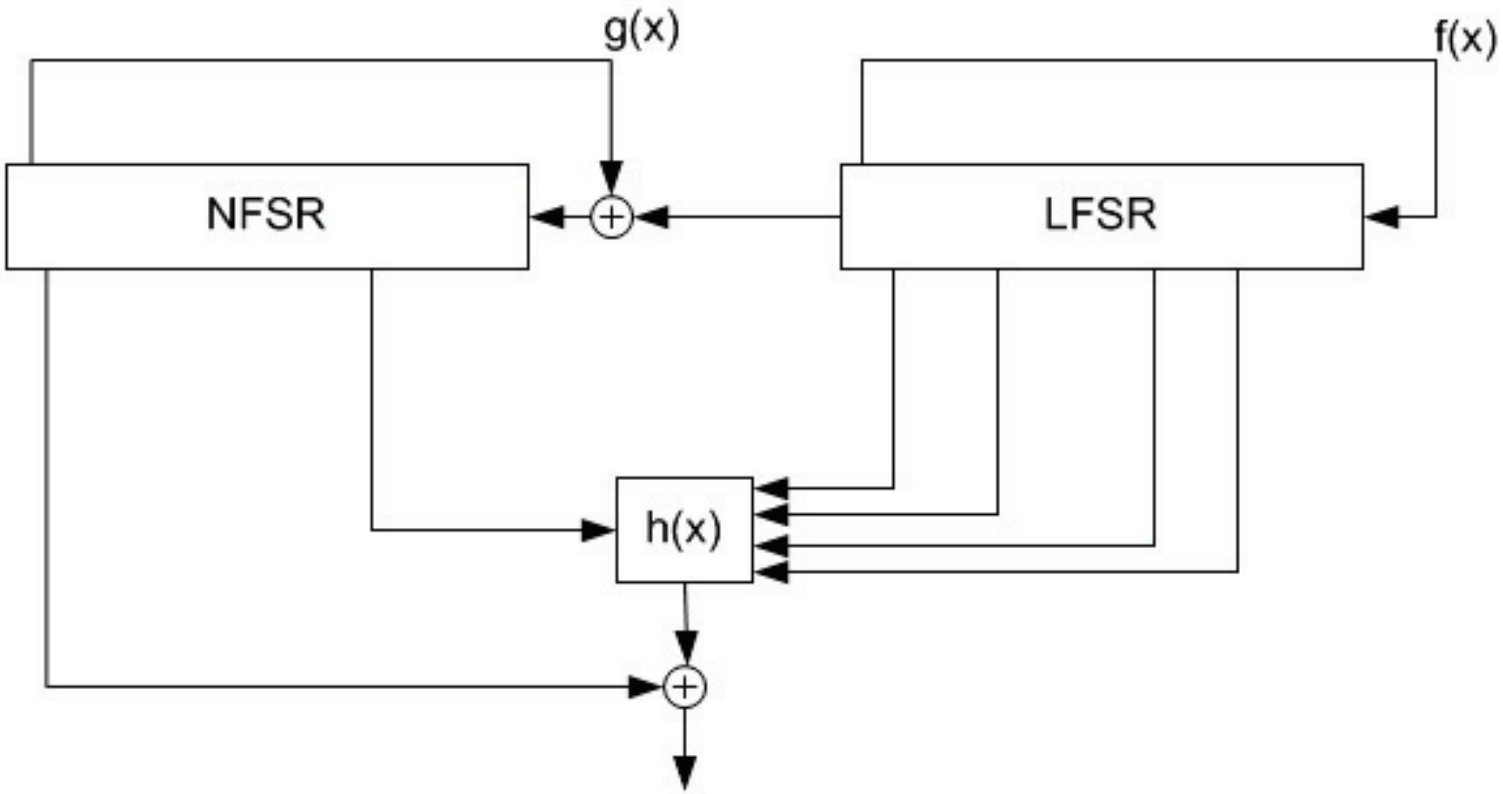
University of Applied Sciences Northwestern Switzerland
School of Engineering

Overview

- Stream Ciphers with Small State
- A generic TMD tradeoff distinguisher
- High-order differentials
- Cryptanalysis with division property
- Correlation Attacks
- Fast correlation attacks on the Grain family
- Comments/Conclusions

Stream ciphers with small state

eSTREAM finalist Grain v1: State size 160 bits, key size 80 bits.



