Conditional Differential Cryptanalysis of Trivium and KATAN

Simon Knellwolf*, Willi Meier, and María Naya-Plasencia**

FHNW, Switzerland

Abstract. The concept of conditional differential cryptanalysis has been applied to NLFSR-based cryptosystems at ASIACRYPT 2010. We improve the technique by using automatic tools to find and analyze the involved conditions. Using these improvements we cryptanalyze the stream cipher Trivium and the KATAN family of lightweight block ciphers. For both ciphers we obtain new cryptanalytic results. For reduced variants of Trivium we obtain a class of weak keys that can be practically distinguished up to 961 of 1152 rounds. For the KATAN family we focus on its security in the related-key scenario and obtain practical key-recovery attacks for 120, 103 and 90 of 254 rounds of KATAN32, KATAN48 and KATAN64, respectively.

Keywords: Trivium, KATAN, conditional differential cryptanalysis.

1 Introduction

The stream cipher Trivium and the KATAN family of block ciphers are lightweight cryptographic primitives dedicated to hardware implementation. They share a very similar structure based on non-linear feedback shift registers (NLFSR). In [12], conditional differential cryptanalysis, first introduced in [3], has been applied to such constructions. The idea is to control the propagation of differences by imposing conditions on the public variables of the cipher. Depending whether these conditions involve secret variables or not, key-recovery or distinguishing attacks can be mounted. The technique extends to higher order differential cryptanalysis. A similar concept is the dynamic cube attack presented in [9]. Deriving the conditions by hand is a time consuming and error prone task. In this paper we use automatic tools to find and simplify these conditions. The method is applied to KATAN and Trivium. In both cases we obtain new cryptanalytic results.

In the single-key scenario, the KATAN family was already analyzed with respect to conditional differential cryptanalysis in [12]. Table 1 summarizes the

* Supported by the Hasler Foundation `www.haslerfoundation.ch` under project number 08065.

** Supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under grant number 5005-67322.

A. Miri and S. Vaudenay (Eds.): SAC 2011, LNCS 7118, pp. 200–212, 2012.
© Springer-Verlag Berlin Heidelberg 2012

Table 1. Cryptanalytic results for KATAN. All attacks have practical complexity and recover parts of the key. The results in the single-key scenario also apply to KTANTAN.

block size	scenario	rounds	reference
32	single-key	78	[12]
	related-key	120	this paper
48	single-key	70	[12]
	related-key	103	this paper
64	single-key	68	[12]
	related-key	90	this paper

results and compares them to the results in the related-key scenario presented in this paper. The question about the related-key security of KATAN was raised by very efficient such attacks on KTANTAN [1]. The KTANTAN family of block ciphers differs from KATAN only by its key scheduling. The latter has shown some vulnerability which was also exploited for a meet-in-the-middle attack [4].

The most relevant cryptanalytic results on Trivium are obtained by cube attacks [8] and cube testers [2,15]. Our analysis can be seen as a refinement of cube testers. Exploiting these refinements for Trivium is the subject of the second part of this paper. Table 2 summarizes the results and compares them to existing analysis.

Table 2. Cryptanalytic results for Trivium

rounds	complexity	# keys	type of attack	reference
767	2^{45}	all	key recovery	[8]
790	2^{31}	all	distinguisher	[2]
798	2^{25}	all	distinguisher	this paper
806	2^{44}	all	distinguisher	[15]
868	2^{25}	2^{31}	distinguisher	this paper
961	2^{25}	2^{26}	distinguisher	this paper

The paper is organized as follows. Section 2 reviews conditional differential cryptanalysis and describes an approach to analyze the conditions more automatically. In Sections 3 and 4 we apply the technique to KATAN and Trivium.

2 Review of Conditional Differential Cryptanalysis

The idea of conditional differential cryptanalysis has been introduced in [3]. In [12] it has been extended to higher order cryptanalysis and applied to NLFSR-based constructions. We briefly review the concept and then sketch our strategies to analyze the conditions more automatically.

