Analysis of Lightweight Stream Ciphers

PhD public defense

Simon Fischer
EPFL and FHNW

April 18th 2008

Outline

Part 1
Introduction

What is Cryptology?

Cryptology: protect information against unauthorized users.

Two main directions:

1. **Authentication**: Information originates from authorized user.
2. **Confidentiality**: Information is secret to unauthorized user.

Applications of cryptology:

- Electronic banking
- Biometric passports
- Wireless communication
- Secured computer networks
- Embedded systems

Confidentiality

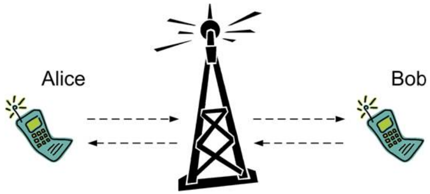
Confidentiality can be achieved with symmetric encryption.

Alice and Bob exchange a secret key and encrypt/decrypt messages.
An adversary tries to decrypt a ciphertext without knowing the key.

Some time ago: "Hello" → "Khoor" (Caesar cipher, shift 3 positions).

Today: Use a secure *block cipher* or *stream cipher*.

The cipher must be very fast for real-time applications (and still secure).



Stream Cipher

Stream ciphers are based on the *One-Time-Pad*:

key:	001010100010111 . . .	(exchange random bits)
	XOR	
plaintext:	000000100010001 . . .	(binary message)
ciphertext:	001010000000110 . . .	

Good things first: From the ciphertext, the adversary can never get the message $\rightarrow$ perfect security!

Bad thing: The secret key is as large as the message $\rightarrow$ impractical.

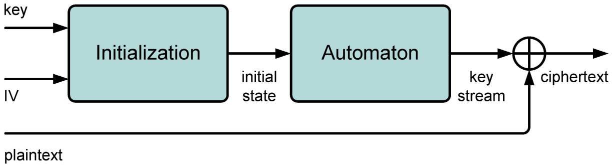
The General Model

Stream cipher:

1. Exchange a small secret key (e.g. 128 bits).
2. Produce a long pseudo-random keystream (e.g. millions of bits).

Two Components:

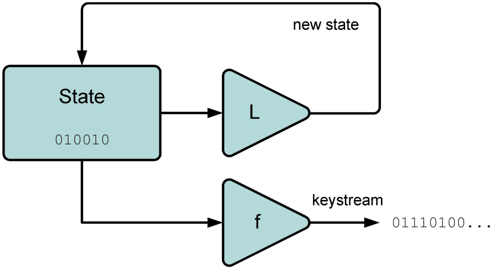
Initialisation: key, IV → initial state
Automaton: initial state → keystream



Automaton of a Stream Cipher

Automaton:

Update function L: state $\rightarrow$ new state
Filter function f: state $\rightarrow$ keystream



Use only few operations: fast and low HW requirements.
Examples: A5 in GSM / Bluetooth / sensor networks / RFID

Attacks

Security not well established, new designs in eSTREAM project.

$$\boxed{\text{Our contribution in this thesis is to attack stream ciphers.}}$$

What is an a attack on a stream cipher?

Assume that keystream is known (known-plaintext attack):

a) Recover the internal state (equivalent to key recovery in many cases).
b) Or distinguish the keystream from uniformly random.

For a secret key of n bits, correct key can be found by trying all 2^n keys.

Any attack faster than 2^n simple operations breaks the cipher (which does not mean that the attack is practical).

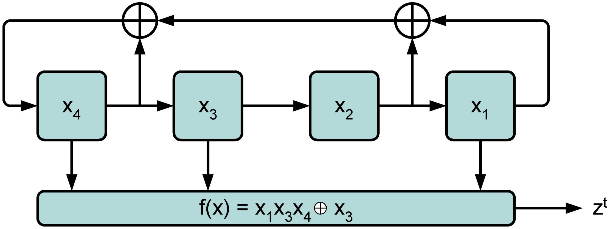
Part 2
Algebraic Attacks

Filter Generator

Focus here on the classical filter generator.

Update function L: A linear feedback shift register (LFSR), it can have maximum period and good statistical properties.

Filter function f: A nonlinear Boolean function of some degree.
It takes a few bits from the state and produces one bit of keystream.



Algebraic Equations

A filter generator is defined by a system of algebraic equations.
A solution of this system gives the secret key.

