Generic related-key and induced chosen IV attacks using the method of key differentiation

Enes Pasalic and Yongzhuang Wei

1. University of Primorska FAMNIT & IAM, Koper, Slovenia,
e-mail: enes.pasalic6@gmail.com
2. Guilin University of Electronic Technology, Guilin City,
Guangxi Province 541004, P.R. China, e-mail: walker_wei@msn.com

Abstract

Related-key and chosen IV attacks are well known cryptanalytic tools in cryptanalysis of stream ciphers. Though the related-key model is considered to be much more unrealistic scenario than the chosen IV model we show that under certain circumstances the attack assumptions may become equivalent. We show that the key differentiation method induces a generic attack in a related-key model whose time complexity in the on-line phase is less than the exhaustive key search. The case of formal equivalency between the two scenarios arises when so-called *differentiable polynomials* with respect to some subset of key variables are a part of the state bit expressions (from which the output keystream bits are built). Then the differentiation over a key cube has the same effect as the differentiation over the corresponding IV cube, so that a generic nature of a related-key model is transferred into a more practical chosen IV model. The existence of such polynomials is confirmed for the reduced round stream cipher TRIVIUM up to some 710 rounds and an algorithm for their detection is proposed. The key differentiation method induces a time/related-key trade-off (TRKTO) attack which (assuming the existence of differentiable polynomials) can be run in a chosen IV model. The resulting trade-off curve of our TMDTO attack is given by $T^2M^2D^2 = (KV)^2$ (V denoting the IV space), which is a significant improvement over the currently best known trade-off $TM^2D^2 = (KV)^2$ [12].

Keywords: Stream ciphers, Cryptanalysis, Chosen IV attacks, Related-key attacks, Time-Memory-Data trade-off attacks, Differntiable polynomials, Key differentiation.

1 Introduction

The use of concepts similar to differential cryptanalysis for block ciphers was frequently employed in cryptanalysis of stream ciphers during the eSTREAM project [14]. These attacks are commonly referred as chosen IV or related-key attacks where the attacker is supposed to freely choose IV values and/or different keys in order to extract certain information about the secret state and/or key bits.

This approach has been successfully used in the cryptanalysis of the stream cipher LILI-128 [3], where different IV's used with a fixed secret key revealed a partial information

about the state bits. Also a similar approach was deployed in [25] in cryptanalysis of TURING cipher. In other direction the eSTREAM proposal Py [8] (and its tweaks PyPy and PyPy6) was successfully cryptanalyzed using the differential cryptanalysis [24].

A general framework of differential cryptanalysis in stream ciphers was studied in [7], where the effect of IV/key difference is analyzed through the differential characteristics at the state variables $((\Delta key, \Delta IV) \rightarrow \Delta S)$. In more recent papers [16, 17, 28], the approaches based on statistical analysis or efficient application of chosen IV attacks have been investigated. In this context we also mention a generic attack known as Cube Attack [11], where the method of tweaking the polynomials (as a generic method applicable in a black box scenario) may lead to efficient attacks on certain stream cipher designs. In this approach, the attacker is supposed to heuristically find the so-called cube indices of an IV vector so that the chosen IV attack with respect to the summation of outputs over all cube points eventually results in linear equations. While this has been successfully applied to TRIVIUM [10] with the KSA reduced to 672 rounds, the verfication of the algorithm [5] has shown that the method fails in providing linear equations when the number of rounds is increased to 735 or 770.

Related-key attacks on stream cipher are often of limited practical significance, though there are examples of its efficient applications and it is regarded as a standard attack scenario (cf. [7]). For instance, related-key weaknesses of the stream cipher RC4 led to a practical attack on the WEP protocol [18]. Furthermore, a related-key approach was used to mount a distinguishing attack on the family of stream ciphers Py, PyPy, PyPy6 and TpyPy [27]. The output keystream sequence of these ciphers could be distinguished from a true (pseudo)random sequence when the identical IV is used to encrypt two messages with different and related keys. Related-key attacks have also been widely employed in block cipher cryptanalysis, see e.g. [26, 13, 23].

Though the main idea of the above mentioned cube attacks is somewhat similar to the approach taken in this work, there are significant differences between the two. In the first place, our approach does not necessarily seeks for linear (low degree) equations. The generic approach that combines the ideas of a chosen IV and a TMDTO attack only requires the existence of so-called *differentiable polynomials* as a part of the keystream bit expressions. In other words, we consider the case when the output keystream contains a polynomial of the form $d(IV, K) = \prod_{l=1}^{m}(IV_{i_l} + k_{j_l})g(IV^*, K^*)$, where $g(IV^*, K^*)$ is some complex function in the key and IV variables but does not depend on the key variables $k_{j_1}, \ldots, k_{j_m}$ (and possibly does not depend on the variables $IV_{i_1}, \ldots, IV_{i_m}$ either). In addition, if the IV cube $IV_{i_1}, \ldots, IV_{i_m}$ is not covered by some larger IV cube (where covering has a standard notion cf. Section 2.3) the differentiation over the key cube or alternatively over the IV cube given above leads to the cancellation of all the terms but the polynomial $g(IV^*, K^*)$. Since this polynomial does not depend on the key variables $k_{j_1}, \ldots, k_{j_m}$ the removal of these variables induces a generic TMDTO attack with the currently best known trade-off curve. The trade-off curve is given by $T^2M^2D^2 = (KV)^2$, where T, M, D denote respectively the time, memory and data complexity while K and V denote the key and IV space. This is a significant improvement over the currently best known trade-off $TM^2D^2 = (KV)^2$ of Dunkelman and Keller [12].

In difference to standard cube attacks our approach does not put any further conditions

on the degree and the complexity of the function $g(IV^*, K^*)$ which may be arbitrary complex function in the remaining key and IV variables. On the other hand, our method needs a certain justification that the scenario described above is realistic. While the existence of such polynomials (and their identification) in the state bit expressions after running the KSA may be highly unlikely, many ciphers initially load the key and IV bits into the state in a linear manner. This state is called *presetup state*, and is further processed through KSA to yield the initial state of the cipher. For instance, the presetup state for TRIVIUM, SNOW 2.0 [15], GRAIN-128 [20] and many other ciphers, is formed by loading the key and IV bits in a specific linear manner. We demonstrate that the linear relations of the form $\prod_{l=1}^{m}(IV_{i_l}+k_{j_l})$ are indeed present in a real-life ciphers such as TRIVIUM. TRIVIUM actually generates 66 such relations of the form (IV_i+k_j) in the first 66 steps of the KSA procedure (in a consecutive manner). Then, gradually, more complex linear relations are built (containing more terms) due to its simple state update mechanism that, apart from taking some state bits in a linear manner, takes two consecutive terms to build a quadratic term out of these. Therefore, quite likely the state bits of TRIVIUM may contain the expressions of the form $d(IV, K) = \prod_{l=1}^{m}(IV_{i_l} + k_{j_l})g(IV^*, K^*)$.

Even though, in the case of TRIVIUM, we may only suspect the existence of differentiable polynomials in the keystream bits satisfying the above mentioned conditions, there is no guarantee that the linear relations generated in the early phase of the KSA are later preserved. Therefore, a simple algorithm for the purpose of testing the unknown keystream expressions for the existence of these differentiable polynomials is devised. Notice also that when in particular TRIVIUM is considered our observation may largely reduce the search for suitable IV cubes. Indeed, if for instance moderate cube sizes are tested, say cubes of size 15 for instance, since we are only checking the cubes with consecutive indices we would just check some 65 cubes (assuming 80 cube indices). In the case of the exhaustive search $\binom{80}{15} = 2^{52.5}$ cubes need to be checked which is infeasible, and definitely unfavourable compared to our method. Our algorithm has proved successful in identifying the existence of differentiable polynomials for the reduced round TRIVIUM of up to 710 rounds. Nevertheless, due to utterly increased complexity and the form of the state bit equations other differentiable polynomials (which are nonconsecutive with respect to the key and IV indices in the linear product) are also discovered.

In a more restrictive model of related-key attack, using the assumption that a subset of the key variables and all the IV variables are kept fixed while the remaining key variables are varied over all possible values, we show that the key differentiation leads to a time/related-key trade-off (TRKTO) curve rather than to a standard time-memory trade-off (TMTO) curve [4, 19, 9, 22]. Instead of the relation $T \cdot M = N$ characteristic for a TMTO curve, the relation $T \cdot RK = K$, involving the space of related keys RK and the key space K, is approximately satisfied. The parameters T, M and N stands respectively for time, memory complexity and the state size. The motivation for studying this attack model, apart from a theoretical contribution and the fact that related-key attacks are considered as a valid attack scenario, may in certain settings also be justified from the application point of view. Namely, a closer inspection of the expression $d(IV, K) = \prod_{l=1}^{m}(IV_{i_l} + k_{j_l})g(IV^*, K^*)$, as a part of a much larger keystream bit expression $s(IV, K)$ (so $d(IV, K) \in s(IV, K)$), indicates that essentially

the differentiation over a key cube (varying some key variables over all possible values) gives identical result as the differentiation over the corresponding IV cube. That is, assuming that the remaining terms in $s(IV, K)$ does not cover the above IV cube we get $\bigoplus_{IV_{i_1},\ldots,IV_{i_l}} b(x) = \bigoplus_{k_{j_1},\ldots,k_{j_l}} b(x) = g(IV^*, K^*)$. This means that under the conditions described previously the two models of attack are equivalent in the sense that in both cases the result may be a complete elimination of a certain subset of key variables. The existence of differentiable polynomials then essentially leads to a generic TMTDO attack which is mounted in a chosen IV model, and the resulting trade-off curve of our TMDTO attack is given by $T^2M^2D^2 = (KV)^2$, where T, M, D denote respectively the time, memory and data complexity while K and V denote the key and IV space. This is a significant improvement over the currently best known trade-off $TM^2D^2 = (KV)^2$ of Dunkelman and Keller [12].