2.1 Conditional Differential Cryptanalysis

Suppose a prototypical NLFSR-based cipher with an internal state of length ℓ which is initialized with a key k and an initial value x. Let $s_0, s_1, \ldots$ be the consecutive state bits generated by the cipher, such that $(s_i, \ldots, s_{i+\ell})$ is the state after i rounds, and let h be the output function of the cipher such that $h(s_i, \ldots, s_{i+\ell})$ is the output after i rounds. Every state bit is a function of (k, x) and the same is true for the output of h. For some fixed i, let $f = h(s_i, \ldots, s_{i+\ell})$.

In differential cryptanalysis one computes derivatives of f. Following [13], the derivative of f with respect to a is defined as

$$\Delta_a f(k, x) = f(k, x) + f(k, x \oplus a).$$

A biased output distribution distinguishes the cipher from an ideal primitive and may reveals information on the key. The idea of conditional differential cryptanalysis is to derive conditions on x that control the propagation of the difference up to some round r. This results in a system of equations

$$\Delta_a s_i(k, x) = \gamma_i, \quad 0 \leq i < r. \tag{1}$$

The γ_i are either 0 or 1 and describe the differential characteristic. Values x that satisfy all conditions are called *valid.* The goal is to find a large sample of valid inputs $\mathcal{X}$, such that a bias can be detected in the output of $\Delta_a f$ on $\mathcal{X}$. The conditions may also involve variables of the key. This allows for key recovery or classification of weak keys.

The technique extends to higher order derivatives (corresponding to higher order differential cryptanalysis). The d-th derivative of f with respect to $a_1, \ldots, a_d$ is defined as

$$\Delta^{(d)}_{a_1,\ldots,a_d} f(k, x) = \sum_{c \in L(a_1,\ldots,a_d)} f(k, x \oplus c),$$

where $L(a_1, \ldots, a_d)$ is the set of all 2^d linear combinations of $a_1, \ldots, a_d$. In [12] it was proposed to analyze the first order propagation of each difference a_i and to merge the obtained conditions. This technique was successfully applied to Grain-128 in and we will apply it to Trivium in this paper.

2.2 Automatic Strategies for Analyzing the Conditions

Analyzing the conditions is a crucial part of conditional differential cryptanalysis. There is a trade-off between the number of controlled rounds and the size of the sample $\mathcal{X}$. Controlling more rounds means to impose more conditions, which reduces the number of valid inputs that can be derived. In general, the conditions are not independent of each other and may be simplified during the process. This makes the analysis complicated and prone to error when done by hand. In the case of higher order derivatives this tends to be even more intricate.

In order to do a more automatic analysis, we represent the system of conditions as an ideal J in the ring of Boolean polynomials $\mathbb{F}_2[K, X]$. All (k, x) in the algebraic variety of J satisfy the imposed conditions[1]. We then use the PolyPoRi library [5] to perform computations in Boolean polynomial rings. Specifically, we use modular reductions to analyze new conditions with respect to already imposed conditions, and to obtain a simple representation of J.

We distinguish two strategies for computing J. The strategies differ in whether the differential characteristic is fixed in advance (for example by linearization) or if it is derived in parallel with the conditions.

Differential Characteristic Fixed in Advance. This is the simple strategy and we will use it in our analysis of KATAN. Consider the system of equations given by (1) and assume that $\gamma_0, \ldots, \gamma_{r-1}$ are given. Algorithm 1 either returns the ideal describing the exact conditions on k and x for following the characteristic, or it returns with a message that the characteristic is impossible.

