AIDA Breaks BIVIUM (A&B) in 1 Minute Dual Core CPU Time

Michael Vielhaber

Hochschule Bremerhaven, FB2
An der Karlstadt 8, D–27568 Bremerhaven, Germany
and
Universidad Austral de Chile, Instituto de Matemáticas
Casilla 567, Valdivia, Chile

vielhaber@gmail.com

Abstract

The stream cipher BIVIUM (both BIVIUM-A and BIVIUM-B), a modification of the eSTREAM finalist TRIVIUM, can be broken completely by the Algebraic IV Differential Attack, AIDA, using $2^{27.5}$ simulations or one minute of dual core processing.

AIDA uses the subspaces of two 32-dimensional vector spaces over subsets of IV bits to recover 56 of the 80 key bits. The remaining 24 key bits are most easily determined by brute force search.

We applied the Fast Reed-Muller Transform to speed up the search for linear equations in the key bits and the Wavefront Model to rule out nonlinear relations in the key bits early on.

Keywords: AIDA, algebraic IV differential attack, algebraic normal form, fast multiplication, Reed-Muller transform, inclusion exclusion principle.

1 Introduction

The Algebraic IV Differential Attack, AIDA, was introduced by the present author in 2007 [15]. The same attack was republished by Dinur and Shamir in 2008, renaming AIDA as "cube attack" [4][17].

AIDA consists in finding and then exploiting low-complexity parts (*e.g.* linear in the key bits) of the Algebraic Normal Form, ANF, of a given cryptographic function. Applying the Inclusion-Exclusion-Principle or the Reed-Muller transform, this ANF part is represented as a sum over many entries of the Disjunctive Normal Form, DNF. These DNF entries in turn are just truth table values or simulation results of the given Boolean cryptographic function.

TRIVIUM is one of the four final survivors of the eSTREAM Hardware Portfolio. TRIVIUM (with full setup length) is at this moment secure against all attacks. It consists in a 288 stage nonlinear feedback shift register (NLFSR) with 3 nonlinear 8quadratic) feedback functions at roughly evenly spaced positions.

BIVIUM has been suggested by Raddum [11] as a smaller TRIVIUM sibling, using 177 stages and only 2 feedback positions. Crucially, the setup length is also reduced, from $4 \cdot 288 = 1152$ for TRIVIUM to $4 \cdot 177 = 708$ for BIVIUM. The two versions BIVIUM-A and BIVIUM-B differ only by the output function, which is linear in the stage bits for both cases.

ONE.FIVIUM [15] was introduced as a version of TRIVIUM with reduced setup length of 576 steps, not otherwise changing the 288-stage design.

Previous attacks on TRIVIUM, BIVIUM, and ONE.FIVIUM:

The ONE.FIVIUM paper [15] introduced the Algebraic IV Differential Attack AIDA, breaking ONE.FIVIUM by giving 53 key bits in $53 \cdot 2^6 < 2^{12}$ simulations. Sun et al. [12] give a corollary to AIDA againt ONE.FIVIUM. AIDA was thus initially able to break TRIVIUM with a reduced setup length of 576. Later results by Dinur and Shamir obtained key bits for a setup length of 774 [4]. The current record is set by the present author in [18], showing that even a setup length of 792, or (11/16)th of the full 1152 time steps, is insecure, revealing at least 4 key bits linearly. The same paper however gives strong indications that the search for linear equations will be unsuccessful for TRIVIUM with full setup length.

BIVIUM-A [11], has been successfully broken. BIVIUM-B, on the other hand, has only been attacked and broken theoretically, with effort below the brute force 2^{80} threshold, but nowhere near the one minute, or $2^{27.5}$ simulations sufficient for AIDA.

BIVIUM was attacked by McDonald, Charnes, and Pieprzyk [9] via SAT solvers, requiring 21 seconds for BIVIUM-A, but $2^{42.5}$ seconds for BIVIUM-B. Maximov and Biryukov [8] break BIVIUM with $c \cdot 2^{36}$ effort, with c around 1000.