Stream ciphers with small state

Rule: State at least twice the key size, due to time-memory-data tradeoffs (TMD-TO).

eSTREAM candidates follow this rule.

For 80 bit security, can we go lower than 160 bit state size?

One idea: Make state update key-dependent, to prevent state recovery.

Sprout (Armknecht-Mikhalev, 2015): State size only 80 bits. Modelled on stream cipher Grain v1.

Has been broken by several methods, including TMD tradeoffs and use of k-normality of Boolean functions.

Plantlet: A tweak of Sprout.

80-bit key. 90-bit IV. Simplified round key function. Larger state: 108-bit.

Stream ciphers with small state

LIZARD: modelled on Grain v1 as well.

State update independent of key, but initialization mechanism so that key recovery is provably prevented.

Security:

- Against key recovery: 2^{80}
- Complexity of generic distinguisher: 2^{60}
- Comes with security proof against key recovery based on generic TMD-TO

Use in packet mode: 16% reduced power consumption over Grain v1.

Packet length 2^{18} bits, to fit (many) application scenarios.

Stream ciphers with small state

LIZARD design

Beyond-the-birthday-bound security level of $\frac{2}{3}n$ w.r.t. generic TMD-TO's aiming at key recovery.

Security proof: Theoretical work by Hamann and Krause. Based on formal ideal primitive model.

Information-theoretic $\frac{2}{3}n$ security bound, which is tight.

Stream ciphers with small state

Differences of LIZARD to Grain v1:

- Smaller state size (121 compared to 160 bits).
- Key size: 120 bit (rather than 80 bits): necessary assumption for security proof.
- Key is introduced not only once, but twice in initialization.
- Quite different output function: Similar to FLIP stream cipher, uses many inputs.
- Two register feedbacks are both nonlinear.

Cryptanalytic results on Lizard (Banik-Isobe-Cui-Guo, FSE 2018), and (Maitra-Sinha-Siddhanti-Anand-Gangopadhyay, IEEE Trans. Computers. 2018).

Don’t contradict claims by designers.

A generic TMD tradeoff distinguisher

Jointly with Matthias Hamann, Matthias Krause and Bin Zhang.

Assume a stream cipher that continuously uses the non-volatile key in state update.

TMD tradeoffs by Babbage and by Biryukov-Shamir won't work for state recovery.

A generic distinguisher by Englund-Hell-Johansson (2007):

Allows a resynchronization collision attack.

Succeeds if part of the state that depends on both the key and IV is smaller than twice the key size.

Is motivated by analysis of OFB mode of block cipher, where size of IV space is same as size of state space.

Does not carry over directly to stream ciphers like Plantlet, e.g., IV is smaller than state.

A generic TMD tradeoff distinguisher

Assume:

Continuous-key-use (CKU) stream cipher: After initialization, key is used as additional input to state update function.

Key schedule determines way in which key influences state update, can depend on any part of state (FSRs, counters).

Key k arbitrary but fixed.

- n: Size of inner state (in bit)
- l: IV length
- 2^λ: Limit of keystream bits per IV
- I_k: Set of initial states CIPHER computes over all IVs.

A generic TMD tradeoff distinguisher

Assumption 1 (Near-Injectivity)
There are (virtually) no different IVs that produce the same keystream for a secret key k (i.e. size of $I_k \approx 2^l$).

Assumption 2 (Initial State Randomness)
Let $\sigma \approx n/2 - \lambda$. Let T denote $2^{n/2}$ keystream blocks of length $\tilde{n}$ ($\tilde{n}$ slightly larger than n), obtained from 2^σ IVs by sliding a $\tilde{n}$-bit window over each of the 2^σ keystreams of length $\leq 2^\lambda$.

Then w.h.p. a subset of $2^{n/2-(n-l)} = 2^{l-n/2}$ inner states underlying the $2^{n/2}$ keystream blocks T belong to set I_k.

A generic TMD tradeoff distinguisher

Distinguisher:

Step (1) Obtain $2^{n/2}$ keystream blocks of length $\tilde{n}$ ($\tilde{n}$ slightly larger than n) based on 2^{σ} different IVs:
Slide a $\tilde{n}$-bit window over each of the 2^{σ} keystreams of length $\leq 2^{\lambda}$ bit, and save keystream blocks.
If collision occurs: distinguish CIPHER and stop.