The paper is organized as follows. In Section 2 the main differential attack scenarios are discussed. It is then shown that the key differentiation in general eliminates the presence of the key variables the differentiation is performed on. The notion of differentiable polynomials is introduced in Section 3, and furthermore an algorithm for testing their existence is given. A generic related-key attack based on the method of key differentiation is described in Section 4, and its corresponding IV model is also discussed here. Furthermore, a comparison to cube testers and chosen IV statistical analysis is given in this section. In Section 5, we derive new TMDTO curves based on the key differentiation method and compare our curves with the currently best known ones. The concluding remarks are given in Section 6.

2 Key and IV differentiation attack models

In this section we give a brief overview of different attack scenarios in connection to their algebraic representations. In particular, the key differentiation method in a related-key model of attack is introduced.

2.1 Algebraic representation of the initialization process

Technically, given a stream cipher that allows an exact algebraic description, the cipher is fully specified by the key/IV setup algorithm (KSA) and encryption algorithm whose input parameters includes the key, IV, and state size.

Prior to the execution of KSA the key and IV bits are initially loaded into the state of the cipher, thus so-called *presetup state* $S^{ps} = (s_0^{ps}, s_1^{ps}, \ldots, s_{L-1}^{ps})$ is created. The KSA procedure then processes these bits in an iterative and nonlinear manner. The result of this phase is a so-called *initial state* $S = (s_0, s_1, \ldots, s_{L-1})$ of the cipher. For simplicity, we assume the key and IV vector are of the same length κ, and the size of the state $L \geq 2\kappa$ is chosen to be at least twice the key length to withstand time-memory-data trade-off attacks, see for instance [9, 22]. Now given a fixed key K and a publicly known $IV^{(i)}$ the state bits $s_r^{(i)}$, $r = 0, \ldots, L-1$ can be expressed as,

$$s_r^{(i)} = f_r(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) = g_r(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) + h_r(k_1, \ldots, k_\kappa) \tag{1}$$

Let $\mathcal{I}, \mathcal{J} = \{1, 2, \ldots, \kappa\}$ denote the set of indices for the IV vector respectively the key bits. Then another convenient algebraic representation, used in [28], is to represent the secret bits as a collection of terms as follows,

$$\begin{aligned} s_r^{(i)} &= \bigoplus_{I=\emptyset}^{\{1,2,\ldots,\kappa\}} a_{r,I} IV_I^{(i)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right] = \\ &= \underbrace{\bigoplus_{I\subset\mathcal{I};I\neq\emptyset} a_{r,I} IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right]}_{g_r^{(i)}} \oplus \underbrace{[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J]}_{h_r}, \quad \text{for } 0 \le r \le L-1, \end{aligned} \tag{2}$$

where $a_{r,I}, b_{r,I,J}$ are algorithm specific binary coefficients, and for a given subset $I = \{i_1, \ldots, i_l\} \subset \mathcal{I}$ the term IV_I denotes $IV_I = IV_{i_1} IV_{i_2} \cdots IV_{i_l}$ and similarly $K_J = k_{j_1} k_{j_2} \cdots k_{j_p}$ for some $J = \{j_1, \ldots, j_p\} \subset \mathcal{J}$. We note that treating both the key and $IV^{(i)}$ as variables the coefficients $a_{r,I}, b_{r,I,J}$ do not depend on the specific choice of IV's.

Given an algebraic description of the state of the cipher after running the KSA, the attacker can try to reduce the complexities of these relations by either applying the chosen IV attack, related key attack or alternatively to combine these two. Though similar in their essence, the cryptanalytic aspects significantly differ depending in which scenario the attack is performed.

2.2 Chosen IV attacks

A chosen IV attack is a very realistic scenario in which the attacker chooses a certain subset of all possible IV's in order to combine the relations of resulting state bits and to hopefully derive simple relations between the state bits and secret key bits. Assume that for a fixed key, the attacker is able to trace a certain subset of IV'bits say I^* for which relatively simple relations may be derived. The easiest way to understand this approach is to fix some portion of IV bits to zero in all positions but those that correspond to some $I^* = \{i_1, i_2, \ldots, i_m\}$ that we vary over all its possible values. Thus, given such an IV and its subset of indices I^* such that $IV_i = 0$ if $i \notin I^*$, any secret state bit s_r, $r = 0, \ldots, L-1$, can be expressed as,

$$\begin{aligned} s_r &= \bigoplus_{\xi\subset I^{(l)}} a_{r,\xi} IV_\xi^{(l)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,\xi,J} K_J\right] \oplus [\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J] = IV_{i_1}^{(l)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,i_1,J} K_J\right] \oplus \\ &\oplus \quad IV_{i_2}^{(l)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,i_2,J} K_J\right] \oplus \cdots \oplus IV_{i_1}^{(l)} IV_{i_2}^{(l)} \cdots IV_{i_m}^{(l)} [\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I^*,J} K_J] \oplus [\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J]. \end{aligned} \tag{3}$$

Then summing over all possible values of the IV cube specified by the indices from I^*, i.e., adding together the state bit expressions for all possible values of the IV variables $IV_{i_1}, \ldots, IV_{i_m}$ the attacker obtains,

$$\bigoplus_{I=\emptyset}^{I^*} s_r = \oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I^*,J} K_J = g(k_1, \ldots, k_\kappa).$$

Nevertheless, the same approach is valid in general if the differentiation is performed over a sufficiently large IV cube such that this cube is not covered by any other IV cube, which

essentially means that the KSA of the considered cipher has additional property that the coefficients $a_{r,I} = 0$ in (2) for any I such that $I^* \preccurlyeq I$. Here, we have used the covering relation for monomials IV_{I^*} and IV_I, where $IV_{I^*} \preccurlyeq IV_I$ means that $I^* = \{i_1, \ldots, i_m\} \subset I = \{i_1, \ldots, i_p\}$. This is exactly the idea that has been used in [28], where an semi-exhaustive search was performed over the suitable set of indices I^* of length (weight) 6 to derive linear expressions relating keystream bits of a reduced KSA TRIVIUM [10] and the secret key bits. This means that for the particular IV cubes found in [28] the function $g(k_1, \ldots, k_\kappa)$ was extremely simple (essentially a linear function in the secret key bits) which then allowed an efficient attack on a reduced round TRIVIUM. This framework has been later generalized in [11] and [1], where the same idea was presented in a wider scope; the attack is known as "cube attack".

2.3 Related-key attacks

In a related-key model the attacker can observe the operation of a cipher under several different keys whose values are initially unknown, but where some mathematical relationship connecting the keys is known to the attacker. In connection to the chosen IV model let now the set $J = \{j_1, \ldots, j_p\}$ denote the set of positions over which the key variables are varied, while the remaining key variables are kept fixed. Then given some fixed IV vector the equation (2) can be rewritten as follows,

$$s_r^{(i)} = \bigoplus_{M=\emptyset}^{\{1,2,\ldots,\kappa\}} b_{r,I} K_M = K_J g(k_{j_{p+1}}, \ldots, k_{j_\kappa}) + \bigoplus_{I;J \not\preccurlyeq I} b_{r,I} K_I, \quad r = 0, \ldots, L-1, \quad (4)$$

where g does not depend on the key variables with indices from J. Then, similarly to the chosen IV case, varying the key variables $k_{j_1}, \ldots, k_{j_p}$ over all possible values gives,

$$\bigoplus_{(k_{j_1}, \ldots, k_{j_p}) \in \{0,1\}^p} \left(K_J g(k_{j_{p+1}}, \ldots, k_{j_\kappa}) + \bigoplus_{I;J \not\preccurlyeq I} b_{r,I} K_I \right) = g(k_{j_{p+1}}, \ldots, k_{j_\kappa}).$$

The cancellation of the remaining terms in the first sum is due to the fact that K_J equals to zero for all $(k_{j_1}, \ldots, k_{j_p}) \in \{0,1\}^p$ but for $(k_{j_1}, \ldots, k_{j_p}) = (1, \ldots, 1)$. On the other hand, in the second sum due to the condition $J \not\preccurlyeq I$ there will always be some variable k_{j_l} such that $j_l \in I$ but $j_l \notin J$, which in turn implies the cancellation of such a monomial when the summation is performed over all possible values of $(k_{j_1}, \ldots, k_{j_p}) \in \{0,1\}^p$.

Example 1 *Let the secret state bit s_r be evaluated as $s_r = k_1k_3k_4k_5 \oplus k_1k_3k_6 \oplus k_2k_5k_6 \oplus k_2k_6 \oplus k_3 \oplus k_1$; after the IV value has been specified. Then differentiating w.r.t. k_1 and k_3 we have,*

$$\sum_{(k_1,k_3) \in \{0,1\}^2} s_r = k_4k_5 + k_6.$$

Note that k_2 is also eliminated but only due to the particular form of s_r.

It is important to notice that the differentiation over a key cube always removes the key variables over which the differentiation is performed from any state bit polynomial which is not the case in a chosen IV model when the differentiation is performed over an IV cube.

3 Differentiable polynomials

The two attack models discussed in the previous section both have certain limitations. In a related-key model the key differentiation method is always possible which inevitably leads to a certain reduction of the key variables, but the attack assumptions are rather restrictive. On the other hand, in a chosen IV model the current approaches such as the cube attack may lead to finding low degree equations in key variables (even linear) but there is no guarantee that such equations really exist.

In what follows we introduce a notion of *differentiable polynomials* that will help us to treat the state polynomials in an efficient way. For convenience, if some function that generally depends on all the key and IV variables, which is denoted by $g(IV, K)$, does not depend on some subset of the key and IV variables with indices from J^* and I^*, respectively, we write $g(K \setminus K_{I^*}, IV \setminus IV_{J^*})$. For simplicity, let us assume that some state bit of the cipher is given by,

$$\begin{aligned} s_r &= (k_{j_1} + IV_{i_1})(k_{j_2} + IV_{i_2}) \cdots (k_{j_m} + IV_{i_m}) g(k_1, \ldots, k_\kappa, IV_1, \ldots, IV_\kappa) \\ &= \prod_{i \in I^*, j \in J^*} (k_j + IV_i) g(K \setminus K_{I^*}, IV \setminus IV_{J^*}) = p_{J^*,I^*} g(K \setminus K_{I^*}, IV \setminus IV_{J^*}), \quad (5) \end{aligned}$$

where $I^* = \{i_1, \ldots, i_m\} \subset \mathcal{I}$ and $J^* = \{j_1, \ldots, j_m\} \subset \mathcal{J}$.