The idea of summing over a hypercube of simulation results appears already in Lai's paper [7] on higher differentials. Lai however, sums over key bits, to eliminate any effect of these bits on the result and hence to diminish the overall degree. AIDA on the contrary sums over IV bits in order to switch from ANF to DNF, quite a different approach. AIDA and Lai's attack seem to be only superficially related.

The general idea of using algebraic methods is of course well-known. Most important, Sean O'Neil developed "Algebraic Structure Defectoscopy" [10], a means to sort out cryptographically weak functions by their failure to produce all the small degree parts in the ANFs.

Another typical tool are Gröbner bases, in particular the algorithm F_4, see Faugère [5]. However, it seems that until now no significant result against TRIVIUM and ciphers of similar complexity can be obtained, since the size of the required Gröbner base explodes.

Statistical attacks were described by Biham and Dunkelman (Differential Cryptanalysis [1]). Their result is that the best differential encountered for TRIVIUM has probability below 2^{-79} already after 378 (out of 1152) setup steps, and therefore the attack does not succeed.

Another statistical attack, by Fischer, Khazaei, and Meier [6], obtains weakly biased approximations of key bits for TRIVIUM with 672 setup steps. Similarly, Kara and Turan [14] obtain an approximation of bias 2^{-31} after 288 setup steps.

2 The Algebraic IV Differential Attack AIDA

For details on AIDA, seealso [15] and [19]. Essentially, any Boolean functions can be written in Disjunctive or Algebraic Normal Form, DNF or ANF, respectively:

$$\begin{aligned} f(x_1,..,x_n) &= \bigvee_{H\subset\{1,...,n\}} d_H \cdot \left(\bigwedge_{h\in H} x_h \wedge \bigwedge_{h\notin H} \overline{x}_h\right) \qquad (DNF) \\ f(x_1,..,x_n) &= \bigoplus_{H\subset\{1,...,n\}} a_H \cdot \bigwedge_{h\in H} x_h \qquad (ANF) \end{aligned}$$

where $\vee, \wedge, \oplus$ are the logical *or*, *and*, and *exclusive or*, respectively, $\overline{x}_h = 1 - x_h$ is the negated variable, and $d_H, a_H \in \mathbb{F}_2$ exclude (if 0) or include (if 1) the respective term (d, a as in DNF, ANF).

Abbreviating $x_H^\wedge := \bigwedge_{h\in H} x_h$, $x_H^\vee := \bigvee_{h\in H} x_h$, and with $\overline{H} := \{1,\ldots,n\}\backslash H$, we may write the normal forms as

$$f(x_1,\ldots,x_n) = \bigvee_H d_H x_H^\wedge \overline{x}_{\overline{H}}^\wedge = \bigoplus_H a_H x_H^\wedge$$

Assume n_1 IV bits, n_2 key bits, hence $n = n_1 + n_2$. In the case of Trivium/Bivium, $n_1 = n_2 = 80, n = 160$. In terms of index sets H vs. I, J, we have $I = H \cap \{1,\ldots,n_1\}$ and $n_1 + J = \{n_1 + j | j \in J\} = H \cap \{n_1+1,\ldots,n_1+n_2\}$.

Now, the ANF can be split into terms according to the IV bits as follows:

$$f(v_1,\ldots,v_{n_1},k_1,\ldots,k_{n_2}) = \bigoplus_{I\subset\{1,...,n_1\}} \left(v_I^\wedge \wedge \big(\bigoplus_{J\subset\{1,...,n_2\}} a_{I,J} k_J^\wedge \big) \right),$$

since $\wedge$ distributes over $\oplus$ (by the respective law in the finite field $\mathbb{F}_2$). For the coefficients, we have $a_{I,J} = a_H \in \mathbb{F}_2$ with the correspondence from H to I, J as before.

