Key differentiation attacks on stream ciphers

Abstract

In this paper the applicability of differential cryptanalytic tool to stream ciphers is elaborated using the algebraic representation similar to early Shannon's postulates regarding the concept of confusion. In 2007, Biham and Dunkelman [3] have formally introduced the concept of differential cryptanalysis in stream ciphers by addressing the three different scenarios of interest. Here we mainly consider the first scenario where the key difference and/or IV difference influence the internal state of the cipher $(\Delta key, \Delta IV) \rightarrow \Delta S$. We then show that under certain circumstances a chosen IV attack may be transformed in the key chosen attack. That is, whenever at some stage of the key/IV setup algorithm (KSA) we may identify linear relations between some subset of key and IV bits, and these key variables only appear through these linear relations, then using the differentiation of internal state variables (through chosen IV scenario of attack) we are able to eliminate the presence of corresponding key variables. The method leads to an attack whose complexity is beyond the exhaustive search, whenever the cipher admits exact algebraic description of internal state variables and the keystream computation is not complex. A successful application is especially noted in the context of stream ciphers whose keystream bits evolve relatively slow as a function of secret state bits. A modification of the attack can be applied to the TRIVIUM stream cipher [8], in this case 12 linear relations could be identified but at the same time the same 12 key variables appear in another part of state register. Still, a significant decrease in the degree and complexity of state bit expressions after the KSA is achieved. Computer simulations, currently in progress, will answer the question for what number of initialization rounds the attack is faster than exhaustive search.

1 Introduction

In 1990, Biham and Shamir [5] have proposed a new cryptanalysts' tool called differential cryptanalysis. This technique, along with linear cryptanalysis of Matsui [16], has been considered as one of the most powerful attacks on block ciphers to date. However, the same ideas have not been applied in the same extent in cryptanalysis of stream ciphers until recently. The use of concepts similar to differential cryptanalysis for block ciphers was frequently employed frequently during the eSTREAM project [10]. These attacks are commonly referred as chosen IV or key related 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 utilized in the cryptanalysis of the stream cipher LILI-128 [1], 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 [15] in cryptanalysis of

TURING cipher. In other direction the eSTREAM proposal Py [4] (and its tweaks PyPy and PyPy6) was successfully cryptanalyzed using the differential cryptanalysis [14], more precisely for any given key the subset of IV's was identified so that there was two IV's that would generate the identical initial state (and therefore identical keystream). The ideas of differential cryptanalysis have also been successfully applied in so-called fault analysis [6]. In such an attack the adversary introduces errors during the computation which leads to errors in the output. The difference between an unfaulty and a faulty computation would then reveal certain information about the computation itself.

This paper aims to give a thorough analysis of chosen IV and key related attacks in the view of Shannon's algebraic representation of encryption schemes. In difference to a general framework studied in [3], where the effect of IV/key difference is analyzed through the differential characteristics at the state variables $((\Delta key, \Delta IV) \rightarrow \Delta S)$, we take another approach and consider the differentiation of the key variables in the algebraic expressions of the initial state variables. Though somewhat related to approaches recently taken in [11, 12, 18], where statistical analysis or efficient application of chosen IV attacks is investigated, our approach only addresses particular designs for which linear relations between the key and IV bits may be derived at some stage of the initialization procedure. Then it turns out that there is an efficient way to eliminate the presence of the key bits that are linearly mixed with IV bits in secret state bit expressions. Therefore the attack is always successful in such a kind of scenarios, though the modifications of the attack may be applicable to other settings as well. In this context we also mention a new generic attack known as Cube Attack [9], where the method of tweaking the polynomials (as a generic method applicable in a black box scenario) results in an efficient attack on certain stream cipher designs.

Though the related key attack (differential key attack) is a more unrealistic scenario than a chosen IV attack, we show that this scenario is far more dangerous as we can completely eliminate the presence of certain key variables under the assumption that the IV is virtually kept fixed. To satisfy the condition that IV is kept fixed, we essentially consider a chosen IV attack and a fixed key (which is much more realistic scenario), and then in cases that a certain subset of IV and key bits are related in a linear manner after the initial loading, say $IV_i \oplus k_j$, the IV difference is transformed into the key difference.

