Improving Key Recovery to 784 and 799 rounds of Trivium using Optimized Cube Attacks

Pierre-Alain Fouque [1] Thomas Vannet [2]

[1]Université de Rennes 1

[2]NTT Secure Platform Laboratories

March 13, 2013

Table of contents

Introduction
- Trivium
- Cube Attacks
  - Polynomial testing
  - Polynomial interpolation

Exploiting polynomials of degree 2
- Testing the degree
- Heuristically interpolating
- Solving the system ?

The Moebius Transform

Exploiting the cipher structure

Conclusion

Outline

Introduction
- Trivium
- Cube Attacks

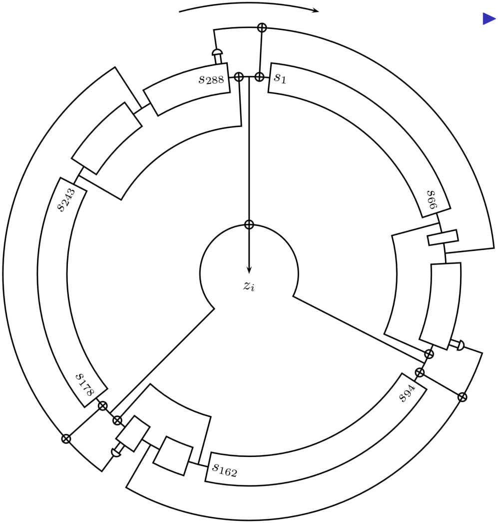
Exploiting polynomials of degree 2

The Moebius Transform

Exploiting the cipher structure

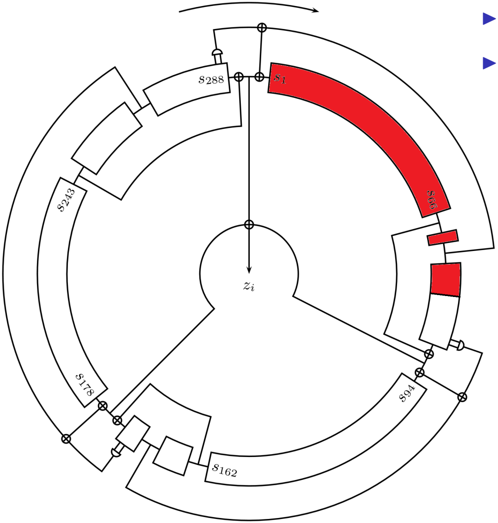
Conclusion

Trivium



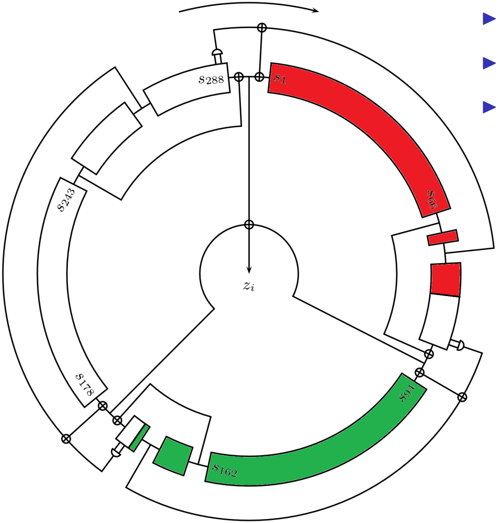
- Stream cipher on 3 NLSFR

Trivium



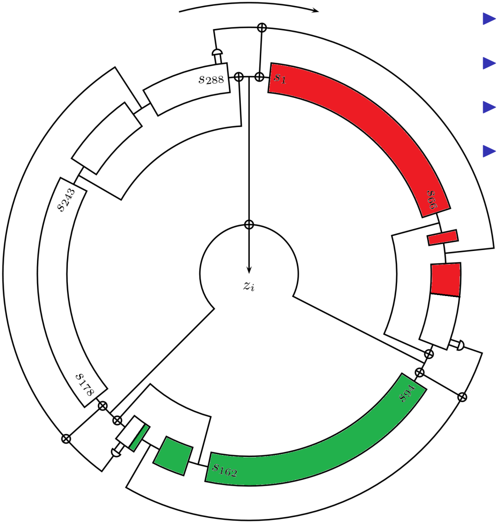
- Stream cipher on 3 NLSFR
- 80-bit key $x_1, \ldots, x_{80}$

