## Stream Ciphers: Cryptanalytic Techniques

#### Thomas Johansson

#### Department of Electrical and Information Technology. Lund University, Sweden

#### ECRYPT Summer school 2007

- Introduction and preliminaries, ideas for cryptanalysis
- Generic approaches: Statistical attacks, Time-memory tradeoff
- Linear complexity, correlation attacks, linear approximation attacks
- Case study: Achterbahn
- Algebraic attacks; case study: Toyocrypt
- Other attacks and conclusions

# Security of a stream cipher

- The standard assumption KNOWN PLAINTEXT ATTACK
- This implies knowledge of the keystream

 $\mathbf{z} = z_1, z_2, \ldots, z_N.$ 

When IV is used the opponent knows 
$$\mathbf{z}_1 = z_{1,1}, z_{1,2}, \ldots, z_{1,N}$$
, for  $IV = 1$   
 $\mathbf{z}_1 = z_{2,1}, z_{2,2}, \ldots, z_{2,N}$  for  $IV = 2$   
:  
generated by the same key k. Could be a *chosen IV attack*.

- KEY RECOVERY ATTACK Recover the secret key k.
- DISTINGUISHING ATTACKS Build a distinguisher that can distinguish the running key  $Z = z_1, z_2, \ldots, z_N$  from random (or  $z_1, z_2, \ldots$  in the IV case)
- OTHER ATTACKS RELATED: Prediction of the next symbol, ... UNRELATED: Side-channel attacks (power analysis, timing attacks, etc.), ...

- Universal distinguishers Apply known statistical tests
- Time-memory tradeoff attacks
   Decrease computational complexity by using memory
- Guess-and-determine Guess unknown things on demand
- Correlation attacks Dependence between output and internal unknown variables
- Linear attacks Apply linear approximations
- Algebraic attacks

View your problem as the solution to a system of nonlinear equations

## Definition of the generator

#### The generator,



key	keystream
0000	0110100110110100
0001	1010111001000010

Version 2 (including IV):



IV	key	
0000	0000	0110100110110100
	0001	1010111001000010
0001	0000	1100101101010101
10	0001	0101001100110100

Exhaustive key search: Search all  $2^k$  different keys and compare the keystream with the received value.

Rough security goal: There should be no attack better than exhaustive key search.

Distinguishing attack: Run a general statistical analysis on the running key

$$Z=z_1,z_2,\ldots,z_N$$

to see if it acts like a random sequence.

You can use any statistical test.

Pearson's chi-square test: A test of goodness of fit establishes whether or not an observed frequency distribution differs from a theoretical distribution.

$$\chi^{2} = \sum_{i=1}^{n} \frac{(O_{i} - E_{i})^{2}}{E_{i}}$$

where  $O_i$  = an observed frequency;  $E_i$  = an expected (theoretical) frequency, asserted by the null hypothesis.  $\chi^2$  is approximately  $\chi^2$ -distributed with n-1 degree of freedom when N is large.

The chi-square distribution for n-1 degree of freedom shows the probability of observing this difference (or a more extreme difference than this).

Frequency test:  $n_0$  is the number of 0's,  $n_1$  is the number of 1's,  $N = n_0 + n_1$ .

$$X = \frac{(n_0 - N/2)^2}{N/2} + \frac{(n_1 - N/2)^2}{N/2} = \frac{(n_0 - n_1)^2}{N}$$

Poker test: Split z into l non-overlapping parts of length m. Let  $n_i$  be the number of sequences of "type i" and length m, for  $i = 1..2^m$ .

$$\chi^2 = \sum_{i=1}^{2^m} \frac{(n_i - l \cdot 2^{-m})^2}{l \cdot 2^{-m}}$$

will have  $2^m - 1$  degrees of freedom.

NIST statistical test suite, DIEHARD, ...

Run any available software.

The problem is that it is unlikely that you will find a statistical weakness in this way...

Filiol, Saarinen

Chosen IV: Ask for the keystreams for various IV values.

In this case: Select some IV bits  $(iv_1, iv_2, \dots iv_t)$ . Keep the remaining IV bits fixed (key is also fixed). Then

$$z_i = F_i(iv_1, iv_2, \dots iv_t),$$

where  $F_i()$  is an unknown Boolean function in t variables.