Definition 1 *A polynomial $d(K, IV) = p_{J,I} g(K \setminus K_{I^*}, IV \setminus IV_{J^*})$ with the separation of the key and IV variables is called* differentiable *with respect to the key variables $k_{j_1}, \ldots, k_{j_m}$ and the IV variables $IV_{i_1}, \ldots, IV_{i_m}$.*

Notice that the above polynomial $d(K, IV)$ may be differentiated either over the IV cube with indices from I^* or over the key cube with indices from J^*, that is,

$$\bigoplus_{I=\emptyset}^{I^*} d(K, IV) = \bigoplus_{J=\emptyset}^{J^*} d(K, IV) = g(K \setminus K_{I^*}, IV \setminus IV_{J^*}).$$

A major difference between the two approaches that the key variables $k_{j_1}, \ldots, k_{j_m}$ will vanish after the differentiation over the key cube even though g does depend on these variables, that is, if g is of the form $g(K, IV \setminus IV_{J^*})$, cf. Section 2.3. The next example illustrates the case when the differentiation over an IV cube does not necessarily imply the removal of the key variables. Such a class of polynomials is called *non-differentiable polynomials* with respect to some subset of key variables.

Example 2 *Let $s_r = k_1(k_1 \oplus IV_1)k_2(k_2 \oplus IV_2)g(k_3, \ldots, k_\kappa, IV_3, \ldots, IV_\kappa)$. Then clearly the differentiation with respect to IV_1 and IV_2 cannot eliminate the presence of k_1, k_2 as,*

$$\bigoplus_{IV_1, IV_2} s_r = k_1 k_2 g(k_3, \ldots, k_\kappa, IV_3, \ldots, IV_\kappa).$$

On the other hand, if $s_r = (k_1 \oplus IV_1)(k_2 \oplus IV_2)g(k_3, \ldots, k_\kappa, IV_3, \ldots, IV_\kappa)$ then,

$$\bigoplus_{IV_1, IV_2} s_r = g(k_3, \ldots, k_\kappa, IV_3, \ldots, IV_\kappa).$$

Thus, we can only eliminate those key variables in p_{J^*,I^*} that do not appear in the expressions of the form $k_j(k_j+IV_i)$. This is the reason why the effect of the differentiation of the above polynomial $d(IV,K)$ was the elimination of the considered key variables regardless of the scenario of attack.

It is completely unrealistic to expect that the state bit expressions are of the form as $d(IV,K)$ is, no matter how complex the function $g(K \setminus K_{I^*}, IV \setminus IV_{J^*})$ might be. Nevertheless, it might be the case that $d(IV,K) \in s_r$ and the remaining terms are such that for any IV cube that covers IV^* ($IV^* \preccurlyeq IV$) the coefficient $a_{r,I}=0$ in (2) so that the terms (polynomials) $a_{r,I}IV_I\left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J}K_J\right]$ are not present in the symbolic expression of s_r for any I such that $IV^* \preccurlyeq IV$. This may be justified by the fact that the state update algorithm is not sufficiently fast and nonlinear to ensure that the symbolic expressions of the state bits behave as randomly generated functions in the key and IV variables. This is also the main reason for the weakness of reduced round TRIVIUM where the differentiation over relatively small sized IV cubes (of length 12) gave a rise to linear equations. Therefore, if the attacker has some a priori knowledge about the structure of the cipher (s)he might guess that some differentiable polynomials of the above form constitute a part of the state bit (keystream) polynomials.

We now generalize our approach to consider arbitrary complex polynomials using the above representation. A standard representation given by (1) and (2) was obtained by extracting the IV monomials and the corresponding polynomials in the key bits. Instead, we can extract a suitable maxterm of the form $p_{J^*,I^*}=\prod_{i\in I^*, j\in J^*}(k_j+IV_i)$ and the corresponding polynomial $g(K\setminus K_{I^*}, IV\setminus IV_{J^*})$. That is, let us assume that the output keystream bit z^t can be written as,

$$z^t = p_{J^*,I^*}g(K\setminus K_{I^*}, IV\setminus IV_{J^*}) \bigoplus_{I=\emptyset}^{\{1,2,\ldots,\kappa\}} a_{r,I}IV_I^{(i)}\left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J}K_J\right], \;\; a_{r,I}=0 \text{ if } I^* \preccurlyeq I. \quad (6)$$

The term $p_{J^*,I^*}=\prod_{i\in I, j\in J}(k_j+IV_i)$ is called a maxterm with respect to variables from I as $a_{r,I}=0$ for all $I^* \preccurlyeq I$. Then, we have

$$\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(K\setminus K_{I^*}, IV\setminus IV_{J^*}).$$

This means that most of the terms are cancelled leaving out only the function g that does not depend on the key variables with indices from J^*. If for instance g is affine function in the key and IV bits this gives one linear equation, and the whole procedure is repeated using some other maxterms (including other key and IV variables). A special case arises when $g(K\setminus K_{I^*}, IV\setminus IV_{J^*})=1$ since in this case we cannot perform the differentiation w.r.t. all the IV variables as we do not get any dependency on the key variables. Thus, in this specific case we may for instance compute,

$$\bigoplus_{IV_{i_2},\ldots,IV_{i_m}} z^t = k_{j_l}+IV_{i_l},$$

where apart from the above assumption we also assume that the coefficient $a_{r,I\setminus i_l}=0$.

Example 3 *Let the state bit s_r of the cipher be given by,*

$$\begin{aligned} s_r &= (k_1 \oplus IV_1)(k_2 \oplus IV_2)(k_3 \oplus IV_3)g_1(k_4, \ldots, k_\kappa, IV_4, \ldots, IV_\kappa) + \\ &+ (k_1 \oplus IV_1)(k_2 \oplus IV_2)g_2(k_4, \ldots, k_\kappa, IV_4, \ldots, IV_\kappa) + \\ &+ (k_1 \oplus IV_1)(k_4 \oplus IV_4)g_3(k_5, \ldots, k_\kappa, IV_5, \ldots, IV_\kappa) \end{aligned}$$

The differentiation with respect to the maxterm $p_{J,I} = (k_1 \oplus IV_1)(k_2 \oplus IV_2)(k_3 \oplus IV_3)$ yields,

$$\bigoplus_{IV_j; j \in J} s_r = g_1(k_4, \ldots, k_\kappa, IV_4, \ldots, IV_\kappa),$$

where $J = \{1, 2, 3\}$.

3.1 Differentiable polynomials of TRIVIUM

In this section we demonstrate that our description of a cipher, having a specific structure as previously described, is justified for some real-life algorithms as the case of TRIVIUM confirms. It might be the case that some other ciphers that mix the key and IV variables in a linear manner also admit such a description though we do not pursue this issue further.

TRIVIUM is a hardware oriented stream cipher that reached the final third phase of the eSTREAM project. It uses three LSFRs whose total length is 288, thus the state of the cipher is given by $S = (s_1, s_2, \ldots, s_{288})$. The state is contained in the three LFSRs so that the first register comprises the state bits $s_1, \ldots, s_{93}$, the second register stores $s_{94}, \ldots, s_{177}$, and $s_{178}, \ldots, s_{288}$ are kept in the third register. In the output mode the following computations are performed:

– Define $t_1 = s_{66} \oplus s_{93}$, $t_2 = s_{162} \oplus s_{177}$, and $t_3 = s_{243} \oplus s_{288}$.
– The output keystream is formed as $z = t_1 \oplus t_2 \oplus t_3$.
– Update the t_i's according to

$$t_1 = t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}; \quad t_2 = t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}; \quad t_3 = t_3 \oplus s_{286} \cdot s_{287} \oplus s_{69};$$

– Finally the registers are updated as follows:

$$(s_1, \ldots, s_{93}) = (t_3, s_1, \ldots, s_{92}); (s_{94}, \ldots, s_{177}) = (t_1, s_{94}, \ldots, s_{176});$$

$$(s_{178}, \ldots, s_{288}) = (t_2, s_{178}, \ldots, s_{287});$$

The initialization loads the 80-bit key to $s_1, \ldots, s_{80}$, and the 80-bit IV vector to the positions $s_{94}, \ldots, s_{173}$. The remaining state bits are filled with zeros apart from s_{286}, s_{287} and s_{288} which are set to one; this way the presetup state S^{ps} is completed. The key/IV setup is performed by clocking the cipher $4 \cdot 288 = 1152$ times without producing the output.

Though, there is no linear mixing in the presetup state, it can be easily verified that after running the key/IV setup a certain number of clocks the internal state will contain desired equations that mix the IV and key bits in a linear manner. For convenience, let us denote the state at time i by $S^i = (s_1^{(i)}, \ldots, s_{288}^{(i)})$, where $S^{(0)} = S^{ps}$. Then after the

very first clock the update variables are computed as $t_1 = k_{66} \oplus IV_{78}$, $t_2 = IV_{69} \oplus IV_{80}$, and $t_3 = k_{69}$. Hence, $s_{94}^{(1)} = k_{66} \oplus IV_{78}$ and thereby one linear relation is obtained. It is straightforward to check that the KSA of TRIVIUM admits 12 linear relations after the first 12 clocks,

$$s_{94}^{(i)} = k_{67-i} \oplus IV_{79-i}, \quad i = 1, \ldots, 12,$$

which leads to the following internal state after 12 clocks:

$$(s_1^{(12)}, \ldots, s_{66}^{(12)}, k_{55}, \ldots, k_{66}, s_{79}^{(12)}, \ldots, s_{93}^{(12)}, k_{55} \oplus IV_{67}, \ldots, k_{66} \oplus IV_{78}, s_{106}^{(12)}, \ldots, s_{288}^{(12)}).$$

After these 12 immediate linear relations among the key and IV bits, the state bit s_{94} at $t = 13$ is given by,

$$s_{94}^{(13)} = (k_{54} \oplus IV_{66}) \oplus k_{79}k_{80}.$$

Then, during the further state updates the following recursion is valid,

$$s_{94}^{(13+i)} = (k_{54-i} \oplus IV_{66-i}) \oplus k_{79-i}k_{80-i} \oplus k_{81-i}, \quad i = 1, \ldots, 53.$$

Thus in total there are 66 linear relations among the key and IV bits $k_{67-i} \oplus IV_{79-i}$, for $i = 1, \ldots, 66$. It is important to notice that the linear relations in TRIVIUM of the form $k_{67-i} \oplus IV_{79-i}$, $i = 1, \ldots, 66$, are consecutive due to the selection of the update positions (using two consecutive bits as a quadratic update term). Thus, due to the update algorithm of TRIVIUM, the state bits of a reduced round TRIVIUM will contain differentiable expressions of the form

$$\underbrace{(k_{66-i-j} \oplus IV_{78-i-j})(k_{66-i+1-j} \oplus IV_{78-i+1-j}) \cdots (k_{66-j} \oplus IV_{78-j})}_{K_J}, \tag{7}$$

as their subterm, where $j \in [0, 54]$ and $i + 1$ is then length of the IV cube. For instance, if only 288 rounds of the KSA are used then we were able to perform differentiation over the IV cubes of length four and to retrieve the key bits in the same manner as it was done in [28, 11]. Moreover, the computer simulations performed on reduced round TRIVIUM indicate that differentiable polynomials in the above form are present in the state bit expressions if the KSA is run up to 700 rounds.