Once we have successfully eliminated the presence of certain key variables, through the differentiation of the state variables, a simple key guessing attack may be combined to further reduce the complexity of the state variables. Obviously by guessing many key variables we can make the internal state bit equations to be arbitrary simple functions in remaining key variables. Then, if the keystream generation does not combine these state variables in a complicated manner (or at least a part of the keystream does not depend on the state variables in a complex way) we can efficiently solve the system for the remaining key bits.

As an application of this approach we consider TRIVIUM [8], a hardware oriented stream cipher that reached the final third phase of the eSTREAM project [10]. Unfortunately, TRIVIUM design does not allow the full usage of the attack, so there is no guarantee that the running time of the attack is beyond the complexity of exhaustive search. This is due to the fact that the same key bits that are linearly related to a subset of IV bits, are present in another part of the state register, this time without involving

the IV bits. However, the differentiation efficiently reduces the number of terms and the degree of the expressions for state bits after the key/IV setup algorithm. Currently, we are running a computer program to estimate the expected complexity of the attack depending on the number of rounds.

Another suitable design, due to its specific key and IV loading, is a stream cipher DECIM-128 [2]. But in this case the attack is only applicable to a modified cipher that omits the output decimation.

The paper is organized as follows. In Section 2 a convenient algebraic representation of the internal state variables is introduced. The main differential attack scenarios are discussed in Section 3. It is then shown that the key differentiation in general eliminates the presence of the key variables the differentiation is performed on. Here we also deduce the conditions when an IV chosen attack can be transformed into a key differential attack. The modification of the attack to cover the case when a subset of key variables appear as linearly related to some IV bits but also without the linear connection to IV bits in some other part of the state is discussed in Section 4. In Section 5, we discuss the application of our method to stream ciphers TRIVIUM and DECIM-128. Concluding remarks are given in Section 6.

2 Algebraic representation of the initialization process

For the rest of this paper we only consider synchronous stream cipher as a dominant part of the family, though the similar analysis as given here can be easily extended to asynchronous stream ciphers. Technically, given a stream cipher, 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. The key setup and IV setup are commonly performed in a single phase so that the secret state of the cipher is derived by processing key and IV bits in an iterative manner. The result of this phase is the so-called initial state $S = (s_0, s_1, \ldots, s_{L-1}) = F(IV_1, \ldots, IV_\kappa, k_1, \ldots, k_\kappa)$, for simplicity assuming the same length of the key and IV, and in addition denoting by $L \geq 2\kappa$ the state size; the state size being at least twice the key length to withstand time-memory-data trade-off attacks, see for instance [7, 13]. Now given a fixed key K and the publicly known $\mathrm{IV}^{(i)}$ the system of equations may be written as,

$$\begin{aligned} s_0^{(i)} &= f_0(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) = g_0(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) + h_0(k_1, \ldots, k_\kappa) \\ &\vdots \\ s_{L-1}^{(i)} &= f_{L-1}(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) = g_{L-1}(IV_1^{(i)}, \ldots, IV_\kappa^{(i)}, k_1, \ldots, k_\kappa) + h_{L-1}(k_1, \ldots, k_\kappa) \end{aligned}$$

The reason that we collect the terms that only contain the key bits in function g is that the differential of the form $s_r^{(i)} \oplus s_r^{(j)}$, $0 \leq r \leq L-1$, will not contain the key bit terms for arbitrary $\mathrm{IV}^{(i)}$ and $\mathrm{IV}^{(j)}$. This of course may be generalized for any set of IV's of even weight.

Following the early ideas of Shannon [17], each s_i should depend in a complicated manner on the key bits and involve many (preferably all) of them. This is also true for the IV bits, as otherwise the differential $s_r^{(i)} \oplus s_r^{(j)}$ may result in simple expressions that relate key and IV bits to the secret state bits. One should notice that there is a certain freedom

in selecting these relations, the freedom that may additionally reduce the complexity of the system.