Step (2) For $2^{n/2}$ different IVs, obtain corresponding $\tilde{n}$-bit keystream prefix and look for collision in data created in *Step*(1).
If collision found, distinguish CIPHER and stop.

If no collision is found in Step (1) or Step (2), output RANDOM.

A generic TMD tradeoff distinguisher

Success probability derived from birthday paradox.

Assumption 1: Size of $I_k \approx 2^l$.

Assumtion 2: W.h.p. a subset of $2^{n/2-(n-l)} = 2^{l-n/2}$ inner states underlying the $2^{n/2}$ keystream blocks T collected in Step (1) belong to set I_k.

Assumption 1 assures that in Step (2) we draw uniformly at random $2^{n/2}$ elements from I_k.

Birthday paradox:
When drawing $2^{n/2}$ elements uniformly at random from a set of size 2^l (as in Step (2)), it is likely to find collision with an arbitrarily fixed subset of size $2^{l-n/2}$ (as in Step (1)).

A generic TMD tradeoff distinguisher

Complexity:

(1) Obtain $2^{n/2}$ keystream blocks of length $\tilde{n}$ bits and store them:
   - Data (keystream): $2^{n/2}$;
   - Memory (keystream blocks): $2^{n/2} \cdot \tilde{n}$;
   - Time: $2^{n/2}$.

(2) Obtain $2^{n/2}$ keystream prefixes of size $\tilde{n}$ and search for collision in data created in *Step*(1).
   - Data (keystream prefixes): $2^{n/2} \cdot \tilde{n}$;
   - Memory: negligible;
   - Time: $2^{n/2}$.

Complexity of generic TMD tradeoff distinguisher against CKU stream cipher about $2^{n/2} \cdot \tilde{n}$.

A generic TMD tradeoff distinguisher

Consequence: If key size is larger than $n/2 + \log(\tilde{n})$, distinguisher with complexity below exhaustive key search, irrespective of key scheduling!

Banik (2015): Distinguisher for Sprout cipher. Method not formalized.

A generic TMD tradeoff distinguisher

Application to Plantlet

IV space has size 2^{90}. Mapping to set I_k of initial states is injective, i.e. Assumption 1 is satisfied.

State after initialization has size $61 + 40 + 7 = 108$ bit. From definition, 7-bit counter has binary value 0...0 for all initial states.

In keystream generation, counter takes all values *mod* 80. In every 80th clock cycle counter takes value 0..0.

Each time counter takes value 0...0, have chance of 2^{90-101} (101 bit is combined size of FSRs) that there is a IV which generates this state as initial state.

Picking any $(108 + \epsilon)$-bit keystream block, have chance of $80^{-1} \cdot 2^{90-101} > 2^{-18}$ that underlying 108-bit state is an initial state produced by some IV. Hence Assumption 2 holds.

A generic TMD tradeoff distinguisher

Complexity of distinguishing attack applied to Plantlet:

- Data: $2^{108/2} + 2^{108/2} \cdot (108 + 20) \approx 2^{61}$;
- Memory: $2^{108/2} \cdot (108 + 20) = 2^{61}$;
- Time: $2^{108/2} + 2^{108/2} = 2^{55}$.

Added 20-bit margin to block size to avoid false positives.

Complexity for generic distinguisher. May be improved slightly when exploiting known counters in Plantlet.

Stream ciphers that continuously use the IV

Problem: How to avoid TMD tradeoff distinguishers?

Simple countermeasure: Increase state of stream cipher.

Can we do even with small state?

Idea: Use stream cipher in packet mode together with continuously involving the IV in state update: Continuous-IV-Use (CIU) Stream Cipher.

No round keys necessary.

Stream ciphers that continuously use the IV

Security of CIU stream cipher against TMD tradeoff distinguishers: Two scenarios conceivable.

Based on:

(1) state recovery;

(2) collisions in keystream.

Can provide arguments that stream cipher in packet mode that continuously involves IV in state update is able to resist attacks of type (1) and (2).

Trivium

Designed by De Cannière and Preneel in 2005.

- 80-bit secret key and 80-bit initial value IV (public)
- 3 quadratic NLFSRs, of different lengths
- State size: 288 bit
- 1152 initialization rounds before output is produced
- Increased efficiency by factor up to 64: Implement Boolean functions in parallel
- Linear output function.