We call each of the 2^{n_1} terms $v_I^\wedge \wedge \big(\oplus_{J\subset\{1,...,n_2\}} a_{I,J} k_J^\wedge \big)$ an *ANF part*.

In the case of a fixed (unknown) key $\underline{k} = (k_1,\ldots,k_{n_2}) \in \mathbb{F}_2^{n_2}$, we put $J' = \{1 \le j \le n_2 \mid k_j = 1\}$ and rewrite the ANF as

$$f(v_1,\ldots,v_{n_1},\underline{k}) = \bigoplus_{I\subset\{1,...,n_1\}} a_I(\underline{k}) v_I^\wedge,$$

where

$$a_I(\underline{k}) = \oplus_{J\subset J'} a_{I,J} \in \mathbb{F}_2$$

is the sum of the ANF coefficients for those J with $J \subset J'$, hence $k_J^\wedge = 1$ for the given $\underline{k}$. All other $k_J^\wedge, J \not\subset J'$, evaluate to zero and do not appear.

We show in [19] that by the Inclusion-Exclusion-Principle or the Reed-Muller Transform, we have

$$a_I(\underline{k}) = \bigoplus_{M\subset I} d_{M,J'}.$$

Therefore $2^{|I|}$ queries of the full function $f : \{0,1\}^{n_1+n_2} \to \{0,1\}$ at the DNF or truth table positions with $M \subset I$ (hence $\overline{M} \supset \overline{I}$) are sufficient to calculate $a_I(\underline{k}) \in \mathbb{F}_2$,

where $|I|$ is the size of $I \subset \{1, \ldots, n_1\}$ – without having to know either the ANF (of size up to 2^{160} in case of TRIVIUM), nor $\underline{k}$.

This is the main point of AIDA, see [15, Prop. 3]. The $2^{|I|}$ IV values form an $|I|$–dimensional subvectorspace or hypercube of $\mathbb{F}_2^{n_1} \ni (v_1, \ldots, v_{n_1})$, hence the "other" name of the attack.

The particularly interesting case now occurs for an ANF part being just linear in the key bits,

$$v_I^\wedge \wedge \bigoplus_{J \subset \{1,\ldots,n_2\}} a_{I,J} k_J^\wedge = v_I^\wedge \wedge (a_{I,\emptyset} \cdot 1 \oplus \bigoplus_{j \in \{1,\ldots,n_2\}} a_{I,\{j\}} \cdot k_j),$$

that is when $a_{I,J} = 0$ for $|J| \geq 2$, since $a_I(\underline{k})$ is then

$$a_I(\underline{k}) = a_{I,\emptyset} \oplus \bigoplus_{j \in J'} a_{I,\{j\}}.$$

In this case, we immediately obtain sums of key bits, *i.e.* we will be able to shrink the key space by a factor of two with each and every such equation (while linearly independent).

$$a_H = \bigoplus_{M \subset H} d_M.$$

This is AIDA !

3 Speeding up AIDA

3.1 Phase A: Finding linear relations

This phase still consists mainly in trial and error until success. We started with a 32-dimensional hypercube, which is related to the successful results against (modified) TRIVIUM in [18]. and another hypercube using 32 IV bits not present in the first one. We then applied the fast Reed-Muller transform [13] to speed up the simultaneous evaluation of all subhypercubes. For details consult [19]. Our successful hypercubes mainly have dimension 22. Hence, the actually used 32-dimensional vecor space includes $\binom{32}{22} \approx 2^2$ such hypercubes, which we search simultaneously, while requiring only a factor $2^{32-22} = 2^{10}$ of additional effort. The overall gain is thus a factor of $2^{26}/2^{10} = 65536$.

3.2 Sieving: The Wavefront Model

Efficient sieving for linear relations makes use of the Wavefront Model introduced in [19]. Essentially, we assume that linear relations consist of at most 2 or 3 key bits, plus an optional constant '1'. This is in accordance with the previous results in [15] [4] [18]. We therefore have only

$$2 \cdot (1 + 80 + \binom{80}{2} + \binom{80}{3}) = 170,802 < 2^{19}$$

or even as few as

$$2 \cdot (1 + 80 + 3160) = 6,482 < 2^{13}$$

possible cases to consider.