Note that the above linear relations are only a part of in general huge symbolic expressions that describe a particular state/output keystream bit. Furthermore, these linear relations may also be multiplied with some polynomials $g(IV, K)$ which may or may not depend on a certain portion of the key bits. We cannot trace these huge symbolic expressions unless the number of rounds is relatively small, for instance if the initialization process only takes some 200-300 rounds. Therefore, we need some kind of heuristic algorithm to test for the presence of differentiable polynomials.

3.2 An algorithm for the determination of the existence of differentiable polynomials

Assume that our targeted cube is specified by the indices from $I^* = \{i_1, \ldots, i_m\}$, thus the differentiation is performed over all possible values of $IV_{i_1}, \ldots, IV_{i_m}$. Now, if z^t is given

by (6) the result of differentiation is $g(K \setminus K^*, IV \setminus IV^*)$ which does not depend on the key and IV variables with indices from J^* and I^*, respectively. More precisely, for fixed $\alpha \in IV \setminus IV^*$ and $\beta \in K \setminus K^*$, we have

$$\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha) = \text{const.} \ \forall \gamma \in K^*.$$

Moreover, there are three cases that need to be considered.

(1) For a fixed $\alpha \in IV \setminus IV^*$ and some fixed value of the key variables from K^*, say $\gamma \in K^*$, we can randomly choose different $\beta \in K \setminus K^*$ and check the differentiated value, $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha)$. If the values of $g(\beta, \alpha)$ is changeable then it means that $deg(g) \geq 1$, i.e, function g depends only on the key variables of $K \setminus K^*$.

(2) For a fixed $\alpha \in IV \setminus IV^*$, if this constant value is equal to 0 (i.e, const=0) then it might be the case that $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = 0$ is identically zero which might indicate that the differentiation is performed over an IV cube that is too large. This also means that there is a possibility that z^t cannot be represented in the form given by (6). Nevertheless, this case can be checked further by taking different $\alpha \in IV \setminus IV^*$. If the differentiated value $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha), (\beta \in K \setminus K^*)$ still remains zero then with a very high probability the output keystream bit does not contain a differentiable polynomial in the above form with respect to the IV cube specified by the indices from I^*.

(3) For a fixed $\alpha \in IV \setminus IV^*$, we select some random different $\beta \in K \setminus K^*$ and check $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha)$. If the differentiated value $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha) = 1$, then with a high probability we have $g = 1$.
The algorithm can be described by the following steps:

INPUT : A targeted IV cube with indices from I^*, the corresponding key cube with indices from J^*, and a cipher to be tested.

1. Select some fixed $\alpha \in IV \setminus IV^*$ and randomly chosen different $\beta \in K \setminus K^*$ and $\gamma \in K^*$.

2. For each given $\alpha \in IV \setminus IV^*$ and $\beta \in K \setminus K^*$, compute $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \gamma, \alpha)$ by varying many different $\gamma \in K^*$ and if $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \gamma, \alpha) \neq$ const go back to (1) and choose another $\alpha \in IV \setminus IV^*$ and $\beta \in K \setminus K^*$. If $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \gamma, \alpha) =$ const for many different $\gamma \in K^*$, then perform the next step.

3. For each fixed $\alpha \in IV \setminus IV^*$, check $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha)$ by using different $\beta \in K \setminus K^*$. If the values of $g(\beta, \alpha)$ is changeable then report the *Existence of differentiable polynomial with* $\alpha \in IV \setminus IV^*$. If the values of $g(\beta, \alpha)$ is constant, then perform the next step.

4. If $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha) = 0$, then report the *Nonexistence of differentiable polynomial w.r.t.* $\alpha \in IV \setminus IV^*$. On contrary, if $\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t = g(\beta, \alpha) = 1$, then report the *Existence of differentiable polynomial w.r.t.* $\alpha \in IV \setminus IV^*$ *and* $g = 1$.

Remark:

(1) In Step 2, to check that $g(K \setminus K^*, IV \setminus IV^*)$ does not depend on the key variables with indices from J^*, we may also choose a method similar to Probabilistic Neutral Bits (PNB) method introduced in [2].

(2) In the case that $g(K \setminus K_{J^*}, IV \setminus IV_{I^*}) = 1$, i.e, the function g itself is a constant function and therefore it cannot be used to distinguish the correct subkey in our TMTO attack. Nevertheless, this also indicates that the differentiation should be performed over a smaller cube.

We give a few computer simulated examples below concerning the cubes with consecutive indices of the TRIVIUM cipher. Notice, that we are not restricted to consider only the cubes with consecutive indices (cf. Example 6) as our algorithm works for any choice of the indices.

Example 4 *For a reduced round TRIVIUM, let the targeted cube be* $\{IV_{67}, \ldots, IV_{78}\}$ *(12 bits in total), and let the remaining IV bits be given by* $IV_i = 1$, *for* $i \equiv 0 \pmod 3$, *otherwise* $IV_i = 0$. *Let also the corresponding key cube be given by* $\{k_{55}, \ldots, k_{66}\}$, *i.e.,* $k_{67-i} \oplus IV_{79-i}, i = (1, \ldots, 12)$. *Then computer simulations show that the first bit of the output after 582 rounds of the initalization contains a differentiable polynomial.*

Example 5 *Similarly, if the targeted cube is* $\{IV_{54}, \ldots, IV_{76}\}$ *(23 bits in total) and let the corresponding key cube be* $\{k_{42}, \ldots, k_{64}\}$, *i.e.,* $k_{67-i} \oplus IV_{79-i}$, *for* $i = 3, \ldots, 25$.. *Then the existence of differentiable polynomials could be confirmed for TRIVIUM reduced to 632 rounds.*

Example 6 *For TRIVIUM reduced to 710 rounds it seems that the cubes with consecutive indices are not necessarily preferable. Let*

$$I^* = \{15, 19, 20, 22, 25, 28, 30, 38, 41, 47, 49, 51, 56, 58, 61, 65, 66, 68, 69, 70, 72, 76, 79\},$$

where $\#I^* = 23$, *and let the remaining bits of the initial vector be set to one* ($IV_i = 1$). *Let also the corresponding key cube be related to the IV bits through the relation* $k_j = IV_{j-12}$ *so that* $J^* = \{i - 12 : i \in I^*\}$. *Our simulations show that there are IV cubes of length 22 (and the corresponding key cubes) such that we could identify the existence of differentiable polynomials of the form*

$$\prod_{i=1}^{22}(k_{67-j} \oplus IV_{79-j}) \times g(K \setminus K_{I^*}, IV \setminus IV_{J^*},$$

where (k_i, IV_j) *are pairs satisfying the relation of* $k_{67-j} \oplus IV_{79-j}, j \in \{3, \ldots, 66\}$.*) Then the first bit of the output of 710 reduced round TRIVIUM show the existence of differentiable polynomials.*

4 Generic key differentiation attacks in a related-key model

The main idea of our approach is that the key differentiation allows for an efficient removal of a certain subset of key variables from the state bit expressions. Under a reasonable

assumption that the key differentiation may only be performed over a portion of the key variables, our approach also utilizes some ideas of TMTO attacks so that the tables storing the differentiated expressions are created in the precomputation phase. In brief, for any possible value of a subset of the key variables of cardinality $2^{\kappa-m}$ over which the key differentiation is not performed a small symbolic expression in a few key variables is stored as one entry of the table. The correct values of these key bits are identified by comparing the evaluations of the differentials stored in the table during the precomputation phase to the differentials evaluated via the observed keystreams. This attack is only of theoretical significance, as it will be shown later that its complexity roughly corresponds to an attack based on the birthday paradox of Biham [6], but since the details of the attack are essentially the same as in the case the attack has been performed in a chosen IV model we give a full description.

Let the secret state bits of the cipher after running the KSA be given by $s_i^0 = f_i(IV_1, \ldots, IV_\kappa, k_1, \ldots, k_\kappa)$, $i = 0, \ldots, L-1$. Also for $t \geq 0$ let the new state be updated through $(s_0^{t+1}, \ldots, s_{L-1}^{t+1}) \stackrel{S}{\leftarrow} (s_0^t, \ldots, s_{L-1}^t)$, and the output at time t is computed as $z^t = h^t(s_0^t, \ldots, s_{L-1}^t)$. For convenience, for $J^* = \{k_{i_1}, \ldots, k_{i_m}\} \subset K$ we introduce the following notation $\Delta^{J \subset J^*} z^t = z^t(IV, K) + z^t_{k_i \in J: 1+k_i \leftarrow k_i}(IV, K)$. That is, we perform a formal key differentiation by considering the modulo two sum of the original keystream expression and the expression obtained by replacing the key variables from the set J by their inverted values $k_i + 1$. To reduce the computational complexity in the precomputation phase a key guessing method prior to differentiation is deployed. The attack proceeds as follows.