Trivium

Initialization:

$(s_1, s_2, ..., s_{93}) \leftarrow (k_0, ..., k_{79}, 0, 0, ..,)$
$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (x_0, x_1, ..., x_{79}, 0., , , , 0)$
$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (0, 0, ..., 0, 1, 1, 1)$
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

Trivium

Output generation:

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

Trivium

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.

Other designs inspired by Trivium:

- KATAN block cipher,
- Kreyvium stream cipher for Efficient Homomorphic-Ciphertext Compression (CCFLNPS, FSE 2016).

High-order differentials

Let V be a linear subspace of $\{0,1\}^n$ of dimension d.

Boolean function

$$f : \{0,1\}^n \mapsto \{0,1\}.$$

Derivative of order d of f with respect to V:

$$\Delta_V f(x) = \sum_{v \in V} f(x+v).$$

High-order differentials

Methods based on high-order differentials:

Key recovery with derived functions (FKM, 2008).

Cube attack, Vielhaber, Dinur-Shamir 2008.

Cube: Difference vectors are formed out of all elements in a binary cube of some dimension.

Cube testers, 2009. Test nonrandomness in high degree monomial distribution.

Detect nonrandomness of reduced round intialization of Trivium up to 885 rounds.

Dynamic cube attack on full round Grain-128 by Dinur-Shamir.

High-order differentials

Conditions in high-order differentials?

Impose conditions on ”basic differences” involved in summation, viewed as first-order difference.

Conditions assure that differential path is followed over many rounds.

May involve IV bits and key bits.

Many differences involved: Conditions for differences may contradict each other.

High-order differentials

Careful analysis of conditions for high-order differentials in Trivium initialization:

Practical distinguisher for 961 out of 1152 rounds (KMN 2012).

IV-bits x_{72}, x_{78} neutral for a subset of (weak) keys in a derivative of order 24.

Subset contains 2^{26} weak keys for 961 rounds.

No answer yet whether number of initialization rounds in Trivium suffice.

High-order differentials often heuristic thus far.

Limited by computing power: Summation over 2^d outputs, where d is dimension of cube formed by inputs.

Cryptanalysis with division property

Joint work with Yosuke Todo, Yonglin Hao, Qingju Wang, Takanori Isobe and Chaoyun Li.

New generic tool for cube attacks.

Enables to exploit internal structure of stream ciphers.

Division property: Tool to find integral distinguishers in block ciphers.

Application to stream ciphers: Zero-sums, i.e., sum of outputs over certain cubes are 0. A distinguisher with no practical limitation on size of cube.

Cryptanalysis with division property

Nontrivial how to find information on key bits: For most stream ciphers, no key scheduling as in block ciphers.

Key is not involved in state update after loading.

Division property can be used to find information on the algebraic normal form of ”superpoly”.

Key bits that are not involved in superpoly can be determined.

Cryptanalysis with division property

Cube attack on $f(k, x)$, k secret, x public:

For index set $I \subset \{1, .., n\}$ define t_I as monomial containing all public variables (cube variables), with index in I.

For fixed I, there is unique polynomial p such that ANF of $f(k, x)$ can be written as

$$f(k, x) = t_I p(k, x) + q(k, x),$$

where p does not contain any cube variable, and no monomial in q is divisible by monomial t_I.

p: superpoly of I in f.

Can compute superpoly by summing the outputs of f over all possible configurations of the cube variables.

Classical cube attack succeeds if superpoly is linear in key bits for a cube of size less than 50.

Cryptanalysis with division property

Division property proposed at Eurocrypt 2015 (Yosuke Todo)

Definition ((Bit-Based Division Property)

Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. Let $\mathbb{K}$ be a set whose elements take an n-dimensional bit vector. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, it fulfils the following conditions:

$$\bigoplus_{\vec{x}\in\mathbb{X}} \vec{x}^{\vec{u}} = \begin{cases} \text{unknown} & \text{if there exist } \vec{k} \in \mathbb{K} \text{ s.t. } \vec{u} \succeq \vec{k}, \\ 0 & \text{otherwise,} \end{cases}$$

where $\vec{u} \succeq \vec{k}$ if $u_i \geq k_i$ for all i, and $\vec{x}^{\vec{u}} = \prod_{i=1}^{n} x_i^{u_i}$.

Cryptanalysis with division property

In bit-based division property, three propagation rules (copy, xor, and) are defined. Can evaluate any circuit.

Attackers determine indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, n\}$ and prepare $2^{|I|}$ chosen plaintexts.

Variables indexed by I are taking all possible combinations of values.