Algorithm 1. Deriving conditions for a given characteristic.

```
Input: $a$, $\gamma_0, \ldots, \gamma_{r-1}$
Output: Ideal of conditions
$J \leftarrow \emptyset$
for $i \leftarrow 0$ to $r - 1$ do
    $f \leftarrow \Delta_a s_i(k, x) \oplus \gamma_i \mod J$
    if $f = 1$ then
        return impossible characteristic
    else
        add $f$ to $J$
return $J$
```

Differential Characteristic Derived in Parallel. In some cases it can be difficult to choose a characteristic in advance. This is particularly true for higher order derivatives where several characteristics have to be chosen such that their respective conditions do not contradict each other. A straightforward extension of Algorithm 1 would fail in most cases. Algorithm 2 provides more flexibility. It takes as input only the initial difference, and at each step develops the characteristic based on the conditions imposed so far. At those steps where γ_i can take both values (0 or 1), the algorithm chooses $\gamma_i = 0$ (it prevents the propagation of the difference). Other strategies are possible, but we found this strategy the most successful in our applications.

Algorithm 3 is an extension to the higher order case and we will use it in our analysis of Trivium. Note that this algorithm does not explicitly compute the characteristics. They are not used for the attack, and in Algorithm 2 the characteristic is computed only for illustration.

[1] The algebraic variety of J is the set $\{(k, x) \mid f(k, x) = 0 \text{ for all } f \in J\}$.

Algorithm 2. Deriving characteristic in parallel to conditions.

```
Input: $a$, $r$
Output: Differential characteristic and ideal of conditions
$J \leftarrow \emptyset$
for $i \leftarrow 0$ to $r - 1$ do
    $f \leftarrow \Delta_a s_i(k, x) \mod J$
    if $f = 1$ then
        $\gamma_i \leftarrow 1$
    else
        $\gamma_i \leftarrow 0$
        add $f$ to $J$
return $(\gamma_0, \dots, \gamma_{r-1})$, $J$
```

Algorithm 3. Extension of Algorithm 2 to higher order derivatives.

```
Input: $a_1, \dots, a_d$, $r$
Output: Ideal of conditions
$J \leftarrow \emptyset$
foreach $a \in \{a_1, \dots, a_d\}$ do
    for $i \leftarrow 0$ to $r - 1$ do
        $f \leftarrow \Delta_a s_i(k, x) \mod J$
        if $f \neq 1$ then
            add $f$ to $J$
return $J$
```

The algorithms usually produce a very simple representation of J which directly allows to analyze the dependence on bits of the key, and to derive the respective sample(s) $\mathcal{X}$. If necessary, more advanced techniques can be applied, for example Gröbner basis algorithms.

3 Related-Key Attacks for Reduced KATAN

We now evaluate the security of KATAN against conditional differential cryptanalysis in a related-key attack scenario. More specifically, an attacker is assumed to obtain two ciphertexts for each chosen plaintext: one encrypted under a secret key k and the other encrypted under $k \oplus b$ for a chosen difference b.

3.1 Description of KATAN

KATAN [7] is a family of lightweight block ciphers proposed De Cannière, Dunkelman and Knezevic. The family consists of three ciphers denoted by KATANn for $n = 32, 48, 64$ indicating the block size. All instances accept an 80-bit key. KATANn has a state of n bits which are aligned as two non-linear feedback shift registers. For $n = 32$, the registers have lengths 13 and 19, respectively. They are initialized with the plaintext:

$$\begin{aligned}(s_1, \ldots, s_{19}) &\leftarrow (x_0, \ldots, x_{18})\\(s_{20}, \ldots, s_{32}) &\leftarrow (x_{19}, \ldots, x_{31}).\end{aligned}$$