Off-line precomputation phase

1. Let $G = \{j_1, j_2, \ldots, j_{\kappa-m}\} \subset \{1, 2, \ldots, \kappa\}$ be a set of indices for which we evaluate the s_i's for all $2^{\kappa-m}$ possible choices of the key variables $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$. Thus, given an IV vector the attacker may compute symbolic expressions for the state bit equations $s_i^t = f_i^t(IV, K)$ to obtain $s_i^t = f_i^t(k_{j_{\kappa-m+1}}, \ldots, k_{j_\kappa})$, $i = 0, \ldots, L-1$, $t = 0, \ldots \kappa$ for each possible value of $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$. This way $2^{\kappa-m} \times L$ symbolic state bit expressions $s_i^t = f_i^t(k_{j_{\kappa-m+1}}, \ldots, k_{j_\kappa})$ are **temporarily** stored in a table for each t. Each of these expressions contains no more than 2^m terms (the expressions depend on the m key variables).

2. The attacker then performs the differentiation of the output keystream expressions $z^t = h^t(s_1^t, \ldots, s_{L-1}^t) = \tilde{h}^t(k_{j_{\kappa-m+1}}, \ldots, k_{j_\kappa})$ with respect to some subset $D = \{k_{j_{\kappa-m+1}}, \ldots, k_{j_{\kappa-c}}\}$ of the key bits of cardinality $\#D = m - c$, where c is a small positive integer, e.g. $c = 1$ or $c = 2$. The resulting differentials

$$\bigoplus_{J \subset D; J \neq \emptyset} \Delta^J z^t = \tilde{h}^{t,D}(k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}), \ldots, \bigoplus_{J \subset D; J \neq \emptyset} \Delta^J z^{t+d+r} = \tilde{h}^{t+d+r,D}(k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}),$$

only depend on c key variables, and choosing $d = 2^c$ ensures that the above system of d nonlinear equations of degree at most c can be solved using for instance linearization (introducing new variables for the nonlinear terms) once the differentials $\bigoplus_{J \subset D; J \neq \emptyset} \Delta^J z^t, \ldots, \bigoplus_{J \subset D; J \neq \emptyset} \Delta^J z^{t+d}$ have been computed from the observed

keystream sequences. The whole procedure can be repeated for the other subsets of the key bits $D' = \{k_{j'_{\kappa-m+1}}, \ldots, k_{j'_{\kappa-c}}\}$ so that the nonlinear expressions in c key variables are disjoint with the previous expressions (different key variables are chosen).

The computation of the above differentials is performed for each guessed value of $k_{j_1}, \ldots, k_{j_{\kappa-m}}$ and the functions $\tilde{h}^{t,D}, \ldots, \tilde{h}^{t+d+r,D}$ are stored in a table T which is later used in the on-line phase of the attack.

The additional r differential values $\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d+1}, \ldots, \bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d+r}$ are stored for the purpose of identifying the right key variables $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$, see Step 3. in the on-line phase below. The total storage required is therefore of order $\mathcal{O}((r+d)2^{\kappa-m}) \approx \mathcal{O}(2^{\kappa-m})$, and the time complexity is dominated by the computation of differentials $\mathcal{O}(2^{\kappa-m}2^m) = \mathcal{O}(2^\kappa)$.

On-line phase

1. The attacker observes the output keystream bits $z^{t,J}, \ldots, z^{t+d+r,J}$ (where $J \subset D = \{k_{j_{\kappa-m+1}}, \ldots, k_{j_{\kappa-c}}\}$) generated by a fixed and known IV and different 2^{m-c} keys (alternatively the key is fixed and 2^{m-c} chosen IV's are used if the transformation of the attack scenario is possible). These keys are derived form a single (master) key by varying the key variables $\{k_{j_{\kappa-m+1}}, \ldots, k_{j_{\kappa-c}}\}$ over all possible values. The data complexity is therefore 2^{m-c} related-key keystream sequences of length $d+r \approx 2^c+\kappa$, see below for the specification of r.

2. From the observed keystreams $z^{t,J}, \ldots, z^{t+d,J}, J \subset D$ the attacker can evaluate the differentials $\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^t_J, \ldots, \bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d}_J$ in time instances $t, t+1, \ldots, t+d$. The evaluation only uses the observed keystream bits and is performed once for all. The time complexity is of order $\mathcal{O}(d2^{m-c})$. For instance, at time t the attacker computes,

$$\begin{aligned}[\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^t]^{\text{ev.}} &= (z^t_{|k_{j_{\kappa-m+1}}} + z^t_{|k_{j_{\kappa-m+1}}+1}) + (z^t_{|k_{j_{\kappa-m+2}}} + z^t_{|k_{j_{\kappa-m+2}}+1}) + \cdots + \\ &+ (z^t_{|k_{j_{\kappa-m+1}},k_{j_{\kappa-m+2}},\ldots,k_{j_{\kappa-c}}} + z^t_{|k_{j_{\kappa-m+1}}+1,k_{j_{\kappa-m+2}}+1,\ldots,k_{j_{\kappa-c}}+1}).\end{aligned}$$

These values are used to solve $2^{\kappa-m}$ many systems of nonlinear equations of size 2^c that are stored in the table T,

$$[\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^t]^{\text{ev.}} = \tilde{h}^{t,D,G}(k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}), \ldots, [\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d}]^{\text{ev.}} = \tilde{h}^{t+d,D,G}(k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}),$$

thus retreiving $k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}$ for each guessed key value from G. The time complexity of this step is of order $\mathcal{O}(2^{3c}2^{\kappa-m})$, corresponding to solving $2^{\kappa-m}$ many systems of size 2^c.

3. To identify the correct key bits $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$ (and thereby the correct values of $k_{j_{\kappa-c+1}}, \ldots, k_\kappa$) the attacker compares the differentials of the true outputs

$$[\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d+1}]^{\text{ev.}}, \ldots, [\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d+r}]^{\text{ev.}}$$

to the differentials

$$\bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^t = \tilde{h}^{t+d+1,D,G}(k_{j_{\kappa-c+1}},\ldots,k_{j_\kappa}),\ldots,\ \bigoplus_{J\subset D;J\neq\emptyset} \Delta^J z^{t+d+r} = \tilde{h}^{t+d+r,D,G}(k_{j_{\kappa-c+1}},\ldots,k_{j_\kappa}).$$

The latter differentials are computed by evaluating the nonlinear equations stored in the table T for each guessed value $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$. Due to the computational complexity of Step 3. above, the parameter c is chosen so that $c \ll \kappa - m$. To extract the correct key bits $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$ and $k_{j_{\kappa-c+1}}, \ldots, k_{j_\kappa}$ the constant r must satisfy $r > \kappa - m - c$. The selection of such r then enables the extraction of a unique correct partial key. All other key candidates are expected not to pass the above differential test.

4. Once the correct values of $k_{j_1}, k_{j_2}, \ldots, k_{j_{\kappa-m}}$ and $k_{j_{\kappa-c+1}}, \ldots, k_\kappa$ have been retrieved, the remaining key bits $k_{j_{\kappa-m+1}}, \ldots, k_{j_{\kappa-c}}$ can be found by repeating the same procedure for different disjoint choices of these c key variables.

The complexity estimate is governed by the choice of the parameters m and c. Increasing m makes the attack more unrealistic as the attacker is supposed to observe 2^m sequences generated by different keys and the same IV. Both time and memory complexity in the off-line precomputation phase are dominated by computing and storing the differentials. The time complexity is of order $\mathcal{O}(2^\kappa)$ and the memory complexity of this phase is $\mathcal{O}(2^\kappa)$.

In the on-line phase the attacker performs $2^{3c}2^{\kappa-m} \times m/c$ computations, which corresponds to solving $2^{\kappa-m}$ many systems of nonlinear equations of size 2^c for m/c different disjoint subsets of the c key variables. This is the worst case complexity as these nonlinear equations may be of significantly smaller size than 2^c. Thus, the time complexity in the on-line phase is $\mathcal{O}(2^{3c}2^{\kappa-m})$, and the memory requirements are negligible. Still, the attacker in the on-line phase makes use of the table T of size $\mathcal{O}(2^c 2^{\kappa-m})$. The data complexity is $2^{m-c} \times m/c$ related-key keystream sequences of length $2^c + \kappa$.

We illustrate the whole idea with an example, considering a hypothetical toy cipher.

Example 7 *Let us consider a toy cipher that uses the secret key* $K = (k_1, k_2, k_3, k_4, k_5, k_6) = (0, 1, 1, 1, 0, 0)$ *and some known* IV*. Assume that the first three (secret) initial state bits are given by,*

$$\begin{aligned}
s_0 &= k_1k_2k_3 + k_3k_4 + k_3 + k_4 + k_2k_5 + k_1k_2k_6 + k_3k_5k_6,\\
s_1 &= k_1k_3k_4 + k_2 + k_4 + k_2k_6 + k_3k_5k_6 + k_1k_2k_5 + k_1k_2k_4k_6 + k_1k_2k_4k_5,\\
s_2 &= k_1k_2k_4 + k_2k_3k_4 + k_1 + k_2 + k_3 + k_2k_3k_5 + k_1k_4k_6 + k_4k_6 + k_1k_2k_3k_6,
\end{aligned}$$

and furthermore let $z_1 = s_0 + s_1$*,* $z_2 = s_0 + s_2$*, and* $z_2 = s_0 + s_1 + s_2$*. The attacker for each possible value of* k_5 *and* k_6 *(for a given and known* IV*) evaluates symbolic expressions for*

the differentials $\Delta^{k_1,k_2} z_1$, $\Delta^{k_1,k_2} z_2$, *and* $\Delta^{k_1,k_2} z_3$ *which is summarized in Table 1 below. The computation of these differentials is only given for the case* $(k_5, k_6) = (0, 0)$,

$\mathbf{z_1}$	$\mathbf{z_2}$
$\Delta^{k_1} z_1 = \Delta^{k_1}(s_0 + s_1) = k_2k_3 + k_3k_4$	$\Delta^{k_1} z_2 = \Delta^{k_1}(s_0 + s_2) = k_2k_3 + k_2k_4 + 1$
$\Delta^{k_2} z_1 = \Delta^{k_2}(s_0 + s_1) = k_1k_3 + 1$	$\Delta^{k_2} z_2 = k_1k_3 + k_3k_4 + k_1k_4 + 1$
$\Delta^{k_1,k_2} z_1 = k_3(1 + k_1 + k_2) + k_3k_4 + 1$	$\Delta^{k_1,k_2} z_2 = k_3(1 + k_1 + k_2) + k_3k_4 + k_4(1 + k_1 + k_2)$
$\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_1 = k_3$	$\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_2 = k_3 + k_4$

Also, one can compute $\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_3 = k_3 + k_4$. *For conciseness, we denote this differentials by* $\Delta^{sym} z_i$.