Let $\mathcal{I}, \mathcal{J} = \{1, 2, \ldots, \kappa\}$ denote the set of indices for the IV vector respectively the key bits. Then a convenient algebraic representation, used in [18], is to represent the secret bits as a collection of terms as follows,

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

where $a_I^{(i)}, b_J^{(i)}$ are algorithm specific binary coefficients, and for a given subset $I = \{i_1, \ldots, i_l\} \subset \mathcal{I}$ the term $IV_I = IV_{i_1} IV_{i_2} \cdots IV_{i_l}$ and similarly $K_J = k_{j_1} k_{j_2} \cdots k_{j_p}$ for some $J = \{j_1, \ldots, j_p\} \subset J$. We note that treating both the key and IV as variables the coefficients $a_{r,I}, b_{r,J}$ do not depend on the specific choice of IV's. The above expressions are valid for a wide class of stream cipher, the specific details being the size of the key, IV, state, and the particular KSA used.

If the secret state bits s_r had been generated randomly as the functions of the key and IV bits, then one would expect that the algebraic normal forms of each s_r contains approximately $2^{2\kappa-1}$ terms. In addition the high degree terms of order $2\kappa - 1$ should be present in such expressions as well, thus closely following the Shannon's idea of making complicated relations among the state bits and secret key bits. Given an algebraic description of the state of the cipher after running the KSA (this is commonly infeasible as the number of terms is excessively large), 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.

3 Transforming chosen IV into key related attacks

In this section we consider the main attack scenarios in the realm of the differential cryptanalysis. Though similar in their essence, the cryptanalytic aspects significantly differ depending in which scenario the attack is performed.

3.1 Chosen IV attacks

A chosen IV attack is a very realistic scenario in which the attacker chooses certain subset of all possible IV's in order to combine the relations of resulting state bits and 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 specify IV values to be zero in all positions but those that correspond to I. Thus, given the $\text{IV}^{(i)}$ and a vector $I = \{i_1, i_2, \ldots, i_p\}$ for which $\text{IV}_l = 0, l \notin I$ the secret state bits could be

expressed as,

$$s_r^{(i)} = \bigoplus_{\xi \subset I} a_{r,I} IV_\xi \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,\xi,J} K_J\right] \oplus \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right] = IV_{i_1} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,i_1,J} K_J\right] \oplus$$
$$\oplus \quad IV_{i_2} \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,i_2,J} K_J\right] \oplus \cdots \oplus IV_{i_1} IV_{i_2} \cdots IV_{i_p} [\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J] \oplus [\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J]$$

Then keeping the key fixed we can further manipulate these expressions by specifying different IV's from this subset of indices. For instance, taking the $\mathrm{IV}^{(1)}$ and $\mathrm{IV}^{(2)}$ such that they differ in only one position, i.e. $\mathrm{IV}_{i_1}^{(1)} \neq \mathrm{IV}_{i_1}^{(2)}$ and $\mathrm{IV}_i^{(1)} = \mathrm{IV}_i^{(2)}$ for all $i \neq i_1$, would result in,

$$s_r^{(1)} \oplus s_r^{(2)} = \oplus_{J=\emptyset}^{\mathcal{J}} b_{r,i_1,J} K_J. \tag{2}$$

This is exactly the idea that has been used in [18], where an semi-exhaustive search was performed over the suitable set of indices I of weight 6 to derive linear expressions relating keystream bits of a reduced KSA TRIVIUM [8] and the secret key bits. However, the authors derive such relations by setting the remaining IV bits (74 bits in the case of TRIVIUM) to zero which makes the attack slightly unrealistic. The above expressions suggests that the type of relations similar to (2) may be derived for any fixed IV values making such an approach more realistic.

Thus, a pseudo algorithm for finding simple relations after the KSA procedure may be formulated as follows:

1. Locate some subset of indices I for which the particular KSA does not generate sufficiently complicated relations in secret key bits.

2. Based on this observation for any fixed combination of remaining IV bits not in I find relations of the form $s_r^{(1)} \oplus s_r^{(2)} \oplus \cdots \oplus s_r^{(m)}$ by varying the IV vectors $\mathrm{IV}^{(1)}, \mathrm{IV}^{(2)}, \ldots, \mathrm{IV}^{(m)}$ over the subset I, for all $r \in [0, L-1]$.