By running through all IV values we get for example

$$F_1(0,0,\ldots,0) = 0, F_1(0,0,\ldots,1) = 0, F_1(0,\ldots,1,0) = 1,\ldots,$$

i.e., the truth table of  $F_1$ .

We can reconstruct  $F_1$  to, for example, ANF,

$$F_1(iv_1, iv_2, \dots iv_t) = iv_2 + iv_1iv_2 + \dots$$

A chosen IV statistical attacks examines statistical properties of F (by possibly repeating the above several times).

Compute the ANF of  $F_i$ .

Count the number of monomials of degree d in  $F_i$ , and call this M.

The expected number of monomials of degree d in a random Boolean function is  $\frac{1}{2} \binom{t}{d}$ . Check with a  $\chi^2$  test,  $\chi^2 = (2M - N)^2/N$ , where  $N = \binom{t}{d}$ . Other chosen IV tests are possible, e.g., a bit flipping test. (Flip one IV bit and check how often the output bit is flipped)

Saarinen applied chosen IV tests on all 34 proposals in eSTREAM phase 1. The result was that 6-8 ciphers could be distinguished from random.

Chosen IV tests attacks the initialization process of a cipher. Most eSTREAM candidates that were attacked changed their initilization.

Usually, a close study of the design of a cipher makes the detection of a statistical weakness more probable than universal tests.

Example: RC4 i := 0

```
\label{eq:constraint} \begin{array}{l} j := 0 \\ \mbox{while GeneratingOutput:} \\ i := (i+1) \mbox{ mod } 256 \\ j := (j+S[i]) \mbox{ mod } 256 \\ \mbox{swap}(S[i],S[j]) \\ \mbox{output } S[(S[i]+S[j]) \mbox{ mod } 256] \\ \mbox{endwhile} \end{array}
```

$$P(z_2=0) \approx 2/256.$$

"Proof": Let  $S_t$  be the stored permutation at time t.

1. When  $S_0[2] = 0$  (and  $S_0[1] \neq 2$ ) then  $P(z_2 = 0) = 1$ . 2. When  $S_0[2] \neq 0$  then  $P(z_2 = 0) = 1/256$ .

# Time-memory tradeoff attacks

The basic attack:

At time t the generator is in a certain state  $s_t$ . It will produce output  $z_t$  and go to a new state  $s_{t+1}$ , output  $z_{t+1}$ ,... A CYCLE

Assume that this cycle has  $2^s$  different states.

Select  $2^r$  random states. For each state, generate the roughly s bits of keystream (starting in the state). Put (state, keystream) in a table (size  $2^r$ ) sorted according to keystream.

For each s bit segment of the observed keystream, check if it is in the table.

For r = s/2, the table size is  $2^{s/2}$  and the expected length of keystream is about  $2^{s/2}$ .

#### **Conclusion:**

Number of state bits must be at least twice the number of key bits.

(Lund University)

Assume that we can observe many keystreams generated by different keys.

Trivial attack: Select  $2^r$  random keys, generate k bits of keystream, put (key, keystream) in a table sorted according to keystream.

Then ask for many k bits keystreams, encrypted under different keys. The table size is  $2^{k/2}$  and the expected total length of keystreams is about  $2^{k/2}$ .

There are many variations of these TMTO attacks, including IV values, for example by Hong, Sarkar.

# Attacking a specific cipher: Guess-and-determine

Idea: Guess unknowns when you need to be able to determine something else (run through all guesses).

Example: A5/1



$$s_1 + t_1 + u_1 = z_1$$
  
 $s_{d_1} = x, t_{d_2} = x, u_{d_3} = x + 1$   
 $s_2 + t_2 + u_1 = z_2, \dots$ 

For LFSR-based stream ciphers, the Berlekamp-Massey algorithm can be used.

Linear complexity of s,  $L(\mathbf{s}) = \text{Length}$  of the shortest LFSR that can generate s.

For a length N randomly selected sequence s, the linear complexity is almost always around  $N\!/\!2.$ 

BM-algorithm computes the linear complexity in complexity at most  ${\cal O}(N^2).$ 

Let s and s' be two sequences.