The key is expanded to 508 bits according to the linear recursion

$$k_{j+80} = k_j + k_{j+19} + k_{j+30} + k_{j+67}, \quad 0 \leq j < 428,$$

where $k_0, \ldots, k_{79}$ are the bits of k. At each round of the encryption process two consecutive bits of the expanded key are used. The round updates further depend on a bit c_i. The sequence of c_i is produced by an 8-bit linear feedback shift register which is used as a counter. It is initialized by $(c_0, \ldots, c_7) = (1, \ldots, 1, 0)$ and expanded according to $c_{i+8} = c_i + c_{i+1} + c_{i+3} + c_{i+8}$. Round i, for $0 \leq i < 254$, corresponds to the following transformation of the state:

$$\begin{aligned}t_1 &\leftarrow s_{32} + s_{26} + s_{28}s_{25} + s_{23}c_i + k_{2i}\\t_2 &\leftarrow s_{19} + s_7 + s_{12}s_{10} + s_8s_3 + k_{2i+1}\\(s_1, \ldots, s_{19}) &\leftarrow (t_2, s_1, \ldots, s_{18})\\(s_{19}, \ldots, s_{32}) &\leftarrow (t_1, s_{19}, \ldots, s_{31})\end{aligned}$$

After 254 rounds, the state is output as the ciphertext. All three members of the KATAN family use the same key expansion and the same sequence of c_i. The algebraic structure of the non-linear update functions is the same. They differ in the length of the non-linear registers and the tap positions for the non-linear update functions. All members perform 254 rounds, but for KATAN48 the non-linear registers are updated twice per round and for KATAN64 even thrice (using the same c_i and k_i for all updates at the same round).

3.2 Basic Analysis Strategy

As in the analysis of KATAN in [12] we use first order differentials. The basic strategy is as follows:

1. Find a key difference b whose expansion does not introduce differences for many rounds after some round r. The idea is to cancel all differences introduced by b up to round r and to maximize the number of rounds, where no differences are introduced again.
2. Compute backwards from round r in order to find a plaintext difference a that cancels the differences introduced by b. This fixes a differential characteristic.
3. Use Algorithm 1 to compute the ideal J, describing the conditions for the characteristic to be followed.
4. Derive a sample of valid plaintexts and empirically find the maximal number of rounds for which a bias can be detected in the ciphertext differences.

The automated techniques for condition analysis allow to test many configurations for a and b. The maximal number of consecutive rounds b does not introduce differences is 39 (the key expansion is a 80-bit linear feedback shift register with maximum period and two bits are used per round). It is easy to compute differences which have this maximal run of zeros at any desired round r, and the choice of b essentially reduces to a choice of r. We try to find the largest r that can be controlled by conditions. If key bits are involved in the conditions, several samples will be derived and tested for the correct guess.

3.3 Analysis of KATAN32

We now describe the details to attack 120 rounds of KATAN32. We use the key difference $b = [6, 14, 25, 44]$ which means differences at positions 6,14,25 and 44 of the key. The expanded key difference is given in Table 3. Note that no differences are introduced after round $r = 22$ for the subsequent 39 rounds. By backward computation we find that the plaintext difference $a = [6, 9, 19]$ cancels all differences up to round 22. The corresponding characteristic is given in Table 4.

Table 3. Expanded key difference $b = [6, 14, 25, 44]$

Rnds	Round key differences
0-19	00 00 00 10 00 00 00 10 00 00 00 00 01 00 00 00 00 00 00 00
20-39	00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40-59	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60-79	00 00 10 00 00 00 00 00 01 00 00 00 00 00 00 10 00 00 00 00
80-99	00 01 00 00 00 00 00 10 10 00 00 00 01 00 01 00 00 00 00 00
100-119	10 10 10 00 00 01 00 01 00 00 00 00 10 10 10 10 00 00 00 00
...	...
240-253	01 01 00 10 11 10 11 10 11 10 00 01 10 11