3. Once such relations are found these are used in the keystream generation process to derive low degree relations relating the keystream bits and the secret key bits (through the secret initial state bits).

3.2 Related key attacks

In this scenario 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. This may be regarded as quite impractical scenario since a secure key management and derivation usually should not allow to locate the part of the bits which remain unchanged after the key derivation. Nevertheless, many KSA schemes allow a simple mixture of the IV and secret key bits and therefore there is a simple and elegant way to transform the chosen IV attack into the key related attack.

To illustrate this approach let the key K be fixed and for simplicity consider two different IV vectors $\mathrm{IV}^{(1)}$ and $\mathrm{IV}^{(2)}$ that differ in certain positions specified by some subset I, that is $\mathrm{IV}_i^{(1)} \neq \mathrm{IV}_i^{(2)}$ for all $i \in I$. Furthermore, we assume that the KSA procedure is such that a part of the cipher state of cardinality p is initialized by a simple XOR-ing of the IV and key bits, and then the KSA is run to generate the final initial state of the cipher. The initial loading of the IV and key bits is referred to as pre-setup phase,

resulting in the pre-setup state $S^{ps} = (s_0^{ps}, s_1^{ps}, \ldots, s_{L-1}^{ps})$, where the XOR-ing is applied as follows,

$$s_{jl}^{ps(1)} = \mathrm{IV}_{il}^{(1)} \oplus k_{m_l}; \;\; s_{jl}^{ps(2)} = \mathrm{IV}_{il}^{(2)} \oplus k_{m_l}, \quad l \in \{1, 2, \ldots, p\}$$

for a suitable choice of indices sets say I, J, M that entirely depends on the specific loading mechanism. Neglecting the due details it is easy to realize that instead of considering the difference in IVs we may consider the relation so that $\mathrm{IV}^{(1)} = \mathrm{IV}^{(2)}$ but the key is no longer fixed and consequently one may define K',

$$K' = \begin{cases} k'_m = k_m; & m \notin M, i \notin I \\ k'_m = 1 \oplus k_m; & m \in M, i \in I \end{cases}$$

For simplicity, an IV differential in a single position i, i.e. $IV_i^{(2)} = IV_i^{(1)} \oplus 1$, would result in the key difference through,

$$s_r^{ps(1)} \;\; = \;\; IV_i^{(1)} \oplus k_m; \;\; s_r^{ps(2)} = IV_i^{(2)} \oplus k_m = IV_i^{(1)} + (k_m \oplus 1) = IV_i^{(1)} + k'_m.$$

Thus, we may analyze the effect of a single bit difference in IVs that is transformed in the key difference. As before, at the end of the initialization, this difference would give,

$$\begin{aligned} \Delta^{k_m} s_r \stackrel{\mathrm{def}}{=} s_r^{k_m} \oplus s_r^{k'_m} \;\; &= \bigoplus_{I \subset \mathcal{I}; I \neq \emptyset} a_{r,I} IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K_J\right] \bigoplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J \\ &\oplus \bigoplus_{I \subset \mathcal{I}; I \neq \emptyset} a_{r,I} IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,I,J} K'_J\right] \bigoplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K'_J, \end{aligned} \tag{3}$$

where $k_j = k'_j$ for all j except for some $j = m$. This implies that in the above sum most of the terms are canceled leaving only those K_J's for which $m \in J$. Then for any $I \subset \mathcal{I}$ and $J \subset \mathcal{J}$ such that $m \in J$ we can compute

$$\begin{aligned} &IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J\right] \oplus IV_I \left[\oplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K'_J\right] = \\ &IV_I[k_m g(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa) \oplus h(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa)] \oplus \\ &IV_I[k'_{i_1} g(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa) \oplus h(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa)] = \\ &IV_I g(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa). \end{aligned}$$

It is straightforward to verify that the same arguments apply to the differential

$$\bigoplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K_J \oplus \bigoplus_{J=\emptyset}^{\mathcal{J}} b_{r,J} K'_J = h(k_1, \ldots, k_{m-1}, k_{m+1}, \ldots, k_\kappa),$$

thus, in general, lower degree expression (the decrease of the degree is at least one) in the secret key bits are derived for $\Delta^{k_m} s_r$.