If s'' is constructed as  $s''_i = s_i + s'_i$  then  $L(\mathbf{s}'') \le L(\mathbf{s}) + L(\mathbf{s}')$ .

If s'' is constructed as  $s''_i = s_i \cdot s'_i$  then  $L(\mathbf{s}'') \leq L(\mathbf{s}) \cdot L(\mathbf{s}')$ .

## The nonlinear combination generator



- The linear complexity of the keystream sequence z is at most  $S(L_1, \ldots, L_l)$ , evaluated over the integers.
- The Boolean function  ${\cal S}$  should have high degree due to attacks from BM-algorithm.

Assume there is a dependence between one LFSR and the output



All possible LFSR sequences are codeword in a linear code C. Reconstructing the initial state is the problem of decoding the code C on BSC  $(1/2+\epsilon)$ .

**First approach:** Test all possible LFSR sequences (Siegenthaler). This will require keystream length roughly  $N = L/(1 - h(0.5 + \epsilon))$  to find the correct one (ML decoding), where L is the LFSR length. If we have very long keystream we can decode with less complexity.

**Second approach:** If there are low weight parity checks (low weight feedback polynomial), we can use iterative decoding (Meier, Staffelbach).

There are lots of other proposed methods to reconstruct the LFSR.

### Linear approximation attacks - basic ideas

- Replace nonlinear parts by a linear approximation.
- Find an expression including keystream symbols where all unknown variables are eliminated,

$$\sum_{i} d_i z_{n+i} = 0.$$

- Binary case, let  $B_n = \sum_i d_i z_{n+i}$ . Then  $P(B_n = 0) = 1/2 + \epsilon$ .
- Collect as many samples as we need to distinguish the sequence  $B_1, B_2, \ldots$  from random.
- We need roughly  $1/\epsilon^2$  samples.
- Piling-up lemma: Let  $P(X_i = 0) = 1/2 + 1/2\epsilon_i$  and  $X = X_1 \oplus X_2$ ,  $X_1$  and  $X_2$  independent. Then  $P(X = 0) = 1/2 + 1/2\epsilon$ , where

$$\epsilon = \epsilon_1 \epsilon_2.$$

# Case study: Achterbahn



- Nonlinear combination using NLFSR sequences, all with large period.
- 8 NLFSRs. Sizes between 22 and 31 bits.
- Reduced variant takes output of each NLFSR as input to Boolean function.
- Full variant takes a linear combination of some bits in NLFSR as input to Boolean function.

 $S(x_1, \dots, x_8) = x_1 + x_2 + x_3 + x_4 + x_5 x_7 + x_6 x_7 + x_6 x_8 + x_5 x_6 x_7 + x_6 x_7 x_8.$ 

# Description of Achterbahn (original)



- Each NLFSR is clocked similarly to a LFSR, except that the feedback bit is not a linear function, but a polynomial of degree 4. Details of this clocking are not improtant for us.
- NLFSR i is denoted  $R_i$  and has length  $N_i$ .
- Let  $x_i(t)$  be the output of  $R_i$  at time t.
- The period  $T_i$  of the sequence from  $R_i$  is  $T_i = 2^{N_i} 1$ .
- The linear complexity  $L_i$  of the sequence from  $R_i$  is large (close to  $2^{N_i}$ ).

 $S(x_1,\ldots,x_8) = x_1 + x_2 + x_3 + x_4 + x_5x_7 + x_6x_7 + x_6x_8 + x_5x_6x_7 + x_6x_7x_8.$ 

• The keystream bit is computed by  $z(t) = S(x_1(t), \dots, x_8(t))$ . The linear complexity of the keystream sequence z is at most

$$L = S(L_1, \ldots, L_8),$$

- It would be insecure to combine the small nonlinear registers using a linear function. Indeed, in this case, the linear complexity L of Achterbahn would be bounded by  $8 \times 2^{31}$  since 31 is the length of the largest register.
- For Achterbahn, S is not linear, but its algebraic degree is 3. Roughly, the linear complexity of Achterbahn's outputs is :

$$L \le 2^{28} \times 2^{29} \times 2^{31} = 2^{88}.$$

# Linear Cryptanalysis

$$\begin{aligned} z(t) &= S(x_1(t), \dots, x_8(t)) \\ &= x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t) \oplus x_5(t) \oplus x_7(t) \\ &\quad x_6(t)x_7(t) \oplus x_6(t)x_8(t) \oplus x_5(t)x_6(t)x_7(t) \oplus x_6(t)x_7(t)x_8(t). \end{aligned}$$