Using Algorithm 1 we compute the following ideal J given by

$$\begin{gathered}
J = \langle x_{11} + 1, x_1, x_7, x_8 + 1, x_{22}, x_4, x_5 + x_{10} + x_{16} + k_5, x_6 + x_9 + x_{17} + k_3, \\
x_0 + x_3x_{10} + x_3x_{16} + x_3k_5 + k_{15}, x_2x_{20} + x_2x_{24} + x_2x_{29} + x_2k_4 + x_{12} + k_{13}, \\
x_3x_{10}x_{21} + x_3x_{10}x_{23}x_{26} + x_3x_{10}x_{25} + x_3x_{10}x_{30} + x_3x_{10}k_2 + x_3x_{16}x_{21} \\
+x_3x_{16}x_{23}x_{26} + x_3x_{16}x_{25} + x_3x_{16}x_{30} + x_3x_{16}k_2 + x_3x_{19} + x_3x_{21}x_{24} \\
+x_3x_{21}k_5 + x_3x_{23}x_{26}k_5 + x_3x_{23} + x_3x_{25}k_5 + x_3x_{28} + x_3x_{30}k_5 + x_3k_2k_5 \\
+x_3k_6 + x_3 + x_9 + x_{10}x_{12}x_{19} + x_{10}x_{12}x_{21}x_{24} + x_{10}x_{12}x_{23} + x_{10}x_{12}x_{28} \\
+x_{10}x_{12}k_6 + x_{10}x_{12} + x_{17} + x_{18}x_{19} + x_{18}x_{21}x_{24} + x_{18}x_{23} + x_{18}x_{28} + x_{18}k_6 \\
+x_{18} + x_{19}x_{23} + x_{19}k_1 + x_{19}k_{16} + x_{20}x_{23} + x_{21}x_{23}x_{24} + x_{21}x_{24}k_1 + x_{21}x_{24}k_{16} \\
+x_{21}k_{15} + x_{23}x_{26}k_{15} + x_{23}x_{28} + x_{23}k_1 + x_{23}k_6 + x_{23}k_{16} + x_{23} + x_{25}k_{15} + x_{27} \\
+x_{28}k_1 + x_{28}k_{16} + x_{30}k_{15} + k_1k_6 + k_1 + k_2k_{15} + k_3 + k_6k_{16} + k_8 + k_{25}\rangle .
\end{gathered}$$

Table 4. Differential characteristic for $a = [6, 9, 19]$ and $b = [6, 14, 25, 44]$

Round	Difference in state
0	00000010010000000001000000000000
1	00000001001000000000100000000000
2	00000000100100000000010000000000
3	00000000010010000000001000000000
4	00000000001001000000000100000000
5	00000000000100100001000010000000
6	00000000000010010000100001000000
7	00000000000001001000010000100000
8	00000000000000100100001000010000
9	00000000000000010010000100001000
10	00000000000000001001000010000100
11	00000000000000000100100001000010
12	00000000000000000010010000100001
13	00000000000000000000001000010000
14	00000000000000000000000100001000
15	00000000000000000000000010000100
16	00000000000000000000000001000010
17	00000000000000000000000000100001
18	00000000000000000000000000010000
19	00000000000000000000000000001000
20	00000000000000000000000000000100
21	00000000000000000000000000000010
22	00000000000000000000000000000001
23	00000000000000000000000000000000
	...
62	00000000000000000000000000000000
63	10000000000000000000000000000000
64	01000000000000000000000000000000

All pairs (k, x) in the algebraic variety of J will follow the characteristic given in Table 4. The conditions involve 10 bits of the key which can not be chosen. However, we can guess them and adjust x accordingly. It is not difficult to derive a sample of 2^{20} valid inputs for each guess. One adjusts a linear variable of each condition in order to nullify the expression. The remaining variables can be freely chosen. The correct guess is detected by a significant bias in the difference of state bit 18 after 120 rounds. Testing one sample costs 2^{21} queries and at most 2^{10} samples have to tested. Hence, the attack needs not more than 2^{31} queries to the cipher. The number of different queries can be even smaller, since the samples for the different guesses may overlap. The attack recovers 10 bits of the key, and we note that the recovered bits are essentially those of the first few rounds. This enables us to mount the same procedure starting at a later round, and finally to recover the full key at essentially the same cost.