Given 2^{32} subhypercubes (a number we can decrease further be considering only subcubes of a minimum Hamming weight) at some 32 time steps, *e.g.* time steps 709 to 740 immediately after the setup, we then have less than

$$2^{13+32+5}$$

potential matches. Hence, 50 bits of information are sufficient to rule out all but one false positives.

There are essentially three methods to sieve for linearity: Gaussian elimination, Blum-Luby-Rubinfeld BLR linearity tests as described in [2] and suggested in [4], and the Wavefront Model [19].

Gaussian elimination with 80 key bits plus the constant '1' requires $81+N$ simulations to obtain N bits of information.

The original BLR test uses four simulations with keys $0, x, y, x \oplus y$ to obtain one bit of information, ruling out linearity with probability about one half. Hence, $1 + N \cdot 3$ simulations are required to give N bits of information. An obvious modification is to use keys $00\ldots0, 11\ldots1, x, \overline{x}$ *i.e.* to always set y to the all-one vector. We then require $2 + N \cdot 2$ simulations for N bits of information.

The Wavefront Model is in essence a purged Gaussian elimination, with 2^{13} or 2^{19} instead of 2^{81} positive cases. Hence, $13 + N$, and $19 + N$, respectively, simulations are sufficient for N bits of information.

We have the following matrix of ranges for $N \in \mathbb{N}$, where the algorithm given at the kleft (row) is better than the algorithm given in the respective column. Example: $W(2)$ is better than BLR tests for $N \geq 6$.

		Algorithm	W(2)	W(3)	G	B(x)	B(x, y)
W(2)	=	Wavefront(K_1, K_2)	=	$\mathbb{N}$	$\mathbb{N}$	$11 - \infty$	$6 - \infty$
W(3)	=	Wavefront(K_1, K_2, K_3)	—	=	$\mathbb{N}$	$17 - \infty$	$9 - \infty$
G	=	Gaussian elimination	—		=	$79 - \infty$	$40 - \infty$
B(x)	=	BLR($x, \overline{x}$)	$1 - 11$	$1 - 17$	$1 - 79$	=	$\mathbb{N}$
B(x, y)	=	BLR($x, y, x \oplus y$)	$1 - 6$	$1 - 9$	$1 - 40$	—	=

As can be observed from Table 2, the BLR tests are even worse than Gaussian elimination for $N > 40$ (original BLR) and $N \geq 80$ (modified BLR). The Wavefront Model with three key bits is faster, hence more efficient than BLR already for $N \geq 10$ (original BLR) and $N \geq 18$ (modified BLR). Using the wavefront Model with only two key bits (which covers the vast majority of all results obtained so far), the threshold point is even lower, the Wavefront model being superior to modified BLR for N as low as $N \geq 12$, while the originally suggested BLR test is slower even for $N = 7$. This amounts to only 128 cases, *e.g.* 16 hypercubes at 8 time steps, a number likely to be exceeded even before any practical linear relations turn up.

4 The Main Result

We tabulate the linear key equations, we obtained from the two 32-dimensional hypercubes, using the Fast Reed-Muller transform on them. We obtained two more linear equations, whose subcube dimensions are however higher than the effort needed to search brute force through the remaining 24 key bits.

Putting up the table took us several days of CPU time (Phase A). The simulations now required to sum within these hypercubes (Phase B) will take about one minute on a fast dual core processor (or 150 seconds on a single core of the 1,8 GHz AMD Turion TL-56, to give a notebook speed). This is deciphering on the fly.

The Phases C and D [17] (Gaussian elimination or matrix inversion), so highly appraised as an important addition to AIDA in [4], actually can be done with paper and pencil: The 56 linear equations include 40 single key bits, and 16 equations requiring a total of 19 additions mod 2.