Introduce the notation  $l(t) = x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t)$ . We have linear approximations,

$z(t) = l(t) \oplus x_5(t)$	with probability	10/16,
$z(t) = l(t) \oplus x_6(t)$	with probability	12/16,
$z(t) = l(t) \oplus x_7(t)$	with probability	12/16,
$z(t) = l(t) \oplus x_8(t)$	with probability	10/16.

In particular, we focus on the second approximation,

$$z(t) = l(t) \oplus x_6(t), \tag{1}$$

with probability  $\frac{12}{16} = 0.75 = 0.5 \ (1 + 0.5)$ . Therefore the **bias** is  $\varepsilon = 0.5$ .

Let

$$ll(t) = l(t) \oplus l(t+T_1).$$

This expression does not contain any term in  $x_1$ . Similarly, define

$$lll(t) = ll(t) \oplus ll(t+T_2),$$
  
$$llll(t) = lll(t) \oplus lll(t+T_3).$$

Here llll(t) contains no term in  $x_2$  or  $x_3$ , so it is a combination of bits coming from the register  $R_4$  only. Thus it satisfies

$$llll(t) = llll(t + T_4).$$

In other terms, we have the following relation on the bits l(i),

$$\begin{array}{rcl} 0 &=& l(t)+l(t+T_1)+l(t+T_2)+l(t+T_3)+l(t+T_4) \\ &+& l(t+T_1+T_2)+l(t+T_1+T_3)+l(t+T_1+T_4) \\ &+& l(t+T_2+T_3)+l(t+T_2+T_4)+l(t+T_3+T_4) \\ &+& l(t+T_1+T_2+T_3)+l(t+T_1+T_2+T_4)+l(t+T_1+T_3+T_4) \\ &+& l(t+T_2+T_3+T_4)+l(t+T_1+T_2+T_3+T_4). \end{array}$$

- $l(t) \oplus l(t+T_i)$  does not depend on the variable  $x_i$ .
- The sequence generated by  $R_i$  has characteristic polynomial  $x^{T_i} 1$ . Hence, we have  $x_i(t + T_i) \oplus x_i(t) = 0$ .

Example:

Sequence produced by function  $F(t) = x_1(t) + x_2(t)$  has characteristic polynomial

$$g(x) = (x^{T_1} - 1)(x^{T_2} - 1)$$

giving a parity check equation involving 4 terms.

$$F(t) \oplus F(t+T_1) \oplus F(t+T_2) \oplus F(t+T_1T_2) = 0$$

# Combining parity checks and approximations

Use the approximation  $z(t) = x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t) \oplus x_6(t)$ . Creating parity checks as above will involve 32 keystream bits (and 32 approximations) distant at most

$$T_{max} = T_1 + T_2 + T_3 + T_4 + T_6 = 381681659 \simeq 2^{28.51}$$

positions.

But  $l(t) \oplus x_6(t)$  is only an approximation of the output function. However we sum up 32 times the linear approximation over different values of t, which has the effect of multiplying the biases.

The parity check is satisfied by the sequence z(t) with probability

0.5 
$$(1 + \varepsilon^{32}) = 0.5 \left(1 + \frac{1/21}{2^{32}}\right).$$

Therefore if we consider a sequence of  $2^{64}$  output bits and evaluate all the parity checks, we will detect this bias.

(Lund University)

A natural extension - guess the initial content of register  $R_1$ . Then, we can eliminate the term  $y_1(t)$  in the previous linear approximation. Consequently, the weight of the parity check drops from 32 to 16, bringing the bias from  $2^{-32}$  to  $2^{-16}$ .

For the correct guess of  $R_1$ , we detect a bias by looking at  $2^{32}$  keystream bits, while there is usually no bias for incorrect guesses.