Trivium



- Stream cipher on 3 NLSFR
- 80-bit key $x_1, \ldots, x_{80}$
- 80-bit IV $v_1, \ldots, v_{80}$

Trivium



- Stream cipher on 3 NLSFR
- 80-bit key $x_1, \ldots, x_{80}$
- 80-bit IV $v_1, \ldots, v_{80}$
- 1152 initialization rounds

Trivium (feedback function)

Algorithm 1 Updates Trivium's internal state $s_1, \ldots, s_{288}$

$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$z_i \leftarrow t_1 + t_2 + t_3$
$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
$(s_{178}, s_{279}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

Known Attacks

- Full key recovery on 735 rounds in 2^{30} queries [DinSha09]
- 35 key bits recovered after 767 rounds in about 2^{36} queries [DinSha09]
- Distinguisher up to 806 rounds [KneMeiNay10]

Contributions

- Full key recovery on 784 rounds in 2^{39} queries
- 12 key bits and 6 quadratic expressions recovered after 799 rounds in about 2^{39} queries, leading to key recovery in 2^{62} queries

Cube Attacks

- Introduced by Dinur and Shamir at EUROCRYPT 2009
- We consider the polynomial representation of a cipher
- Offline phase : Extract low-degree expressions in key bits
- Online phase : Evaluate the expressions and solve a system to recover the key

Cube Attacks

- Cube $C = \{v_{c_1}, \ldots, v_{c_k}\}$ of size k
- $P(x_1, \ldots, x_n, v_1, \ldots, v_p) \in \mathbb{F}_2[x_1, \ldots, x_n, v_1, \ldots, v_p]$
- $P = v_{c_1} \ldots v_{c_k} P_C + P_R$
- $\displaystyle\sum_C P = P_C.$
- P_C is a black box polynomial that can be queried
- Complexity of a query : 2^k
- We need to test whether P_C has a low degree and interpolate it if it is the case
- The cube is chosen by a random walk depending on the degree of P_C

BLR Test

Algorithm 2 Tests linearity of a polynomial

P a black box polynomial
repeat
  X_1, X_2 two random inputs in $\mathbb{F}_2^k$
  if $P(X_1 + X_2) + P(X_1) + P(X_2) \neq P(0)$ **then**
   return false
  end if
until r tests have been carried out
return True

BLR Test

- The algorithm requires 3 queries for every linearity test
- Similarly, it would require 7 queries for a test of degree 2 : Replace the test in BLR with $P(X_1+X_2+X_3)+P(X_1+X_2)+P(X_1+X_3)+P(X_2+X_3)+P(X_1)+P(X_2)+P(X_3) \neq P(0)$

Interpolating

Algorithm 3 Interpolates a linear polynomial

P a black box linear polynomial

$p_0 \leftarrow P(0)$

for $i = 1$ to 80 **do**

  $p_i \leftarrow P(x_1 \leftarrow 0, \ldots, x_i \leftarrow 1, \ldots, x_{80} \leftarrow 0) + p_0$

end for

return $x_0 + \sum_{i=1}^{80} p_i x_i$

Interpolating

- Complexity : 81 queries for a black box polynomial of degree 1
- For degree k, $\sum_{i=0}^{k} \binom{80}{i}$ queries are necessary since each query returns a binary information

Shortcomings and solutions

- The original attack limits itself to linear polynomials while degree 2 polynomials can be just as useful and easier to find
- The suggested random walk is not efficient, we suggest a different approach testing many parameters at once
- The cube attack does not exploit the structure of the cipher, we study it to find low-density subpolynomials

Outline

Introduction

Exploiting polynomials of degree 2
- Testing the degree
- Heuristically interpolating
- Solving the system ?

The Moebius Transform

Exploiting the cipher structure

Conclusion

Weakened BLR Test

- The original BLR algorithm assumes the inputs are independently chosen at random
- In practice, reusing previous inputs proves to be efficient
- Pick 10 random inputs $X_1, \ldots, X_{10}$
- Test linearity for every couple (X_i, X_j) (45 total)
- 45 linearity tests are performed in 55 queries, against 135 with the true BLR test