Our system of equations:

$$\begin{aligned} f(x) &= z^0 \\ f(L(x)) &= z^1 \\ f(L^2(x)) &= z^2 \\ &\vdots \end{aligned}$$

Example:

$$\begin{aligned} x_1x_3x_4 \oplus x_3 &= 0 \\ x_2x_3x_4 \oplus x_1x_3x_4 &= 1 \\ x_1x_3 \oplus x_2x_4 &= 1 \\ &\vdots \end{aligned}$$

Have many nonlinear equations with n variables and degree at most $\deg f$.
Solution with linearization: new variable for all $\binom{n}{\deg f}$ monomials.

Algebraic Attacks

Problem: Simple linearization is not efficient if $\deg f$ is large.
Idea: Reduce degree of equations $\rightarrow$ **algebraic attacks**.

Find g of small degree d, such that $f \cdot g = 0$. Obtain new equations of degree d:

$$\begin{aligned} f(L^t(x)) &= z^t & &| \cdot g(L^t(x)) \\ 0 &= g(L^t(x)) & &\text{if } z^t = 1 \end{aligned}$$

Attack: Time complexity with linearization is $\binom{n}{d}^3$, data $\binom{n}{d}$.

Algebraic immunity $\mathcal{AI}$ is minimum degree of annihilator for f or $f \oplus 1$.
See e.g. [Armknecht-Carlet-Gaborit-Künzli-Meier-Ruatta 06].

Fast Algebraic Attacks

FAA's [Courtois 03]: Improvement of AA's [Courtois-Meier 03].

1. Find g and h of low degrees e and d, such that $\boxed{f \cdot g = h}$
2. Precompute a linear combination for all t

$$\bigoplus_{i=0}^{D} c_i \cdot h(L^{t+i}(x)) = 0$$

3. This gives an equation of degree e

$$\bigoplus_{i=0}^{D} c_i \cdot g(L^{t+i}(x)) \cdot z^{t+i} = 0$$

Time complexity $\binom{n}{e}^3$ for solving, e can be much smaller than d.
Data complexity $D = \binom{n}{d}$ can be the same as in AA's.

New Algorithm for Arbitrary Functions

Goal: Determine existence of (g, h) for fixed degrees (e, d).

We have a theorem to isolate single coefficients of h. With clever choice of parameters, can solve small system in coefficients of g only.

New and very efficient algorithm to determine immunity against FAA's.

Question: Some functions have large $\mathcal{AI}$ (immune against AA's).
Are they also immune against FAA's?

Experiments with DGM functions of maximum $\mathcal{AI}$ [FSE 05].
Apply our algorithm up to 10 inputs, find $\boxed{d = \mathcal{AI}, e = 1}$

Symmetric Functions

Symmetric functions:

- Output $f(x)$ depends only on weight of input x
- Suitable for efficient HW implementations

Have algorithms for symmetric functions which are much more efficient. Successfully applied to many symmetric functions with maximum $\mathcal{AI}$.

Theoretical result for Majority functions (with maximum $\mathcal{AI}$):

- Can predict degrees of g and h for any n.
- For example, for 9 inputs it is $\boxed{d = \mathcal{AI}, e = 1}$

Some functions with maximum $\mathcal{AI}$ are very vulnerable to FAA's.

Augmented Function

Recall: FAA's need (at least) as much data as AA's.

Q: Can we improve AA's to reduce data complexity?
A: Use multiple output bits to find new equations.

Approach: Construct the **augmented function** (AF) of a stream cipher

$$x \mapsto (f(x), f(L(x)), \ldots, f(L^{m-1}(x))$$

- Find equations for AF with fixed output z of m bit.
- These equations are called **conditional equations** (CE's).
- CE's are equations in x which hold true for all preimages of z.

How to Find Conditional Equations

Use matrix approach to find CE's [Courtois 03].

Example: AF with $n = 3$, assume some output z with preimages $x =$100, 110, 011, 001. Find linear CE.

$$M = \begin{array}{cccc|c} 1 & x_1 & x_2 & x_3 & \text{preimages} \\ \hline 1 & 1 & 0 & 0 & x = 100 \\ 1 & 1 & 1 & 0 & x = 110 \\ 1 & 0 & 1 & 1 & x = 011 \\ 1 & 0 & 0 & 1 & x = 001 \end{array}$$

Solution: $0 = 1 \oplus x_1 \oplus x_3$ holds for each preimage.