This attack costs about  $2^{55}$  computational steps and requires  $2^{32}$  keystream bits. For the full Achterbahn, the number of guesses for  $R_1$  is  $2^{29}$  instead of  $2^{23}$  increasing the complexity of the key recovery from  $2^{55}$  to  $2^{61}$ .

### • Approximation:

 $z(t) = x_1(t) \oplus x_2(t) \oplus x_3(t) \oplus x_4(t) \oplus x_6(t)$ 

with bias  $\epsilon=0.5$ 

• The parity check equation is:

$$(x^{T_1} - 1)(x^{T_2} - 1)(x^{T_3} - 1)(x^{T_4} - 1)(x^{T_6} - 1) = 0$$

and it has 32 terms  $\Rightarrow$  total bias  $\epsilon = 2^{-32}$ . A distinguishing attack requiring  $2^{64}$  samples exists.

• Improvement: Guess  $R_1 \Rightarrow$  parity check has only 16 terms so  $2^{32}$  samples are required by the distinguisher. Need to add a factor of  $2^{22}$  giving a key recovery attack with computational complexity  $2^{54}$  and using  $2^{32}$  keystream bits.

# Description of Achterbahn (version 2)



- 10 NLFSRs instead of 8. Sizes between 19 and 32 bits.
- Still has a reduced and a full variant.

# Description of Achterbahn (version 2)

 $S(x_1, \dots, x_{10}) = x_1 + x_2 + x_3 + x_9 + G(x_4, x_5, x_6, x_7, x_{10}) + (x_8 + x_9)(G(x_4, x_5, x_6, x_7, x_{10}) + H(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{10})),$ 

#### where

$$G(x_4, x_5, x_6, x_7, x_{10}) = x_4(x_5 \lor x_{10}) + x_5(x_6 \lor x_1/27) + x_6(x_4 \lor x_{10}) + x_7(x_4 \lor x_6) + x_{10}(x_5 \lor x_7)$$

and

$$H(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{10}) = x_2 + x_5 + x_7 + x_{10} + (x_3 + x_4)\overline{x}_6 + (x_1 + x_2)(x_3\overline{x}_6 + x_6(x_4 + x_5)).$$

#### Resiliency of the function is 5.

• Length of register  $R_i$  is denoted  $N_i$ .

$N_1$	19	$N_6$	27
$N_2$	22	$N_7$	28
$N_3$	23	$N_8$	29
$N_4$	25	$N_9$	31
$N_5$	26	N <sub>10</sub>	32

- Period of register  $R_i$  is denoted  $T_i$ , hence  $T_i = 2^{N_i} 1$ .
- Bias  $\epsilon$  of an approximation A of S is given as  $\epsilon = 2Pr(S = A) 1$ . Samples needed to distinguish sequence generated by S, using A is given as

# samples needed = 
$$\frac{1}{\epsilon^2}$$

# Achterbahn version 2 and Nonlinear approximations

- Nonlinear approximations can be used as well as linear ones.
- The cubic approximation

$$C(x_1,\ldots,x_{10}) = x_4 + x_6 x_9 + x_1 x_2 x_3.$$

with bias  $2^{-6}$ .

• Guess the state of  $R_4$  and use the characteristic polynomial

$$g(x) = (x^{T_6 T_9} - 1)(x^{T_1 T_2 T_3} - 1)$$

- Total bias is  $\epsilon = 2^{-24}$  so  $2^{48}$  samples are needed. Computational complexity is  $2^{48}2^{N_4} = 2^{73}$ .
- Distance between first and last bit in parity check is  $T_1T_2T_3 + T_6T_9 \approx 2^{64}$  bits.
- Solution: Restrict keystream length to  $2^{63}$  bits.

• We use the quadratic approximation

$$Q(x_1,\ldots,x_{10}) = x_1 + x_2 + x_3 x_8 + x_4 x_6$$

with bias  $2^{-5}$ .

 $\bullet$  Denote keystream sequence by z(t) and sequence produced by Q by  $z^\prime(t).$ 

• We use the quadratic approximation

$$Q(x_1,\ldots,x_{10}) = x_1 + x_2 + x_3 x_8 + x_4 x_6$$

with bias  $2^{-5}$ .