Weakened BLR Test for degree 2

- Pick 10 random inputs $X_1, \ldots, X_{10}$
- Test linearity for every couple (X_i, X_j) (45 total)
- For every i_1, i_2, i_3, test if $P(X_{i_1} + X_{i_2} + X_{i_3}) + P(X_{i_1} + X_{i_2}) + P(X_{i_1} + X_{i_3}) + P(X_{i_2} + X_{i_3}) + P(X_{i_1}) + P(X_{i_2}) + P(X_{i_3}) \neq P(0)$
- After the linearity test, only $P(X_{i_1} + X_{i_2} + X_{i_3})$ is unknown
- To sum up, we perform 45 linearity tests and 45 degree 2 tests in 100 queries (450 queries required if independent inputs are used)

Interpolating (heuristic)

- We need to restrict the space potentially covered by the degree 2 polynomials
- First rounds of Trivium : $x_i + x_{i+25} \cdot x_{i+26} + x_{i+27}$
- We performed a formal interpolation on cubes of size 30 after 784 rounds
- Assume this form and check that it is correct
- The interpolation was successful over 95% of the time with only 81 queries

Solving the system ?

- Solving a linear system requires few equations, but a system of degree 2 may require a lot more
- All obtained polynomials have the form $x_i + x_{i+25} \cdot x_{i+26} + x_{i+27}$
- With cubes of size 35, bruteforcing 40 key bits does not increase the complexity
- In this configuration, for every 2 bruteforced bits, a linear relation is obtained
- In most cases, polynomials of degree 2 cost no more than linear polynomials to obtain and bring as much information

Outline

Introduction

Exploiting polynomials of degree 2

The Moebius Transform

Exploiting the cipher structure

Conclusion

Moebius Transform

- $P = \sum_{\sigma \in \{0,1\}^n} \alpha_\sigma X^\sigma$ with σ, $\alpha_\sigma \in \mathbb{F}_2$
- $P^m : \begin{array}{rcl} \{0,1\}^n & \rightarrow & \mathbb{F}_2 \\ \sigma & \rightarrow & \alpha_\sigma \end{array}$
- Basically, it is a an efficient tool for interpolating high degree polynomials
- Time complexity : $n \cdot 2^n$
- Memory complexity : 2^n

Moebius Transform (application)

- Cube $C = \{v_{c_1}, \ldots, v_{c_k}\}$ of size k
- $Q(v_{c_1}, \ldots, v_{c_k})$ is a restriction of $P(x_1, \ldots, x_n, v_1, \ldots, v_p)$
- $D \subset C$ and for $i \in \{1, \ldots, k\}$ $d_i = 1 \iff v_{c_i} \in D$
- $Q^m(d_1, \ldots, d_k)$ is the associated value of P_D
- In a cube of size 40, over 3.8 millions of cubes of size 34
- The only freedom resides in the choosing of the cube

Outline

Introduction

Exploiting polynomials of degree 2

The Moebius Transform

Exploiting the cipher structure

Conclusion

The density problem

- Measurements done with the Moebius Transform

Observed polynomial density after 799 rounds

Monomial size	Density (random cube)	Density (chosen cube)
33	49.89%	38.44%
34	49.55%	28.36%
35	48.25%	16.82%
36	44.19%	7.31%
37	34.07%	1.84%
38	16.47%	0.15%
39	3.66%	0%

Exploiting the cipher structure

- Output of Trivium is a sum of 6 registers
  $s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$
- Each of those is a product of 2 registers around 96 rounds before added to some terms of degree one
- We assume those terms have a degree lower than the cube size and neglect them
- $P = \sum_{i=1}^{6} P_{i,1} P_{i,2} = v_{c_1} \dots v_{c_k} P_C + P_R$

Exploiting the cipher structure

- $P = \sum_{i=1}^{6} P_{i,1} P_{i,2} = v_{c_1} \ldots v_{c_k} P_C + P_R$
- We assume that for every partition $\{C_1, C_2\}$ of the cube, C_k yields a low-degree polynomial on $P_{i,j}$
- Find two disjoint cubes producing the 0 polynomial on those 12 registers
- Hopefully, the union of those cubes will produce a low-degree expression

Exploiting the cipher structure (improvement)

- C_1 and C_2 of size k
- Every subcube of size at least $k-3$ has an associated $P_C = 0$ on the 12 registers
- Realize a Moebius Transform on $C_1 \cup C_2$
- Result : After 799 rounds, the density is greatly reduced and we find maxterms for the first time

Outline

Introduction

Exploiting polynomials of degree 2

The Moebius Transform

Exploiting the cipher structure

Conclusion

Conclusion

- We addressed 3 major issues from the standard attack
- Key bits recovered in practical time up to 799 rounds
- While it may go a bit further, density issues suggest the full cipher is still secure