4.1 The Linear Relations

All IV vector spaces are subspaces of one of the generating sets $B_1 = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 23, 27, 35, 41, 42, 43, 44, 46, 47, 48, 50, 51, 55, 59, 63, 67, 69, 71, 73, 75, 77, 79, 80\}$ and $B_2 = \{2, 4, 6, 8, 10, 12, 14, 16, 23, 25, 29, 32, 34, 36, 38, 40, 43, 45, 47, 51, 53, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80\}$

Line 1 reads: Run 2^{23} simulations assigning all possible combinations to the IV bits with indices 1,3,...,79. Sum up the 2^{23} resulting values at time step 711. The result mod 2 is key bit K_1.

K1	@711	{1,3,5,7,9,15,19,35,42,43,46,48,50,51,55,59,63,67,69,73,75,77,79}	23
K2	@711	{1,5,7,9,11,19,27,35,42,43,46,48,50,51,63,67,69,71,73,75,77,80}	22
K3	@710	{2,4,8,10,12,23,29,34,36,43,47,51,53,62,64,66,68,70,74,76,80}	21
K4	@710	{1,3,5,9,15,23,27,35,42,43,44,46,48,50,51,55,63,67,69,71,73,77,80}	23
K5+E	@711	{2,4,6,8,10,14,16,29,32,34,36,38,43,45,47,51,53,60,62,66,74,76,80}	23
K6	@710	{1,3,5,7,9,11,15,23,35,41,43,44,46,48,51,55,59,63,67,71,73,77,80}	23
K7	@710	{2,4,6,8,10,12,23,29,34,36,38,43,45,47,51,53,60,62,64,66,70,74,78}	23
K9+E	@721	{2,4,6,10,12,14,23,25,32,36,38,... ...,40,43,45,51,53,64,66,68,70,72,74,76,78,80}	25
K11	@710	{2,4,6,8,10,12,14,23,25,29,32,34,36,43,45,53,62,66,70,72,74,76,78,80}	24
K13	@711	{2,4,8,10,14,29,32,34,40,43,45,47,51,53,62,64,66,70,74,76,78,80}	22
K15+K42	@711	{4,6,8,10,14,16,23,29,34,43,45,47,51,53,60,62,64,66,68,78,80}	21
K16	@711	{4,6,8,10,23,32,34,36,38,43,45,47,51,53,60,62,64,66,68,78,80}	21
K17+E	@710	{2,4,6,8,10,14,16,23,29,34,36,38,43,45,47,51,62,64,66,70,76,80}	22
K18	@710	{2,4,6,8,10,14,23,29,32,34,36,47,51,53,62,64,66,68,70,76,78,80}	22
K19 +K4+E	@710	{1,3,5,7,9,11,19,23,35,41,42,43,44,46,48,50,51,55,63,71,73,75,77,80}	24