- Denote keystream sequence by  $\boldsymbol{z}(t)$  and sequence produced by  $\boldsymbol{Q}$  by  $\boldsymbol{z}'(t).$
- Use characteristic polynomial

$$g(x) = (x^{T_3 T_8} - 1)(x^{T_4 T_6} - 1)$$

which gives a parity check equation with 4 terms:

 $d(t) = z(t) \oplus z(t + T_3T_8) \oplus z(t + T_4T_6) \oplus z(t + T_3T_8 + T_4T_6)$ 

 $\bullet$  With probability  $\alpha = \frac{1}{2}(1+2^{-20})$  we have

$$d(t) \stackrel{\alpha}{=} z'(t) \oplus z'(t+T_3T_8) \oplus z'(t+T_4T_6) \oplus z'(t+T_3T_8+T_4T_6) = x_1^t \oplus x_2^t \oplus x_1^{t+T_3T_8} \oplus x_2^{t+T_3T_8} \oplus x_1^{t+T_4T_6} \oplus x_2^{t+T_4T_6} \oplus x_1^{t+T_3T_8+T_4T_6} \oplus x_2^{t+T_3T_8+T_4T_6}.$$

- Amount of samples needed is  $2^{40}$ .
- With  $N_1 = 19$  and  $N_2 = 22$  the computational complexity is  $2^{19+22+40} = 2^{81}$ .
- Distance between first and last bit in parity check is  $T_3T_8 + T_4T_6 \approx 2^{53}$ .

• We note that

$$R_1(t) = R_1(t+T_1) = R_1(t+2^{19}-1).$$

so for all keystream bits distance  $T_1$  apart,  $x_1$  will always contribute with the same value.

Take the sequence

$$\begin{aligned} d'(t) &= z(tT_1) \oplus z(tT_1 + T_3T_8) \oplus z(tT_1 + T_4T_6) \oplus z(tT_1 + T_3T_8 + T_4T_6) \\ & \stackrel{\alpha}{=} x_2^{tT_1} \oplus x_2^{tT_1 + T_3T_8} \oplus x_2^{tT_1 + T_4T_6} \oplus x_2^{tT_1 + T_3T_8 + T_4T_6} \oplus \gamma(t), \end{aligned}$$

where

$$\gamma(t) = x_1^{tT_1} \oplus x_1^{tT_1 + T_3T_8} \oplus x_1^{tT_1 + T_4T_6} \oplus x_1^{tT_1 + T_3T_8 + T_4T_6}$$

is a constant (0 or 1).

• Amount of keystream needed:



Computational complexity:



• Computational complexity, full variant: 2<sup>65</sup>

# Improving the computational complexity

Assumption: The attacker observes  $2^{59.02}$  keystream bits. step 1: Produce d'(t).

 $d'(t) = z(tT_1) \oplus z(tT_1 + T_3T_8) \oplus z(tT_1 + T_4T_6) \oplus z(tT_1 + T_3T_8 + T_4T_6)$ 

and save the sequence in a 2<sup>40</sup> bit memory. Computational complexity: **??** 

step 2:

Straightforward approach: Compare d'(t) with

$$x_2^{tT_1} \oplus x_2^{tT_1+T_3T_8} \oplus x_2^{tT_1+T_4T_6} \oplus x_2^{tT_1+T_3T_8+T_4T_6}$$

for  $0 \le t < 2^{40}$  and all initial states of  $R_2$ . But  $T_2 = 2^{22} - 1 \ll 2^{40}$  so  $d'(t + iT_2)$ ,  $\forall i$ , will be compared with the same value.

**Improvement**: Build a table with values in d'(t).

step 2: Build a table with values in d'(t).

Position in d'(t)	# Zeros	# Ones
$0+iT_2$		
$1+iT_2$		
$2+iT_2$		
:		
$T_2+iT_2$		

Computational complexity:  $2^{40}$ . Memory needed:  $2^{22}$  words. step 3: Recover  $R_2$ . For each initial state of  $R_2$  the sum of the four bits

$$x_2^{tT_1} \oplus x_2^{tT_1+T_3T_8} \oplus x_2^{tT_1+T_4T_6} \oplus x_2^{tT_1+T_3T_8+T_4T_6}$$

 $0 \le t < T_2$ , is found. All positions can be taken modulo  $T_2$ . Add the number in the stored table depending on if it is 0 or 1. The bias will be detected for the initial state. Computational complexity:  $2^{44}$  ( $2^{47}$  for full variant).