3.4 Summary of Results

Table 5 presents the best configurations we found for the different members of the KATAN family. It contains the differences a and b, the number of rounds for which a bias can be detected and the cost of the attack. The latter is computed as $2^{|\mathcal{X}|+\kappa+1}$, where $|\mathcal{X}|$ is the sample size and κ is the number of key bits that must be guessed.

Table 5. Summary of the results for KATANn

n	plaintext difference	key difference	# rounds	cost
32	$[6, 9, 19]$	$[6, 14, 25, 44]$	120	2^{31}
48	$[1, 2, 10, 11, 19, 20, 28, 38, 39, 44, 45]$	$[8, 27]$	103	2^{25}
64	$[6, 7, 8, 19, 20, 21, 34, 58, 59, 60]$	$[2, 21]$	90	2^{27}

4 Weak Keys for Reduced Trivium

We now apply conditional differential cryptanalysis to the stream cipher Trivium. Our analysis leads to a classification of weak keys for reduced variants.

4.1 Description of Trivium

Trivium [6] was designed by De Cannière and Preneel and was selected for the final eSTREAM portfolio [10]. It takes a 80-bit key k and a 80-bit initial value x as input. The internal state consists of 288 bits which are aligned in three non-linear feedback shift registers of lengths 93, 84 and 111, respectively. They are initialized as follows:

$$\begin{aligned}
(s_1, \ldots, s_{93}) &\leftarrow (k_0, \ldots, k_{79}, 0, \ldots, 0) \\
(s_{94}, \ldots, s_{177}) &\leftarrow (x_0, \ldots, x_{79}, 0, 0, 0, 0) \\
(s_{178}, \ldots, s_{288}) &\leftarrow (0, \ldots, 0, 1, 1, 1).
\end{aligned}$$

The state is then updated iteratively by the following round transformation:

$$\begin{aligned}
t_1 &\leftarrow s_{66} + s_{93} \\
t_2 &\leftarrow s_{162} + s_{177} \\
t_3 &\leftarrow s_{243} + s_{288} \\
z &\leftarrow t_1 + t_2 + t_3 \\
t_1 &\leftarrow t_1 + s_{91}s_{92} + s_{171} \\
t_2 &\leftarrow t_2 + s_{175}s_{176} + s_{264} \\
t_3 &\leftarrow t_3 + s_{286}s_{287} + s_{69} \\
(s_1, \ldots, s_{93}) &\leftarrow (t_3, s_1, \ldots, s_{92}) \\
(s_{94}, \ldots, s_{177}) &\leftarrow (t_1, s_{94}, \ldots, s_{176}) \\
(s_{178}, \ldots, s_{288}) &\leftarrow (t_2, s_{178}, \ldots, s_{287}).
\end{aligned}$$

No output is produced during the first 1152 rounds. After this initialization phase the value of z is output as the key stream at each round.

4.2 Basic Strategy of Analysis

We will use a derivative of order $d = 24$ in our analysis. For the 24 differences, we derive conditions using Algorithm 3. Instead of deriving a set of valid inputs we will derive neutral variables for the derivative. Neutral variables have been used in a similar context in [2,11], for example. Let $\Delta f(k, x)$ be the derivative under consideration, and let e_i be the 1-bit difference at bit position i of x. By the *neutrality* of x_i in Δf we mean the probability that $\Delta f(k, x) = \Delta f(k, x \oplus e_i)$ for a random key k. Using a single neutral variable as a distinguisher needs at least two evaluations of Δf. In the case of a d-th derivative this reveals to 2^{d+1} queries to f. If the neutrality of x_i is p, the resulting distinguishing advantage is $|1/2 - p|$.