K20	@710	{2,4,8,10,12,16,23,25,29,34,36,43,45,47,51,53,60,62,64,66,68,70,76}	23
K21	@711	{1,3,5,7,9,11,19,23,27,35,41,42,44,46,48,50,51,59,63,67,77,79}	22
K22+E	@710	{1,3,5,9,11,19,23,35,43,44,46,48,50,51,55,59,63,67,69,73,75,77,79}	23
K24	@711	{1,3,7,9,19,23,35,43,46,48,50,51,55,59,63,67,69,71,73,77,80}	21
K25	@711	{1,3,5,7,9,11,19,23,27,41,43,46,48,50,51,55,59,63,67,71,73,80}	22
K26			
+K11+E	@711	{1,3,7,9,15,17,19,23,27,35,43,46,48,50,51,63,67,69,71,73,77,80}	22
K27+E	@711	{6,8,14,16,29,32,34,36,38,43,45,47,51,53,60,62,64,66,68,76,78,80}	22
K28	@711	{1,3,5,7,9,11,19,23,35,41,44,46,48,50,51,55,63,67,73,77,80}	21
K29	@711	{1,3,5,7,9,11,19,23,35,41,43,46,48,50,51,55,63,67,73,77,80}	21
K30	@711	{1,7,9,15,19,23,27,35,41,42,44,46,48,50,51,55,63,67,71,73,77,80}	22
K31+E	@710	{2,4,8,10,12,16,23,29,32,34,36,38,43,45,47,51,62,66,68,70,74,76,80}	23
K32	@711	{5,7,9,11,15,19,23,27,43,44,46,48,50,51,55,59,63,67,71,73,77,80}	22
K33+K31	@711	{1,3,5,7,9,19,23,27,42,43,44,46,47,48,50,51,55,63,67,71,73,75,77,79}	24
K34+K57	@711	{1,5,7,9,11,17,19,23,35,43,44,48,50,51,55,59,63,67,69,73,75,77,79}	23
K35	@721	{2,4,6,10,12,14,25,32,34,36,38,...	
		...,40,43,45,51,53,60,62,64,66,68,70,74,76,78,80}	26
K37	@712	{4,6,8,10,23,29,34,43,45,47,51,53,60,62,64,66,68,72,78,80}	20
K38	@714	{3,5,7,9,11,15,17,19,27,42,44,46,47,48,50,51,55,59,63,67,69,73,75,79}	24
K39+K37	@712	{4,6,8,10,14,23,29,34,43,45,47,51,53,60,62,66,72,74,76,78,80}	21
K40	@711	{1,3,5,7,9,11,15,19,23,35,41,43,44,46,48,50,51,59,63,67,77,80}	22
K41	@710	{2,4,6,8,10,12,29,32,34,36,40,43,45,47,51,53,60,62,64,66,68,70,74,76}	24
K42	@711	{1,3,5,7,9,11,15,19,23,35,41,43,46,48,50,51,55,63,67,69,73,80}	22
K43	@711	{1,3,5,7,9,11,17,19,27,35,42,44,46,48,50,51,55,63,67,73,75,79}	22
K44	@711	{2,4,6,8,12,16,23,29,32,34,36,43,45,47,51,53,60,62,66,70,78,80}	22
K45	@711	{1,3,5,7,9,11,15,19,23,27,35,43,46,48,50,51,55,59,63,67,73,79}	22
K46	@710	{2,4,10,23,25,29,32,36,43,45,47,51,53,64,66,68,70,72,74,76,78,80}	22
K48	@710	{2,4,6,8,10,12,14,23,29,36,47,51,53,60,62,64,66,68,70,72,74,80}	22
K53	@711	{1,3,5,7,9,11,19,23,35,43,44,46,48,50,55,59,63,67,69,71,73,77,79}	23
K54	@712	{4,6,8,10,12,29,34,45,47,51,53,60,62,64,66,70,72,76,78,80}	20
K55	@710	{2,4,8,10,16,23,29,34,36,43,45,47,51,53,64,66,68,70,76,78,80}	21
K56	@712	{4,6,8,10,23,29,34,36,43,47,51,53,60,62,64,66,72,76,78,80}	20
K57	@711	{2,4,6,8,16,23,25,29,34,36,38,43,45,47,51,53,60,62,66,68,78,80}	22
K58	@710	{2,6,8,10,12,14,23,29,36,40,45,47,51,60,62,64,66,70,74,76,78,80}	22
K59	@711	{2,4,6,8,14,23,29,32,34,43,47,51,53,60,62,66,10,70,72,76,80}	21
K60+K24	@711	{1,5,7,9,11,19,23,27,42,43,46,48,50,51,59,63,67,71,73,75,77,80}	22
K61	@711	{2,4,8,10,14,23,29,34,43,45,47,53,60,62,64,66,70,72,76,78,80}	21
K64	@711	{1,3,5,7,9,11,19,27,35,42,46,48,50,51,55,59,63,67,73,77,80}	21
K65	@710	{2,4,6,8,10,12,23,29,36,47,51,53,62,64,66,68,70,72,74,76,78,80}	22
K66+K45			
+K57	@711	{1,3,5,9,11,15,19,23,27,35,42,43,46,48,50,51,63,67,71,75,77,80}	22
K67+E	@711	{1,3,5,7,9,11,17,19,23,35,42,43,44,46,48,50,51,63,69,71,75,77,79,80}	24
K68+K56	@711	{1,3,7,9,15,17,19,27,35,41,42,43,44,46,48,50,51,55,63,67,69,77,79}	23
K69	@711	{4,6,10,16,23,29,34,36,38,43,45,47,51,53,60,64,66,68,70,74,76,78,80}	23