The division property of such chosen plaintexts is $\mathcal{D}^{1^n}_{\vec{k}}$, where $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise.

Then, the propagation of the division property from $\mathcal{D}^{1^n}_{\vec{k}}$ is evaluated as

Cryptanalysis with division property

$$\{\vec{k}\} \stackrel{\text{def}}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r,$$

where $\mathcal{D}_{\mathbb{K}_i}$ is the division property after i-round propagation.

If the division property $\mathbb{K}_r$ does not have a unit vector $\vec{e}_i$ whose only ith element is 1, the ith bit of r-round ciphertexts is balanced.

Evaluating the propagation of the division property is not easy because the size of $\mathbb{K}_i$ extremely increases.

Cryptanalysis with division property

Propagation search using MILP (Xiang et al AC16) MILP: Mixed Integer Linear Programming.

Models the three propagations

Definition (Division Trail)

Consider the propagation of the division property $\{\vec{k}\} \stackrel{\text{def}}{=} \mathbb{K}_0 \to \mathbb{K}_1 \to \mathbb{K}_2 \to \cdots \to \mathbb{K}_r$. Moreover, for any vector $\vec{k}^*_{i+1} \in \mathbb{K}_{i+1}$, there must exist a vector $\vec{k}^*_i \in \mathbb{K}_i$ such that $\vec{k}^*_i$ can propagate to $\vec{k}^*_{i+1}$ by the propagation rule of the division property. Furthermore, for $(\vec{k}_0, \vec{k}_1, \ldots, \vec{k}_r) \in (\mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r)$ if $\vec{k}_i$ can propagate to $\vec{k}_{i+1}$ for all $i \in \{0, 1, \ldots, r-1\}$, we call $(\vec{k}_0 \to \vec{k}_1 \to \cdots \to \vec{k}_r)$ an r-round division trail.

Cryptanalysis with division property

Let E_k be the target r-round iterated cipher. Then, if there are division trails $\vec{k}_0 \xrightarrow{E_k} \vec{k}_r = \vec{e}_i$, cannot know whether the ith bit of r-round ciphertexts is balanced or not.

On the other hand, if we can prove that there is no division trail $\vec{k}_0 \xrightarrow{E_k} \vec{e}_i$, the ith bit of r-round ciphertexts is always balanced.

Have to evaluate all possible division trails to verify whether each bit of ciphertexts is balanced or not.

Previously, all possible division trails were evaluated by using a breadth-first search. Search requires enormous memory and time complexity. Therefore, it is practically infeasible to apply this method to iterated ciphers whose block length is not small.

Cryptanalysis with division property

MILP can efficiently solve this problem.

Generate an MILP model that covers all division trails, and the solver evaluates the feasibility whether there are division trails from the input division property to the output one or not.

If the solver guarantees that there is no division trail, higher-order differential (integral) characteristics are found.

Important relation between division property and ANF of superpoly:

Cryptanalysis with division property

Proposition

Let $f(\vec{x}, \vec{v})$ be a polynomial, where $\vec{x}$ and $\vec{v}$ denote the secret and public variables, respectively.

For a set of indices $I = \{i_1, i_2, \ldots, i_{|I|}\} \subset \{1, 2, \ldots, m\}$, let C_I be a set of $2^{|I|}$ values where the variables in $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{|I|}}\}$ are taking all possible combinations of values.

Let $\vec{k}_I$ be an m-dimensional bit vector such that $\vec{v}^{\vec{k}_I} = t_I = v_{i_1} v_{i_2} \cdots v_{i_{|I|}}$, i.e. $k_i = 1$ if $i \in I$ and $k_i = 0$ otherwise.

Assuming there is no division trail such that $(\vec{e}_\lambda, \vec{k}_I) \xrightarrow{f} 1$, x_λ is not involved in the superpoly of the cube C_I.

Proposition can be used to identify (hopefully small) index set J such that key bit k_i is not involved in superpoly if i is not in J.

Cryptanalysis with division property

Key Recovery Attack Procedure:

Choose suitable index set I for cube, and index set J for potentially involved key bits.