How to Find Conditional Equations

Use matrix approach to find CE's [Courtois 03].

Example: AF with $n = 3$, assume some output z with preimages $x =$100, 110, 011, 001. Find linear CE.

$$M = \begin{array}{cccc|c} 1 & x_1 & x_2 & x_3 & \text{preimages} \\ \hline 1 & 1 & 0 & 0 & x = 100 \\ 1 & 1 & 1 & 0 & x = 110 \\ 1 & 0 & 1 & 1 & x = 011 \\ 1 & 0 & 0 & 1 & x = 001 \end{array}$$

Solution: $0 = 1 \oplus x_1 \oplus x_3$ holds for each preimage.

Algorithmic Methods

Find CE's by setting up and solving the matrix.
Bottleneck: finding all preimages takes 2^n steps.

Probabilistic algorithm:
Compute only a few random preimages. A random preimage of m-bit output is found in 2^m, sometimes faster with sampling.

Under favorable conditions, CE already exists for small m (where matrix method is efficient). Can choose parameters s.th. CE holds true for a random preimage with large probability.

Results for Filter Generators

Experimental Results:

- Several filter functions f are shown to be **weak**:
  Many low degree equations can be determined efficiently (i.e. they exist already for small block sizes m).
- Each new low degree equation (found by investigating AF) can serve to reduce data complexity of algebraic attacks.
- In some cases, data complexity can be of order n (with time complexity below exhaustive search).
- Have identified functions f which show **resistance** to this approach:
  Low degree eq's exist only for large m, effort for preimages up to 2^n.

A Related Attack on ASG

Alternating Step Generator [Günther 87]:
One LFSR of n bits controls clock of two stop/go LFSR's of n bits each.

Recover one stop/go LFSR [Golic-Menicocci 06], [Johansson 98]:

- Can efficiently compute preimages for fixed output of m bits
- Distribution of preimages strongly correlated to output
- Compute only $T < 2^n$ preimages such that correct one is included

We find $T = 2^H$ with entropy H conditioned by output.

Example:
For $n = 42$ and output of $m = 84$ and weight $w = 14$, it is $H = 24.16$.

New attack with many tradeoffs, best known attack so far.
For $n = 80$, reduce data complexity by factor 2^{13}.

Part 3
Attacks on T-functions

Motivation

Filter Generators (with linear update):

- ✓ Very elegant and efficient (basis of other constructions)
  LFSR has good statistical properties
- ✗ Security strongly relies on filter
  Can be vulnerable to AA and FAA

Natural question: Can we replace LFSR with nonlinear primitive?
Potential answers: NFSR's, FCSR's, Nonlinear T-functions, ...

What is a T-function?

- TF is triangular mapping from n-bit to n-bit, bit i of output depends only on bits $0, 1, \ldots, i$ of input
- Definition naturally extends to multiword mapping (bit-slices)

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Example: arbitrary (crazy) composition of $+$, $\times$, $\oplus$, $\vee$, $\wedge$, $\ldots$

- Such compositions are **fast** in SW and dedicated HW
- TF have many **analyzable** mathematical properties (invertible, single-cycle, $\ldots$)

Definition of TF-0M

Stream cipher TF-0M, proposed by Klimov and Shamir [FSE 04].

State: 4 words x_i of 64 bits each
Update:

$$L : \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_0 & + & (s_0^2 \vee C_0) & & \\ x_1 & + & (s_1^2 \vee C_1) & + & \kappa_0 \\ x_2 & + & (s_2^2 \vee C_2) & + & \kappa_1 \\ x_3 & + & (s_3^2 \vee C_3) & + & \kappa_2 \end{pmatrix}$$

(s_i are simple functions of x, C_i are constants, κ_i are carry bits)

Output: $f(x)$ are first 32 msb's of x_3

Update is **single-cycle** TF with period 2^{256}, intended security is 2^{128}.

Relationship to the Square Mapping

Consider the **integer difference** of two successive outputs of TF-0M

$$f(L(x)) - f(x) = f(s_3^2) + c \mod 2^m$$

Related to square mapping and expected to be biased.