5 AIDA, BIVIUM & TRIVIUM in Perspective

Most other attacks on BIVIUM use SAT solvers and are very sensitive to the algorithm, generally solving BIVIUM-A in a time comparable to AIDA, but failing spectacularly on BIVIUM-B. On the upside, SAT solvers use any piece of continuous stream after the setup. Arbitrary large setup lengths do not weaken the ability of SAT solvers, while AIDA can attack only when the designers have been overconfident in fixing too short a setup length. Any cipher mixing all key and IV bits will eventually be secure against AIDA.

AIDA on the other hand is insensitive to the slight difference in the output function between BIVIUM-A und BIVIUM-B. After all, both act linearly on the same 177 stage register, whose entries uniformly have undergone 8 quadratic feedback modifications per stage.

Also, TRIVIUM with a shortened setup length of 708, also corresponding to 8 quadratic modifications each, is easily broken by AIDA. Even 792 setup steps still reveal linear key equations (see [18, 19]). Full TRIVIUM, on the other hand, is probably secure against AIDA, as would — probably — BIVIUM with 1152 setup cycles.

The main weakness of BIVIUM compared to TRIVIUM (at least concerning AIDA) hence lies not so much in the shorter register length, but in the shorter setup length, insufficient to thoroughly mix all key and IV bits. Generalizing this observation, we propose therefore:

TLC — TRIVIUM-like-cipher

Let a cipher consist in an N stage register ($N \equiv 0 \mod 3$) with F feedback taps, roughly evenly spaced at positions $f_1, \ldots, f_F \equiv 0 \mod 3$ with feedback function

$$s_{f_i} = s_{f_i} \oplus s_{f_{i-1}} \wedge s_{f_{i-2}} \oplus s_{f_i-\alpha_i} \oplus s_{f_i+\beta_i}$$

with $\alpha_i, \beta_i \equiv 0 \mod 3$. The setup consists in filling up stages s_1 to s_{80} with the key, stages s_{94} to s_{173} with the IV, stages s_{N-3} to s_{N-1} with '1', and all other stages with '0'. We now run the cipher S setup cycles. After that moment, output is obtained by (linearly) adding up to 6 stage bits, from positions a multiple of 3.

TLC–Conjecture

Let a TLC be given with parameters N (length), F (number of feedbacks), and S (setup length). Then $\frac{S \cdot F}{N}$ is the (average) number of updates per stage during the setup phase. AIDA attacking this TLC should have the following result:

$$\frac{S \cdot F}{N} = \begin{cases} 0 \ldots 6 & \text{trivial} \\ 7 \ldots 8 & \text{AIDA effectively breaks TLC} \\ 9 \ldots 10 & \text{effort lower than brute force, but not practical} \\ 11 \ldots 12 & \textbf{?} \\ 13 \ldots \infty & \text{secure against linear AIDA} \end{cases}$$

Conclusion

We have shown that the Algebraic IV Differential Attack, AIDA, breaks BIVIUM-B in one minute of Dual-Core CPU time, far faster than any previous attack. We used two means of speeding up the time-consuming Phase A of the attack, namely the Fast Reed-Muller Transform to evaluate many subhypercubes in parallel, and the Wavefront Model to rule out nonlinear relations early on. We present the TLC-Conjecture about the effectiveness of AIDA against Trivium-Like-Ciphers.