	$(k_5, k_6) = (0, 0)$	$(k_5, k_6) = (0, 1)$	$(k_5, k_6) = (1, 0)$	$(k_5, k_6) = (1, 1)$	$\Delta^J z_i^{ev.}$
$\Delta^{sym} z_1$	k_3	$k_3 + k_4 + 1$	$k_3 + k_4 + 1$	$k_3 + k_4$	$\Delta^J z_1^{ev.} = 1$
$\Delta^{sym} z_2$	$k_3 + k_4$	$k_4 + 1$	$k_3 + k_4$	$k_4 + 1$	$\Delta^J z_2^{ev.} = 0$
$\Delta^{sym} z_3$	$k_3 + k_4$	1	$k_3 + 1$	k_4	$\Delta^J z_3^{ev.} = 0$

Table 1: A table with symbolic expressions

The attacker observes the outputs z_1, z_2, z_3 *for the incremented values of the key bits* k_1 *and* k_2, *thus he is able to evaluate the actual differentials (the last column in the table). For the given values of* K *we may check that either* $(k_5, k_6) = (0, 0)$ *or* $(k_5, k_6) = (1, 0)$ *are the partial key candidates since the differentials in the first (or the third) column give a solvable set of equations. For instance, if* $(k_5, k_6) = (0, 0)$ *then,*

$$\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_1^{ev.} = (z_{1|k_1=0} + z_{1|k_1=1}) + (z_{1|k_2=1} + z_{1|k_2=0}) + (z_{1|(k_1,k_2)=(0,1)} + z_{1|(k_1,k_2)=(1,0)}) =$$
$$= (1 + 1) + (1 + 0) + (1 + 1) = 1 = k_3,$$

and similarly $\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_2^{ev.} = 0 = k_3 + k_4$, *and* $\bigoplus_{J\subset\{k_1,k_2\}} \Delta^J z_3^{ev.} = 0 = k_3 + k_4$. *This system also gives the correct values for the key bits* $k_3 = 1$ *and* $k_4 = 1$. *Note that e.g. the differentials in the second column exclude the value* $(k_5, k_6) = (0, 1)$ *since the differential w.r.t.* z_3 *is different than the evaluated one, and the same is true for the fourth column. To filter out the key candidate* $(k_5, k_6) = (1, 0)$ *a few more differentials need to be stored in the table and evaluated during the on-line phase.*
In this example the key bits k_1, k_2 *can then be either found by an exhaustive search or a similar procedure is repeated but now the differentiation is performed over* k_3 *and* k_4 .

4.1 Adopting our related-key algorithm in a chosen IV model

We now once again consider a suitable representation of the keystream polynomial in order to discuss the formalism of translating the generic key-related attack into a chosen IV attack. If the output bit z^t may be represented as,

$$z^t = p_{J^*,I^*} g(K \setminus K_{I^*}, IV \setminus IV_{J^*}) \bigoplus_{I=\emptyset}^{\{1,2,\ldots,\kappa\}} a_{r,I} \mathrm{IV}_I^{(i)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right], \quad a_{r,I} = 0 \text{ if } I^* \preccurlyeq I,$$

where $p_{J^*,I^*} = \prod_{l=1}^{m}(IV_{i_l} + k_{j_l})$, then we already remarked that it is irrelevant whether the differentiation is performed over the IV cube with indices from I^* or over the key cube with indices from J^*. This essentially means that the effect of the differentiation is exactly the same in both cases, more precisely the key variables with indices from J^* are no longer present in the differentiated polynomial.

A major difference compared to standard chosen IV attacks is that the differentiation over the IV cube corresponding to I^* would leave a complex polynomial g as the differential value so that no simple relations among key variables could be deduced. Indeed, the above equation can be rewritten as,

$$z^t = IV_{i_1}IV_{i_2}\cdots IV_{i_m} g(K \setminus K_{I^*}, IV \setminus IV_{J^*}) \bigoplus_{I=\emptyset}^{\{1,2,\ldots,\kappa\}} a'_{r,I} IV_I^{(i)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right],$$

where again $a'_{r,I} = 0$ if $I^* \preccurlyeq I$.

Thus, the result of the IV differentiation is some arbitrarily complex polynomial $g(K \setminus K_{I^*}, IV \setminus IV_{J^*})$ which due to the absence of key variables with indices from J^* is processed in a similar way as in the case of the related-key method.

It is important to notice that our assumption about the knowledge of the remaining IV bits, that is the portion $IV^* = \mathcal{I} \setminus \{IV_{i_1}, \ldots, IV_{i_m}\}$ may give a rise to unfair comparison to some other trade-off attacks and therefore we consider two variants. In the first model, the assumption is that we know the subset $\mathcal{I} \setminus \{IV_{i_1}, \ldots, IV_{i_m}\}$ of IV bits in advance which gives a better trade-off curve. In the second scenario, this subset of bits is not known in advance (which is more realistic assumption) and the algorithm is then supposed to create more tables for storing computed differentials for many choices (the number of choices will follow the birthday paradox) of these IV bits. The algorithm proceeds as follows.

Off-line precomputation phase

1. Assume that a fixed portion $IV \setminus \{IV_{i_1}, \ldots, IV_{i_m}\}$ of the IV vector is known, and suppose the presence of a differentiable polynomial in the keystream bit

$$z^t = p^t_{J^*,I^*} g^t(K \setminus K_{I^*}, IV \setminus IV_{J^*}) \bigoplus_{I=\emptyset : I^* \not\preccurlyeq I}^{\{1,2,\ldots,\kappa\}} a_{r,I} IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right],$$

   has been detected. The attacker now performs the differentiation of the output keystream expressions,

$$\bigoplus_{(IV_{i_1},\ldots,IV_{i_m}) \in GF(2)^m} z^t = g^t(K \setminus K_{I^*}, IV \setminus IV_{J^*}).$$

   Thus, these differentials can be stored in a sorted table T for each possible value of the key variables with indices in $\mathcal{J} \setminus J^*$. This way $2^{\kappa - m} \times d$ differentials are computed at time instances $t = 1, \ldots, d$ by summing the output values for all possible choices of the IV values in the set I. The number of differentials d, for each possible value for the subset of key variables above, is selected so that these binary sequences of

length d are uniquely distinguished with respect to the subkey. Since there are $2^{\kappa-m}$ such sequences our estimate is that $d \approx \kappa$ is sufficient.

The total storage required is therefore of order $\mathcal{O}(2^{\kappa-m})$, and the time complexity is dominated by the computation of differentials, which is $\mathcal{O}(2^{\kappa-m}2^m) = \mathcal{O}(2^\kappa)$.

On-line phase

1. The attacker observes the output keystream bits $z^{t,I^*}, \ldots, z^{t+d,I^*}$ generated by a fixed secret key and for all possible values of the IV variables with indices in I^*, whereas the remaining IV bits are known and kept fixed. This means that z^{t,I^*} is actually a sequence of bits computed for each possible value of $IV_{i_1}, \ldots, IV_{i_m}$. The data complexity is therefore 2^m output keystream sequences of length d.

2. From the observed keystreams $z^{t,I^*}, \ldots, z^{t+d,I^*}$ the attacker can evaluate the differentials
$$\bigoplus_{IV_{i_1},\ldots,IV_{i_m} \in GF(2)^m} z^{t,I^*}, \ldots, \bigoplus_{IV_{i_1},\ldots,IV_{i_m} \in GF(2)^m} z^{t+d,I^*}$$
in time instances $t, t+1, \ldots, t+d$. The evaluation only uses the observed keystream bits and is performed once for all. The time complexity is of order $\mathcal{O}(d2^m)$.The binary differential sequence of length d is then matched to the entries stored in the table T, so that the portion of the key bits $\mathcal{J} \setminus J$ of cardinality $\kappa - m$ is retrieved. The time complexity of this step is negligible, and the last step corresponds to recovering the remaining m key bits by an exhaustive search, thus the time complexity is $\mathcal{O}(2^m)$.

4.2 A comparison to cube testers and statistical chosen IV attacks

In 2009, the concept of the cube testers was introduced by Aumasson et. al in [1]. In difference to the standard cube attacks, the cube testers focus on detecting the nonrandom behaviors of a given cipher system. These nonrandom behaviors usually include the property of balancedness, the presence of linear/neutral variables and some other criteria. It was demonstrated that the cube testers can detect the nonrandomness over 18 reduced rounds of MD6 algorithm only with 2^{17} complexity. Moreover, cube testers can also successfully detect nonrandomness on TRIVIUM reduced to 885 rounds with only 2^{27} complexity (these results are current the best known records). It is clear that cube testers only detect nonrandom behavior rather than recovering the secret key.

Unlike the cube testers, our key differentiation attacks perform a key extraction process. Though the differentiation polynomial detection method in our attack shares a similar principle as the neutral variables tests in cube testers, the goal of our key differentiation attacks is to perform a generic TMTDO attack to recover the secret key rather than detecting the nonrandomness of the cipher system. More precisely, let us still consider that the output keystream bit z^t can be written as,
$$z^t = p_{J^*,I^*} g(K \setminus K_{I^*}, IV \setminus IV_{J^*}) \bigoplus_{I=\emptyset}^{\{1,2,\ldots,\kappa\}} a_{r,I} IV_I^{(i)} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right], \quad a_{r,I} = 0 \text{ if } I^* \preccurlyeq I. \quad (8)$$
where $p_{J^*,I^*} = \prod_{i \in I, j \in J}(k_j + IV_i)$.

For standard cube testers, the nonrandomness of $g(K \backslash K_{I^*}, IV \backslash IV_{J^*}) = \bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t$ may be checked. For instance, the fact that the key variables K_{I^*} are not present in$\bigoplus_{IV_{i_1},\ldots,IV_{i_m}} z^t$ does not give an immediate approach for employing this result in a key recovery attack. On the other hand, our key differentiation attack aims at recovering these secret bits in $(K \setminus K_{I^*})$ using TMTDO methods (see also Section 5, whereas a guess and determined attack is used for the remaining key bits. Therefore, for the recovery of secret key bits, the key differentiation attacks have a more generic approach than cube testers.