Compute distribution:

- Approximate with pure square mapping $z = f(y^2) \mod 2^m$ (uniformly random y, ignore carry)
- Impossible to compute distribution of PSM for $n = 64$. Compute distribution for **small** word sizes $n \ldots$

Distinguishing Attack

Observe that following words occur two times more than expected

$$z = 2^{\frac{n-8}{2}} \cdot i^2 \quad \text{for } i = 0, 1, 2, 3$$

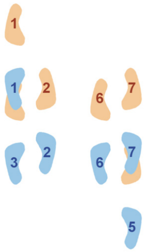
Many more **aggregates** of words with constant bias exist.
Observation seems to be independent of n.

Approximate distribution of $z = f(y^2) \mod 2^m$ in very compact way.
Measure the imbalance for $n = 64$, expected data complexity of 2^{32}.

Can apply same distinguisher for integer differences of TF-0M outputs.
Practical distinguishing attack, despite large state of 256 bits.

We have successfully analysed 3 other proposals based on T-functions.

Part 4
Attacks on Salsa20



Motivation

- ✓ Have well-known LFSR based designs
- ✓ Replace LFSR with nonlinear driving devices
- ✗ Can be critical, as seen for TF

Other approach:

- ▶ Design of a unique stream cipher (not based on LFSR's)
- ▶ Can be related to well-known methods from block ciphers
- ▶ Should be much faster than block ciphers in counter mode

Salsa20

eSTREAM focus candidate [Bernstein 05].

State: matrix of 16 words of 32 bits.
Update: increment a counter.
Output: modify below-diagonal word first. . .

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

Salsa20

eSTREAM focus candidate [Bernstein 05].

State: matrix of 16 words of 32 bits.
Update: increment a counter.
Output: modify below-diagonal word first...

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

Salsa20

eSTREAM focus candidate [Bernstein 05].

State: matrix of 16 words of 32 bits.
Update: increment a counter.
Output: modify below-diagonal word first...

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

$$+, \lll$$

Salsa20

eSTREAM focus candidate [Bernstein 05].

State: matrix of 16 words of 32 bits.
Update: increment a counter.
Output: modify below-diagonal word first...

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

$$\oplus$$

Salsa20

eSTREAM focus candidate [Bernstein 05].

State: matrix of 16 words of 32 bits.
Update: increment a counter.
Output: modify below-diagonal word first...

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

$$\oplus$$

- Repeat for all words in columns, then in rows
- 10 rounds columns, 10 rounds rows
- Output the keystream $Z = X^0 + X^{20}$ of 512 bits

Initialisation

Initialisation:
Trivial, fill state with (key, counter, IV) where counter $= 0$

$$X^0 = \begin{pmatrix} \text{const} & \text{key} & \text{key} & \text{key} \\ \text{key} & \text{const} & \text{IV} & \text{IV} \\ \text{counter} & \text{counter} & \text{const} & \text{key} \\ \text{key} & \text{key} & \text{key} & \text{const} \end{pmatrix}$$

Consider version with key size of 256 bits.

Analyse reduced variants, e.g. 7 rounds with $Z = X^0 + X^7$.

Differential Attack

Reduced variants are analysed with differential methods [Crowley 05], [Fischer-Meier-Berbain-Biasse-Robshaw 06]:

Step 1: Precomputation of forwards truncated differential

- Fix single-bit input-difference (ID) in IV
- Observe single-bit output-difference (OD) in X^r
- For random IV's, this differential has bias ε_d

$$
\begin{array}{ccccccccc}
X^0 & \rightarrow & X^1 & \rightarrow & X^2 & \rightarrow & X^3 & \rightarrow & X^4 \\
Y^0 & \rightarrow & Y^1 & \rightarrow & Y^2 & \rightarrow & Y^3 & \rightarrow & Y^4 \\
\hline
\Delta^0 & \rightarrow & \ldots & & & & & \rightarrow & \Delta^4
\end{array}
$$

Differential Attack

Step 2: Observe ε_d a few rounds later with backwards computation

- Collect keystream pairs $Z = X^0 + X^R$ for random IV's and fixed ID
- Guess part of the key (required in X^0)
- Backwards computation to get biased OD in $X^r = (Z - X^0)^{r-R}$
- For correct key, a bias ε_d is measured

$$\begin{array}{ccccccccc} X^4 & \leftarrow & X^5 & \leftarrow & X^6 & \leftarrow & X^7 & \xleftarrow{\text{key}} & X^7 + X^0 \\ Y^4 & \leftarrow & Y^5 & \leftarrow & Y^6 & \leftarrow & Y^7 & \xleftarrow{\text{key}} & Y^7 + Y^0 \\ \hline \Delta^4 & & & & & & & & \end{array}$$

Approximation

Classical way: try all 2^{256} keys, test each key with N samples (N depends on ε_d and desired p_{fa}).