References

[1] Biham, E., O. Dunkelman, *Differential Cryptanalysis in Stream Ciphers* `http://eprint.iacr.org/2007/218`

[2] Blum, M. , M. Luby, R. Rubinfeld, *Self-testing/correcting with applications to numerical problems*, in: Proc. 22nd STOC, p. 73–83, 1990.

[3] Cannière, C. de, B. Preneel, TRIVIUM *Specifications* `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf`

[4] Dinur, I., A. Shamir, *Cube attacks on tweakable black box polynomials* `http://eprint.iacr.org/2008/385`

[5] Faugère, Jean-Ch. ,*A new efficient algorithm for computing Gröbner bases* (F_4), J. P. Appl. Algebra **139** (1999), p. 61–88.

[6] Fischer, S., S. Khazaei, W. Meier, *Chosen IV Statistical Analysis for Key Recovry Attacks on Stream Ciphers*, Proc. AFRICACRYPT 2008, p. 236–245, LNCS 5023, Springer 2008.

[7] Lai, X., *Higher Order Derivatives and Differential Cryptanalysis* in: Communications and Cryptography – Two Sides of One Tapestry, Blahut et al.,Eds., Kluwer, 1994.

[8] Maximov,A., A. Biryukov, *Two Trivial Attacks on Trivium* eprint 2007/021.

[9] McDonald, C., C. Charnes, J. Pieprzyk, *An Algebraic Analysis of Trivium Ciphers based o9n the Boolean Satisfiability Problem*, eprint 2007/129.

[10] O'Neil, S., *Algebraic Structure Defectoscopy* (*ASD Tests*) `http://eprint.iacr.org/2007/378`

[11] Raddum, H., *Cryptanalytic results on Trivium*, eSTREAM report 2006/039.

[12] Sun, Z., Shi-Wu Zhang, L. Wang, *Chosen IV algebraic attack on One.Fivium*, Proc. ISKE 2008, Intl. Conf. Intelligent Systems and Knowledge Engineering, 2008.

[13] Thornton, M. A., D. Michael Miller, Rolf Drechsler, *Transformations amongst the Walsh, Haar, Arithmetic and Reed-Muller Spectral Domains*, Proc. 4th Intl. Workshop on Appl. of Reed-Muller Expansion in Circuit Design, pp. 215–225, August 2001. http://www.cs.uvic.ca/~mmiller/publications/rm2001.pdf

[14] Turan, M. S., Kara, O., *Linear Approximations for* 2*-round Trivium* http://www.ecrypt.eu.org/stream/papersdir/2007/008.pdf

[15] Vielhaber, M., *Breaking* ONE.FIVIUM *by AIDA an Algebraic IV Differential Attack* http://eprint.iacr.org/2007/413

[16] Vielhaber, M., TRIVIUM*'s output partially autocancels* http://eprint.iacr.org/2008/377

[17] Vielhaber, M., *Shamir's "cube attack": A Remake of AIDA, The Algebraic IV Differential Attack* hs-bremerhaven.de/Binaries/Binary10017/AIDA_Shamir.pdf

[18] Vielhaber, M., "AIDA vs. TRIVIUM 793:1152 Final Score 980:1152" Rump Session Eurocrypt 2009, Cologne, http://eurocrypt2009rump.cr.yp.to

[19] Vielhaber, M.,"Speeding up AIDA, the Algebraic IV Differential Attack, by the Fast Reed-Muller Transform", ISKE 2009, International Conference on Intelligent Systems and Knowledge Engineering, Hasselt/Belgium, 2009.

[20] Weinmann, R.–Ph., A. Pyshkin, *Meaningful results against Trivium with reduced key setup* http://cryptanalysis.eu/blog/2007/11/12/trivium