In 2008, Fischer et. al proposed a new method for key recovery attacks on stream ciphers [17] based on the framework for chosen IV statistical distinguishers [16]. The basic idea comes from the distributed algebraic structure of the encryption (Boolean) function $F(K, V)$, where K and V denote the secret key bits and the initialization vector bits, respectively. Given a partition $V = (U, W)$, an adversary can obtain simpler Boolean functions $C(K, W)$ from F by varying over the bits in U only. If this function $C(K, W)$ has some nonrandom behavior, for instance imbalanced property, or there are many key bits that have no influence on the values of $C(K, W)$, then the adversary can perform a key recovery attack, where the nonrandom behavior is usually detected by using the approximation of the polynomial description of $C(K, W)$, alternatively the probabilistic neutral key bits method [2] is used. Moreover, it is shown that the attack can perform a partial recovery of the key bits for 672 reduced rounds Trivium and to Grain-128 reduced to 80 rounds [17].

There are two main differences between chosen IV statistical analysis in [17] and our key differentiation attacks: (1) The distinguisher of chosen IV statistical analysis comes from the observation of the algebraic structure of function $C(K, W)$. But our distinguisher is based on the differential properties of summed output values of $C(K, W)$ by varying over the bits in U, i.e., the expression of the form $\sum_U C(K, W)$ is considered. (2) The distinguisher used in the chosen IV statistical analysis if a probabilistic one due to the probabilistic approximations (with probability $p < 1$) of the given functions $C(K, W)$. On contrary, the distinguisher in our key differentiation attacks always have probability $p = 1$, which is completely determined by the function $\sum_U C(K, W)$. Therefore, the chosen IV statistical analysis and our key differentiation attacks have different attack principles.

5 Time-memory trade-off attacks

Trivially, to retrieve a secret encryption key the adversary may try one of the two extreme solutions: an exhaustive key search and a table attack. The former attack is performed on-line in a known plaintext scenario by testing all possible keys until the key that matches a given plaintext-ciphertext pair is found. On the other hand, in the table attack the attacker precomputes the encryptions of a chosen plaintext using all possible keys. Then in the on-line phase the observed ciphertext is compared to the entries in the table to identify the key. Here, the on-line time complexity is negligible but the memory requirement is equal to the key space. Of course, the underlying assumption for the table attack is that the used IV is known to the attacker, otherwise a similar table must be constructed for

all possible IVs.

The time-memory trade-off (TMTO) attacks was first introduced by Hellman in the context of a chosen plaintext attack [21] to block ciphers, as a trade-off solution between the two extreme approaches above. The precomputed tables that cover the whole key space are smaller but the attack also implies a non-negligible time complexity in the on-line phase. Denoting by K a total key space the resulting trade-off is given as $TM^2 = K^2$, where M is the memory storage used in the precomputation phase, and T is the time complexity in the on-line phase. A convenient choice for these parameters is $M = T = K^{2/3}$ which gives a complexity beyond the exhaustive key search. The attack also includes a precomputation step that corresponds to K encryptions. The basic idea is to reuse the precomputed table (which is done once for all) for retrieving secret keys in subsequent encryptions, assuming that the same plaintext target has been encrypted. Remark that the attack is meaningful only if $M, T < K$.

In the realm of stream ciphers the first TMDTO (time-memore-data trade-off) attack was proposed independently by Babbage [4] and Golić [19]. Let N denote the size of the internal state of the cipher (in our notation $N = 2^L$). Then, given D data points (keystreams produced by the same secret key and different IVs) the goal of the attacker is to recover any secret state that corresponds to one of these D keystreams. In the precomputation phase the attacker evaluates the function,

$$f : (\log N)\text{-bit state} \to (\log N)\text{-bit output keystream},$$

and sorts the pairs $(\mathbf{s}, f(\mathbf{s}))$ with respect to the second coordinate in a table $\mathcal{T}$. If N/D such pairs are stored then by the birthday paradox one expects that one among the D keystreams will be found in the table. The time complexity is therefore $T = D$ and the precomputation and memory complexity are of the same size, i.e., $P = M = N/D$. The main problem with this approach is that the internal state of modern ciphers is made large (commonly $|N| \geq 2\kappa$) and therefore D must be very large so that the resulting attack (a point on the trade-off curve $N = T \cdot M$) is faster than exhaustive search. In particular, one may choose $T = M = D = N^{1/2}$ which also constitutes an attack, and it suggests that the length of the state is at least 2κ bits to achieve a theoretical κ bit security, see [4].

The TMDTO attack of Babage and Golic was improved by Biryukov and Shamir [9], using the Hellman's tables in the context of stream ciphers. In this case multiple Hellman's tables are created, where N/D of the total internal state. The attack again aims at recovering any of the secret states corresponding to D different keystreams. The main benefit of this approach is an improved TMTO curve given by,

$$TM^2D^2 = N^2, \quad P = M = N/D,$$

which is valid for $1 \leq D^2 \leq T$. The most useful point on the curve is $T = M = N^{1/2}$ and $D = N^{1/4}$. A further refinement of this approach was suggested in [22]. It was observed that the TMTO curve is not related to the state size but rather to the effective size of the state. Therefore, increasing the state of the cipher while keeping the sizes of the key and IV fixed does not affect the resistance to TMTO attacks. If κ and v denote respectively the key and IV length then a TMTO attack can be applied using $T = M = 2^{\frac{1}{2}(\kappa+v)}$ and

$D = 2^{\frac{1}{4}(\kappa+v)}$ rather than using $T = M = N^{\frac{1}{2}}$ and $D = N^{\frac{1}{4}}$, where the state space $N = 2^L$ can be significantly larger than $2^{\kappa+v}$. For $\kappa = v$, the on-line phase of the TMTO attack is $T = M = 2^\kappa$ and the precomputation phase is $P = 2^{\kappa+v}/D = 2^{\frac{3}{2}\kappa}$.

Prior to the work of Dunkelman and Keller [12], all the TMTO attacks against stream ciphers have treated the initialization vector as a part of secret material. That is, the function to be inverted was of the form $f : (K, IV) \rightarrow$ Output Prefix, where "Output Prefix" is a portion of the keystream of length $\kappa + v$ generated by the cipher. As argued in [12], in the context of a chosen IV attack, this treatment is not necessarily optimal.

Let $V = 2^v$ denote the number of all possible IVs and let D be the number of keystreams generated by the same secret key and different IVs. The attacker may in advance select a portion of all possible IV's, of cardinality V/D, and for each IV in this set he prepares Hellman's tables. But this time, since IVs are known, the function to be inverted is $f : K \rightarrow$ Output Prefix. The resulting trade-off curve using this approach turns out to be,

$$TM^2D^2 = (KV)^2.$$

Thus, the curve is essentially the same as the original curve of Biryukov and Shamir, but the difference lies in its extended range of applicability. Namely, the restriction on the parameters in the original curve $T \geq D^2$ is removed which gives more flexibility to select the trade-off parameters.

In the sequel we compare the performance of our attack to the standard TMDTO attacks mentioned above. Since the method of key differentiation is applicable in a related-key and chosen IV scenario, we consider the two cases separately. While in a related-key model our trade-off curve coincides with the key collision attack of Biham [6], in a chosen IV scenario our method yields a better trade-off curve than the above approaches.

5.1 TMDTO attack in a related-key model

The basic assumption of our related-key model is that the IV is known in advance (or even not used) while the keys to be used in encryptions are related in a known (incremental) manner. This means that in a classical TMTO attack the function to be inverted is of the form $f : (K, \text{fixed IV}) \rightarrow$ Output Prefix, where due to the increment of a subset of m key variables only the remaining $\kappa - m$ key bits are to be considered as unknown. As no multiple data points, in a standard sense, are used (the IV is fixed) the Hellman's tables is given by $TM^2 = K^2$, where we set $D = 1$.

As already noticed, our attack induces a trade-off curve given by $MT = K$ or alternatively $MD = K$, as $D = T$. In the related-key model the collision attack of Biham [6], which is essentially an application of the birthday paradox, seems to be the most appropriate method for the sake of comparison. Instead of precomputing the encryptions under all possible keys, in this method encryptions under $2^{\kappa-m}$ randomly chosen keys are stored in a table. Then if the attacker observes 2^m many encryptions with different keys, by the birthday paradox one expects that one of these keys will match to one of the keys stored in a table. The method is very simple and the resulting TMDTO curve is given by $M \cdot T = K$, where $M = 2^{\kappa-m}$ is the memory complexity and T is the on-line time complexity. The data complexity equals to time complexity $D = T$ as for each different key

the attacker only needs to perform a single table look-up, i.e. $T = 1$ for each data point. In our case the attack assumption is more restrictive, that is we assume that a portion of the secret key is incremented in different sessions. This means that the keys are not random. For both attacks, once the master key has been found all the other derived keys are compromised as well. However, if IV is not fixed better methods than the collision attack exist.

5.2 TMDTO attack in a chosen IV model

In the sequel we compare the TMDTO curve of Dunkelman and Keller to the trade-off curve that will be derived for our method. It turns out that our curve will be defined by $T^2M^2D^2 = (KV)^2$, which then implies a better performance of our attack.

In the approach of Dunkelman and Keller, the attacker chooses V/D many IVs in advance and given the D keystreams generated by different IVs with probability 40% the secret key will be found in the precomputed Hellman's tables. When the key differentiation is equivalent to the IV differentiation (the effect of elimination of a certain subset of key variables), we assume that a targeted subset of IV bits of cardinality m is incremented during different sessions, whereas the remaining part of the IV bits of size $v - m$ is fixed and has to be known in advance. The knowledge of these $v - m$ IV bits would then favour our attack, and the resulting trade-off curve would be given by $MT = K$ as in the case of the related-key attack. Indeed, the memory requirement is of size $\mathcal{O}(2^{\kappa-m})$ and the time complexity is $\mathcal{O}(2^m)$, which then gives $MT = K$.