4.3 Choosing the Differences

It turns out that differences of hamming weight one give the best results. That is, the $a_1, \ldots, a_d$ are unit vectors in $\mathbb{F}_2^n$. We note that this special case of a higher order derivative is called a superpoly in [2]. Some heuristic techniques for choosing the differences have been proposed. We use none of them, but briefly explain our choice. The propagation of the single differences should be as independent as possible. This excludes for example, choosing two differences at a distance one. Such neighboring differences influence each other in the very early rounds due to the quadratic monomials in the update functions. Further, the regular structure of Trivium suggests a regular choice of the differences. Motivated by an observation in [14] we chose the differences at a distance of three. Empirical tests confirmed that this choice indeed outperforms all other choices. Specifically, we choose $a_i = e_{3(i-1)}$ for $1 \leq i \leq 24$, where $(e_0, \ldots, e_{n-1})$ is the standard basis of $\mathbb{F}_2^n$. In the following we use the shorthand $\Delta z_j = \Delta^{(24)}_{a_1,\ldots,a_{24}} z_j$, where z_j is the keystream produced in round j. (In the terminology of [2], Δz_j corresponds to the superpoly of $\{x_0, x_3, \ldots, x_{69}\}$.)

4.4 Analysis of Conditions

For the condition analysis we use Algorithm 3 with $r = 200$, that is, each difference is controlled for the first 200 rounds. After processing the first difference (the difference in x_0) we obtain

$$
\begin{gathered}
J = \langle x_1, x_{12}x_{13} + x_{14}, x_{14}x_{15} + x_{16}, x_{77} + k_{65}, \\
x_{62} + x_{75}x_{76} + x_{75}k_{64} + x_{76}k_{63} + k_{50} + k_{63}k_{64} + k_{75}k_{76} + k_{77}, \\
x_{64} + k_{52} + k_{77}k_{78} + k_{79}, k_{12}k_{13} + k_{14} + k_{56}, \\
k_{14}k_{15} + k_{16} + k_{58} \rangle.
\end{gathered}
$$

At this stage, J has the following interpretation: all pairs (k, x) in the algebraic variety of J follow the same differential characteristic up to round $r = 200$ with respect to a_1. We already note that two conditions can not be satisfied by the attacker, since they only involve bits of the key. After processing the remaining differences we have

$$\begin{gathered} J = \langle x_1, x_2, x_4, x_5, x_7, x_8, x_{10}, x_{11}, x_{13}, x_{14}, x_{16}, x_{17}, x_{19}, x_{20}, \\ x_{22}, x_{23}, x_{25}, x_{26}, x_{28}, x_{29}, x_{31}, x_{32}, x_{34}, x_{35}, x_{37}, x_{38}, x_{40}, x_{41}, \\ x_{43}, x_{44}, x_{46}, x_{47}, x_{49}, x_{50}, x_{52}, x_{53}, x_{55}, x_{56}, x_{58}, x_{59}, x_{61}, x_{62}, \\ x_{64}, x_{65}, x_{67}, x_{68}, x_{70}, x_{71}, x_{73}, x_{74}, x_{76}, x_{77}, x_{79}, k_1, k_2, k_4, \\ k_5, k_7, k_8, k_{10}, k_{11}, k_{13}, k_{14}, k_{16}, k_{17}, k_{19}, k_{20}, k_{22}, k_{23}, k_{25}, \\ k_{26}, k_{28}, k_{29}, k_{31}, k_{32}, k_{34}, k_{35}, k_{37}, k_{38}, k_{40}, k_{41}, k_{43}, k_{44}, k_{46}, \\ k_{47}, k_{49}, k_{50}, k_{52}, k_{53}, k_{55}, k_{56}, k_{58}, k_{59}, k_{61}, k_{62}, k_{64}, k_{65}, k_{66}, \\ k_{67} + 1, k_{68}, k_{70}, k_{71}, k_{73}, k_{74}, k_{76}, k_{77}, k_{79} \rangle . \end{gathered}$$

All conditions collapse to conditions on single bits. From x, only the bits x_{72}, x_{75} and x_{78} are not fixed by conditions and not touched by the differences. This makes them candidate neutral bits for Δz_j, when all other variables x_i are set to zero. Empirical results confirm that they are probabilistically neutral up to round 798. Table 6 shows the neutrality which we obtained in an experiment with 100 random keys. Note that a neutrality of zero means that Δz_j is linear in the corresponding variable (which can be exploited as a distinguishing property in the same way as neutrality).