New idea: Find approximation of backwards computation which

- depends only on $m < 256$ key bits
- and with a (significant) bias of ε_a

This reduces the search space to 2^m but increases N, because approximation introduces additional bias (with overall bias $\varepsilon = \varepsilon_d \cdot \varepsilon_a$).

Requires $N2^m$ steps to test subkeys, identify set of $p_{\mathsf{fa}}2^m$ candidate subkeys, verify full key with search over remaining key part.

Total complexity: $C = N2^m + 2^{256}p_{\mathsf{fa}}$.

Probabilistic Neutral Bits

How to identify a suitable approximation?

Approach:
Divide key bits in two sets: significant key bits and PNB's.
Approximation is defined by replacing PNB's by fixed bits.

Definition

Let γ_i be the bias that complementing the key bit k_i does not change the value of backwards computation (neutrality measure).

Significant key bits: all key bits k_i with $|\gamma_i| <$ threshold.

This is our method to compute approximations, can apply the attack.

Experimental Results

For every variant, find optimal differentials with many PNB's.

Salsa20/7:
Attack in 2^{152} and $N = 2^{27}$ (previous attack in 2^{190}). Use a 4-rounds differential, compute 3 rounds backwards with 131 PNB's.

Salsa20/8: best known attack on Salsa20 so far.
Attack in 2^{251} and $N = 2^{30}$. Use a 4-rounds differential, compute 4 rounds backwards with 36 PNB's.

ChaCha7:
Attack in 2^{248} and $N = 2^{27}$. Use a 3-rounds differential, compute 4 rounds backwards with 35 PNB's.

Related result: collision for 3 rounds of Rumba with multibit-differential.

Part 5
Chosen IV Attacks

Motivation

Initialisation: key, IV $\rightarrow$ initial state

Efficiency: Often constructed by a number of simple rounds.
Security: Each state bit should depend on each key and IV bit in a complex way.

Tradeoff in the number of rounds for large diffusion.

If diffusion is not sufficient, there exists a framework of chosen IV statistical distinguishing attacks [Saarinen 06]: obtain a small function by varying a few IV bits, measure imbalance of a fixed monomial.

Open problem: Is this framework useful for key recovery attacks?

Key Recovery Attacks

Approach: Some key bits in the coefficient of a monomial may have limited influence. Can identify PNB's and define approximation which depends on subkey.

Find key recovery attacks on two reduced eSTREAM focus candidates.

Trivium with $672/1152$ rounds:
Vary on 11 bits, identify 51 PNB's, recover subkey of 29 bits in 2^{55}.

Grain-128 with $180/256$ rounds:
Vary on 7 bits, identify 18 PNB's, recover key in 2^{124}.

Conclusions

Conclusions

There is a need for stream ciphers in lightweight applications.

LFSR-based designs can be vulnerable to algebraic attacks.

Replacing LFSR with TF was not very successful.

For other nonlinear driving devices, analysis is difficult for both designer and attacker.

Some unique designs are very promising.

Publications

1. S. Künzli, P. Junod, and W. Meier. Distinguishing Attacks on T-functions. In MYCRYPT 2005.
2. F. Armknecht, C. Carlet, P. Gaborit, S. Künzli, W. Meier, and O. Ruatta. Efficient Computation of Algebraic Immunity for Algebraic and Fast Algebraic Attacks. In EUROCRYPT 2006.
3. S. Fischer, W. Meier, C. Berbain, J.-F. Biasse, and M. Robshaw. Non-randomness in eSTREAM Candidates Salsa20 and TSC-4. In INDOCRYPT 2006.
4. S. Fischer and W. Meier. Algebraic Immunity of S-boxes and Augmented Functions. In FSE 2007.
5. S. Khazaei, S. Fischer, and W. Meier. Reduced Complexity Attacks on the Alternating Step Generator. In SAC 2007.
6. J.-Ph. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. In FSE 2008.
7. S. Fischer, S. Khazaei, and W. Meier. Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In AFRICACRYPT 2008.
8. S. Fischer, W. Meier, and D. Stegemann. Equivalent Representations of the F-FCSR Keystream Generator. In SASC 2008.

Thank you!