Therefore, to make a fair comparison between the two methods, we assume the attacker has no a priori knowledge of this subset of IV bits (which is a more reasonable assumption by the definition of an IV). Now, the attacker is supposed to create $2^{(v-m)/2}$ tables for randomly chosen IVs, where only a portion of $v - m$ bits of these IVs is to be matched to the $v - m$ bits of the IV used in encryption. Then, if $2^{(v-m)/2}$ many sessions with different IVs are observed, then by the birthday paradox there will be a prepared table to match one of these IVs at these $v - m$ positions. This obviously implies a certain increase of the storage, namely by factor $2^{(v-m)/2}$ (compared to the known IV case), that is,

$$M' = 2^{(v-m)/2}M = 2^{(v-m)/2}2^{(\kappa-m)}.$$

Similarly, $D' = 2^{(v-m)/2}D = 2^{(v-m)/2}2^m$, while the time complexity remains the same $T' = T = 2^m$. Indeed, the attacker observes the used IVs and performs no operations until the portion of some IV gives a match. Then he computes the differentials for that session, retrieve $\kappa - m$ secret key bits and find the remaining key bits using an exhaustive search.

It is straightforward to deduce that our attack induces the following TMDTO curve,

$$T'M'D' = KV, \quad \text{or} \quad T'^2M'^2D'^2 = (KV)^2.$$

This attack scenario can be employed in the applications that use a new value for a portion of the IV bits in each encryption session, while the remaining IV bits are incremented for the encryption of each new frame, message etc.. Therefore, multiple data points $D' = 2^{(v-m)/2}2^m = D_sD_V$ are collected, where D_V denotes encryptions under the

same secret keys and different and related IVs, and D_s is the number of different sessions that use different values for the subset of $v-m$ IV bits.

In the above scenario, the method in [12] can also be adopted to take advantage of the known portion of the IV bits. The attacker only needs to consider $v-m$ IV bits while the remaining m bits can be set to arbitrary value (due to the assumption that the encryption in any single session exhaust all possible values of these bits). Thus, it suffices to construct Hellman's tables for $2^{(v-m)/2}$ randomly chosen IV values, where for each IV t tables of size s are constructed. Therefore, the memory requirement is $st2^{(v-m)/2}$. The t tables store the start and the end value of the repeated computation, $x \rightarrow f_i(x) \rightarrow f_i^2(x) = f_i(f_i(x)) \cdots \rightarrow f_i^t(x)$, for $i = 0, \ldots, t-1$, for some s random starting points x. Here, f_is are small modifications of function f to be inverted. Given the IV vector for which the Hellman's tables are created, to find the value of the secret key in the table (that is to find x given some $f(x)$) the t^2 calls to f_is are needed, which is the time complexity of the attack. In order to cover the whole key space the size of Hellman's tables must satisfy $st^2 = K$. Using these relations, the following trade-off curve can be derived,

$$TM^2 = t^2 s^2 t^2 2^{(v-m)} = K^2 V 2^{-m} \Rightarrow T D_V M^2 = K^2 V.$$

Compared to the original TMDTO curve of Dunkelman and Keller given by $TM^2D^2 = (KV)^2$, the modification of the attack scenario yields a certain improvement.

Let us compare the performance of the algorithms taking into account these modifications. Obviously our time complexity is 2^m which cannot be reached by the TMDTO of Dunkelman and Keller unless the memory usage is unreasonably high. For $m = 40$, $|IV| = |K| = 80$, the following trade-off parameters are obtained for our method,

$$M = 2^{60}, \;\; T = 2^{40}, \;\; D = D_s D_V = 2^{20} 2^{40} = 2^{60}.$$

If we now allow even for a larger memory storage $M = 2^{70}$ the trade-off parameters of Dunkelman and Keller are given as,

$$K = st^2, M = st2^{(v-m)/2} \;\Rightarrow\; t = 2^{30}, s = 2^{20} \;\Rightarrow T = 2^{60}$$

which is clearly inferior to our trade-off.

Another comparison (and many others can be given) relates to the particular instance used in [12]. The authors consider an application of their method to a stream cipher with parameters $|K| = 64$, and $|V| = |IV| = 40$. For this instance, they select the following point on their TMDTO curve, $T = M = 2^{48}$, $D = 2^{32}$.

Fixing the memory storage $M = 2^{40}$ and $m = 30$, the modified TMDTO curve $TD_V M^2 = K^2 V$, that compensates for a partial knowledge of the IV bits, gives in this case,

$$K = st^2, M = st2^{(v-m)/2} \;\Rightarrow\; t = 2^{29}, s = 2^6 \;\Rightarrow T = 2^{58}.$$

If $m = 30$ is chosen in our attack, then the corresponding trade-off is,

$$M = 2^{(v-m)/2} \cdot 2^{\kappa - m} = 2^{39}, \;\; T = 2^m = 2^{30}, \;\; D = 2^{(V-m)/2} 2^m = 2^{35}.$$

6 Conclusions

We have developed a theoretical framework for using the technique of key differentiation in the cryptanalysis of certain stream cipher designs. The attack is generic in a related-key model while in a more realistic chosen IV model the key differentiation process may only be performed if the presence of so called differentiable polynomials is identified in keystream expressions. We leave the question of finding other suitable stream cipher algorithms susceptible to this attack open.

References

[1] J-P. Aumasson, I. Dinur, W. Meier, and A. Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In *Fast Software Encryption 2009*, volume LNCS 5665, pages 1–22. Springer-Verlag, 2009.

[2] J-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger. Fast software encryption.

[3] S. Babbage. Cryptanalysis of LILI-128. Preproceedings of NESSIE 2-nd workshop, Egham, 2001.

[4] S. H. Babbage. Improved exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, volume 408, pages 161–166. IEE, 1995.

[5] S. S. Bedi and N. R. Pallai. Cube attacks on Trivium. Cryptology ePrint Archive, Report 2009/015, 2009. `http://eprint.iacr.org/`.

[6] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Information Processing Letters*, vol. 84(3):117–124, 2002.

[7] E. Biham and O. Dunkelman. Differential cryptanalysis in stream ciphers. Cryptology ePrint Archive, Report 2007/218, 2007. `http://eprint.iacr.org/`.

[8] E. Biham and J. Seberry. Py (Roo: A fast and secure stream cipher using rolling arrays. Avaliable on ECRYPT Stream Cipher Project page, 2005. `http://www.ecrypt.eu.org/stream/py/py.ps`.

[9] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *Advances in Cryptology—ASIACRYPT 2000*, volume LNCS 1976, pages 1–13. Springer-Verlag, 2000.

[10] C. Canniere and B. Preneel. TRIVIUM specifications. Avaliable at ECRYPT Stream Cipher Project page, 2005. `http://www.ecrypt.eu.org/stream/trivium.html`.

[11] I. Dinur and A. Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology—EUROCRYPT 2009*, volume LNCS 5479, pages 278–299. Springer-Verlag, 2009.

[12] O. Dunkelman and N. Keller. Treatment of the initial value in Time-Memory-Data Tradeoff attacks on stream ciphers. *Information Processing Letters*, vol. 107(5):133–137, 2008.

[13] O. Dunkelman, N. Keller, and J. Kim. Related-key rectangle attack on the full SHACAL-1. In *Selected Areas in Cryptography 2006*, volume LNCS 4356, pages 28–44. Springer-Verlag, 2006.

[14] ECRYPT. Call for stream cipher primitives. `http://www.ecrypt.eu.org/stream/`.

[15] P. Ekdahl and T. Johansson. SNOW—a new stream cipher. In *Proceedings of First Open NESSIE Workshop, KU-Leuven*, 2000.

[16] H. Englund, T. Johansson, and M. S. Turan. A framework for chosen IV statistical analysis of stream ciphers. In *Advances in Cryptology—INDOCRYPT 2007*, volume LNCS 4859, pages 268–281. Springer-Verlag, 2007.

[17] S. Fischer, S. Khazaei, and W. Meier. Chosen IV statistical analysis for key recovery attacks on stream ciphers. volume LNCS 5023, pages 236–245. Springer-Verlag, 2008.

[18] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses of the key scheduling algorithm of RC4. In *Selected Areas in Cryptography 2001*, volume LNCS 2259, pages 1–24. Springer-Verlag, 2001.

[19] J. Dj. Golić. Cryptanalysis of alleged A5 stream cipher. In *Advances in Cryptology—EUROCRYPT'97*, volume LNCS 1233, pages 239–255. Springer-Verlag, 1997.

[20] M. Hell, T. Johansson, and W. Meier. A stream cipher proposal: Grain-128. In *IEEE International Symposium on Information Theory (ISIT 2006)*, 2006.

[21] M. Hellman. A cryptanalytic time-memory tradeoff. *IEEE Trans. on Inform. Theory*, IT-26(4):pages 401–406, 1980.

[22] J. Hong and P Sarkar. New applications of time memory data tradeoffs. In *Advances in Cryptology—ASIACRYPT 2005*, volume LNCS 3788, pages 353–372. Springer-Verlag, 2005.

[23] S. Hong, J. Kim, S. Lee, and B. Preneel. Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In *Fast Software Encryption 2005*, volume 3557, pages 368–383. Springer-Verlag, 2005.

[24] W. Hongjun and B. Preneel. Differential cryptanalysis of the stream ciphers Py, Py6 and PyPy. In *Advances in Cryptology—EUROCRYPT 2007*, volume LNCS 4515, pages 276–290. Springer-Verlag, 2007.

[25] A. Joux and F. Muller. A chosen IV attack against Turing. In *Selected Areas in Cryptography*, volume LNCS 3006, pages 194–207. Springer-Verlag, 2003.

[26] L. R. Knudsen. Cryptanalysis of LOKI 91. In *AUSCRYPT 92*, volume LNCS 718, pages 196–208. Springer-Verlag, 1993.

[27] G. Sekar, P. Souradyuti, , and B. Preneel. Related-key attacks on the Py-family of ciphers and an approach to repair the weaknesses. In *Advances in Cryptology—INDOCRYPT 2007*, volume 4859, pages 58–72. Springer-Verlag, 2007.

[28] M. Vielhaber. Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. Cryptology ePrint Archive, Report 2007/413, 2007. `http://eprint.iacr.org/`.