Steps as follows:

Cryptanalysis with division property

Phase 1 **Offline Phase: Identify the superpoly.** Assign the non-cube IVs a proper constant value, and prepare a cube by flipping bits in I. Then, for all possible values of the secret variables $\{x_{j_1}, x_{j_2}, \ldots, x_{j_{|J|}}\}$, compute and store the value of the superpolys as $p_{\vec{v}}(\vec{x}) = \bigoplus_{C_I} f(\vec{x}, \vec{v})$. The $2^{|J|}$ values compose the truth table of $p_{\vec{v}}(\vec{x})$ and the ANF of the superpoly is determined accordingly. Search for the preferable superpoly (its truth table is balanced) by changing the value of the non-cube IV.

Cryptanalysis with division property

Phase 2 **Online Phase: Recover the part of secret variables.** After the truth table of the superpoly and the proper assignments of non-cube IVs are given, attackers query the cube C_I to encryption oracle and get one bit of $p_{\vec{v}}(\vec{x})$ through summation. Then, we get one polynomial about involved secret variables, and half of the values in involved secret variables is discarded because the superpoly is balanced.

Phase 3 **Brute-force search phase.** Attackers guess the remaining secret variables to recover the entire value in secret variables.

Cryptanalysis with division property

Phase 1 takes $2^{|I|+|J|}$ encryptions to construct a truth table of size $2^{|J|}$.

Phase 2 requires $2^{|I|}$ encryptions to sum over the cube C_I. Additional $2^{|J|}$ table lookups are also necessary in order to identify the $2^{|J|-1}$ candidate keys but such a complexity is negligible in comparison with the cube summation so the complexity of Phase 2 can be regarded as $2^{|I|}$.

The complexity of Phase 3 is $2^{\kappa-1}$, where κ equals key size, if only one cube is used.

The attack can be meaningful only if $|I| + |J| < \kappa$.

Several improvements on this method possible.

Cryptanalysis with division property

Methods allow to outperfom all known results for a number of interesting ciphers.

Cryptanalysis with complexity lower than exhaustive key search on:

Trivium: 839 rounds (out of 1152 initialization rounds).

Kreyvium: 891 (out of 1152) rounds.

Grain-128a: 184 (out of 256) rounds.

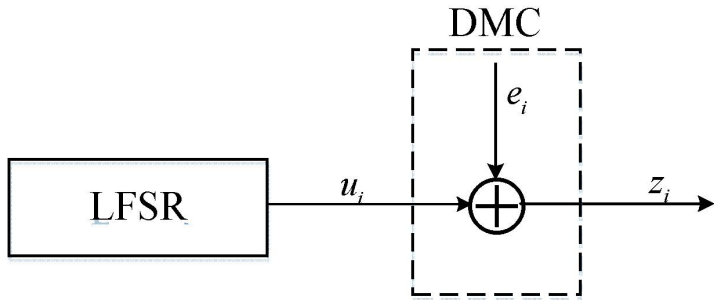
Acorn: 750 (out of 1792) rounds.

TriviA-SC1: 1009 rounds, TriviA-SC2: 1004 rounds.

Morus-640: 7 (out of 16) rounds distinguisher. Work by Tao Ye and Yongzhuang Wei, using further refinements of methods.

Correlation attacks

Exploit statistical correlation between output and some inputs of a Boolean function in state recovery attack.



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

$$z_m = u_m + e_m \bmod 2.$$

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

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

Correlation attacks

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

$$T = \#\{j | z_j = u_j, 0 \leq j \leq N\}$$

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

Feasible for L up to about 50.

Improved procedure: Fast correlation attacks.

Fast correlation attacks on the Grain family

Joint work with Yosuke Todo, Takanori Isobe, Kazumaro Aoki and Bin Zhang.

Classical fast correlation attacks based on parity check equations created from feedback polynomial of LFSR (R. Gallager, Low-density parity check codes, 1963, MS 1988, CJM 2003, Jönsson-Johansson)