(If bias is detected for more states, then we can do the same thing, shifting our sequence one bit.)

- Most expensive operation is to go through all states  $(2^{44} \text{ or } 2^{47})$ .
- However, we still need to use keystream bits  $2^{59.02}$  bits apart when we create the sequence d'(t). But we use only  $2^{40}$  bits.
- Conservative claim: Computational complexity is  $2^{59.02}$  (on both reduced and full Achterbahn).

Describe the relation between known keystream bits and key bits or state bits as nonlinear equations,

$$f(z_1, z_2, \ldots, k_0, k_1, \ldots, k_n) = 0.$$

Try to solve the system of nonlinear equations.

Particulars for stream ciphers: If the generator has a linear update, algebraic attacks are particularly strong.

$$z_1 = f(k_0, k_1, \dots, k_n),$$
  

$$z_2 = f(L(k_0, k_1, \dots, k_n)),$$
  

$$z_3 = f(L^2(k_0, k_1, \dots, k_n)), \dots$$

If we find a low degree relation,

$$z_1=f(k_0,k_1,\ldots,k_n),$$

where for example  $\deg(f) = d$ , all equations

$$z_i = f(L^{i-1}(k_0, k_1, \dots, k_n)),$$

will have the same degree.

Relinearization: If we collect  $\binom{n}{d}$  such equations we can solve the system by relinearization. We replace every monomial (degree  $\leq d$ ) by a new variable, getting a linear system.

If we do not have linear update, the situation is close to the case of algebraic attacks on block ciphers.

We may try to get low degree and/or overdefined systems of equations.

We may try to solve them through XL, XLS, Gröbner basis techniques, ...

Toyocrypt is a stream cipher proposal that entered the second evaluation phase of the Japanese Cryptrec call for primitives, later rejected.

Toyocrypt is a filter generator with filtering function

$$f(s_0, ..., s_{127}) =$$

$$s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10} s_{23} s_{32} s_{42}$$

 $+s_1s_2s_9s_{12}s_{18}s_{20}s_{23}s_{25}s_{26}s_{28}s_{33}s_{38}s_{41}s_{42}s_{51}s_{53}s_{59} + \prod_{i=0}^{62} s_i.$ 

We have relations of the form

$$z_1 = f(s_0, .., s_{127})$$
  

$$z_2 = f(L(s_0, .., s_{127})),$$
  

$$z_3 = f(L^2(s_0, s_1, ..., s_{127})), ...$$

But f has degree 63...

Use low weight multiples: Multiply  $z = f(\mathbf{x})$  by a new polynomial  $g(\mathbf{x})$ ,

$$z \cdot g(\mathbf{x}) = f(\mathbf{x})g(\mathbf{x}),$$

such that  $f(\mathbf{x})g(\mathbf{x})$  has low degree.

Look for a low degree common divisor  $g^\prime$  to high degree monomials in f and multiply with  $(g^\prime-1).$ 

For  $f(s_0, ..., s_{127})$ , use  $g(x) = (s_{23} - 1)$ . Then  $\deg(f(\mathbf{x})g(\mathbf{x})) = 3$ .

We get one new degree 3 equation for each keystream bit. Using relinearization we need  $T = \binom{128}{3}$  bits and complexity  $T^3$  with Gaussian elimination or slightly lower complexity with other methods.

Traditionally not as essential as in block cipher cryptanalysis.

BUT, in chosen IV attacks differential attacks are applicable.

Many recent stream cipher proposals are close to block ciphers, e.g., eSTREAM candidates Salsa20, LEX. Tools from block cipher cryptanalysis will be applicable here.

In a side-channel attack we attack an *implementation* of a stream cipher rather than the algorithm itself.

The attack uses a side-channel, for example measuring the power consumption of the implementation.

The key question: How expensive is it to implement an algorithm in a presumably secure way when side-channels exist?

Not too much work has been done on side-channel attacks on stream ciphers.

- We have reviewed basic ideas of many different approaches to cryptanalysis of stream ciphers.
- We have seen a few case studies.