Table 6. Neutrality of the bits x_{72}, x_{75} and x_{78}

j	72	75	78
772	1.00	1.00	1.00
782	0.05	0.10	0.05
789	0.30	0.20	0.25
798	0.40	0.40	0.30

4.5 Classes of Weak Keys

From the above representation of J we can directly read a class of weak keys, namely the keys satisfying the given 54 conditions on the k_i. This class contains 2^{26} keys. Analogous to Table 6, Table 7 shows the neutrality of the bits x_{72}, x_{75} and x_{78} for a random weak key. We note that x_{75} can not be used anymore as a distinguisher at round $j = 961$, but x_{72} and x_{78} still can.

In order to reduce the number of conditions on the key we processed only a part of the differences by Algorithm 3. For example, for the first 17 differences we obtain only 49 conditions, and for the corresponding class of 2^{31} keys, the bits x_{72}, x_{75} and x_{78} are neutral up to round 868.

Table 7. Neutrality of the bits x_{72}, x_{75} and x_{78} for weak keys

j	72	75	78
953	1.00	1.00	1.00
961	0.00	0.50	1.00

5 Conclusion

We evaluated the security of Trivium and KATAN with respect to conditional differential cryptanalysis. We used an automatic approach to find and analyze the conditions in terms of polynomial ideals. For reduced Trivium we identified a class of 2^{26} keys that can be distinguished for 961 of 1152 rounds. For reduced KATAN we presented a key recovery attack up to 120 of 254 rounds in a related key scenario. KATAN seems to have a comfortable security margin with respect to the approach described in this paper.

Acknowledgements. We thank the reviewers of SAC 2011 for their helpful comments encouraging us to describe our analysis in more detail. This work was partially supported by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References

1. Ågren, M.: Some Instant- and Practical-Time Related-Key Attacks on KTANTAN32/48/64. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 217–233. Springer, Heidelberg (2011)
2. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
3. Ben-Aroya, I., Biham, E.: Differential Cryptanalysis of Lucifer. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 187–199. Springer, Heidelberg (1994)
4. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
5. Brickenstein, M., Dreyer, A.: PolyBoRi: A framework for Groebner-basis computations with Boolean polynomials. Journal of Symbolic Computation 44(9), 1326–1345 (2009)
6. De Cannière, C.: TRIVIUM: A Stream Cipher Construction Inspired by Block Cipher Design Principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006)
7. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
8. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)

9. Dinur, I., Shamir, A.: Breaking Grain-128 with Dynamic Cube Attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011)
10. ECRYPT: The eSTREAM project, `http://www.ecrypt.eu.org/stream/`
11. Fischer, S., Khazaei, S., Meier, W.: Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
12. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 130–145. Springer, Heidelberg (2010)
13. Lai, X.: Higher order derivatives and differential cryptanalysis. In: Blahut, R.E., Costello, D.J., Maurer, U., Mittelholzer, T. (eds.) Communicationis and Cryptography: Two Sides of one Tapestry, pp. 227–233. Kluwer Academic Publishers (1994)
14. Maximov, A., Biryukov, A.: Two Trivial Attacks on Trivium. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 36–55. Springer, Heidelberg (2007)
15. Stankovski, P.: Greedy Distinguishers and Nonrandomness Detectors. In: Gong, G., Gupta, K.C. (eds.) INDOCRYPT 2010. LNCS, vol. 6498, pp. 210–226. Springer, Heidelberg (2010)