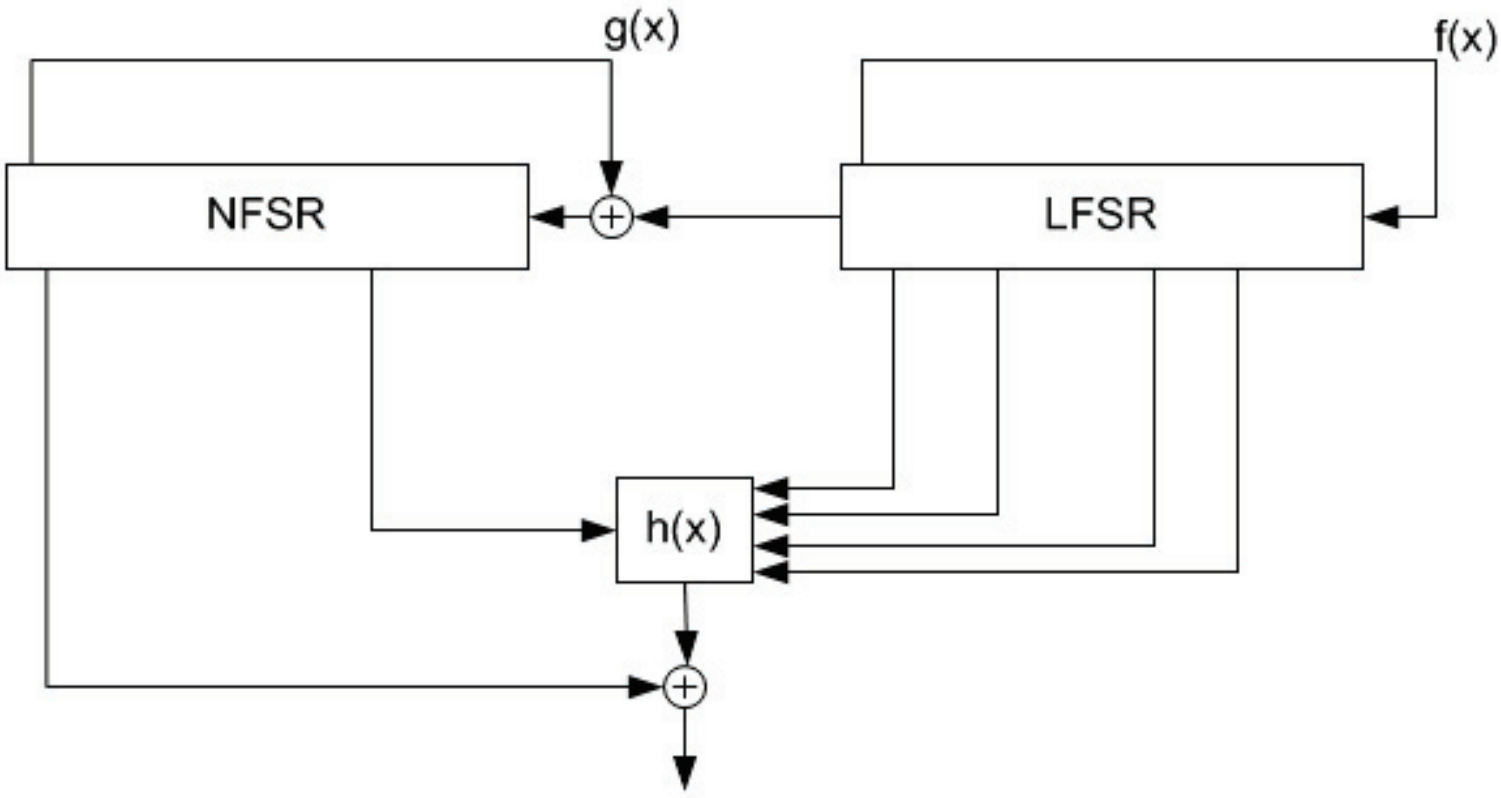
Exploit correlation between keystream bits and state bits of LFSR.

Avoid exhaustive search over full state of the LFSR.

Usually, in modern stream ciphers, such correlations are very small, if they exist at all.

In Grain v1, (virtually) no correlation between keystream bits and state of the LFSR, and similar in Grain-128a.

Fast correlation attacks on the Grain family



No attacks known on keystream generation mode of Grain v1 and Grain-128a (but dynamic cube attack on Grain-128).

Fast correlation attacks on the Grain family

It turns out however, that well chosen sums of keystream bits are correlated (slightly) to sums of secret state bits of the LFSR part.

Similar to linear cryptanalysis: There are a multitude of biased linear masks.

Find a new method to reduce data and time complexity.

New fast correlation attack on Grain v1 with complexity about 2^{76}, and on Grain-128a with complexity 2^{115} (CRYPTO 2018, Todo et. al.)

Questions and Comments

- Can division property be combined with high-order conditional differentials?
- How many initailization rounds of Trivium initialization are at least needed for its security?
- New design criteria for initialization process (to achieve designs with less initialization rounds)?
- NIST lightweight cryptography competition: Will there be proposals using design principles and analysis of stream ciphers?