Example 1 *Assume that the KSA for a given cipher enables a single IV difference to be translated into the key difference in k_1. Let the secret state bit s_r be evaluated as*

$s_r = k_2k_3k_4k_5 \oplus k_1k_3k_4 \oplus k_2k_5k_6 \oplus k_2k_6 \oplus k_3 \oplus k_1$*; after the IV value has been specified. Then differentiating the first key bit* k_1 *we would compute,*

$$\Delta^{k_1} s_r = s_r^{(k_1)} \oplus s_r^{(k_1+1)} = k_3k_4 \oplus 1.$$

In this case we have decreased the degree of s_r *by two in a single step due to the cancellation of the highest degree term* $k_2k_3k_4k_5$ *for which the differentiation does not apply as it does not contain the variable* k_1*.*

A natural question that arise now is how to generalize this approach to include differentiation of as many key bits as possible. At the first sight this is not possible due to the following arguments. Let us define the key differential at positions $k_{j_1}, \ldots, k_{j_m}$ with respect to some initial state bit s_r as,

$$\Delta^{k_{j_1},\ldots,k_{j_m}} s_r = s_r^{(k_{j_1},\ldots,k_{j_m})} \oplus s_r^{(k_{j_1}+1,\ldots,k_{j_m}+1)},$$

where we simply evaluate the differential by specifying the key difference simultaneously at given positions. Without loss of generality, to skip a complicated notation, we assume that the secret state bit s_r contains a single term i.e. $s_r = IV_I k_{j_1}k_{j_2}\cdots k_{j_{m-1}}k_{j_m}k_{j_{m+1}}\cdots k_{j_{m+v}}$. Then it is easy to compute,

$$\begin{aligned}
\Delta^{k_{j_1},\ldots,k_{j_m}} s_r &= IV_I k_{j_1}k_{j_2}\cdots k_{j_{m-1}}k_{j_m}k_{j_{m+1}}\cdots k_{j_{m+v}} \oplus \\
&\quad IV_I(k_{j_1}\oplus 1)(k_{j_2}\oplus 1)\cdots(k_{j_{m-1}}\oplus 1)(k_{j_m}\oplus 1)k_{j_{m+1}}\cdots k_{j_{m+v}} = \\
&\quad IV_I k_{j_{m+1}}\cdots k_{j_{m+v}}[1 + k_{j_1} \oplus \ldots + k_{j_m} \oplus k_{j_1}k_{j_2} \oplus \ldots \oplus k_{j_{m-1}}k_{j_m} \oplus \\
&\quad \ldots \oplus k_{j_2}\cdots k_{j_{m-1}}k_{j_m}],
\end{aligned}$$

the expression containing all the monomials of degree less than m in the key bits $k_{j_1}, \ldots, k_{j_m}$. Consequently, simultaneous differentiation at m positions does only guarantee the degree decrease of order one but not m as desired. Nevertheless, one can get rid of the variables $k_{j_1}, k_{j_2}\cdots, k_{j_m}$ by simply considering the linear combinations of the weighted differentials of the form,

$$\bigoplus_{J\subset\{k_{j_1},\cdots,k_{j_m}\};J\neq\emptyset} \Delta^J s_r$$

For instance, to make the key bits k_{j_1} and k_{j_2} vanish from s_r we would compute the following differentials : $\Delta^{k_{j_1}} s_r, \Delta^{k_{j_2}} s_r$, and $\Delta^{k_{j_1},k_{j_2}} s_r$ and add them together. To confirm the validity of our claim for the above specified s_r we would compute,

$$\begin{aligned}
\bigoplus_{J\subset\{k_{j_1},k_{j_2}\}} \Delta^J s_r &= \Delta^{k_{j_1}} s_r \oplus \Delta^{k_{j_2}} s_r \oplus \Delta^{k_{j_1},k_{j_2}} s_r = \\
&= IV_I k_{j_3}\cdots k_{j_m}k_{j_{m+1}}\cdots k_{j_{m+v}}[(k_{j_2}) \oplus (k_{j_1}) \oplus (k_{j_2} \oplus k_{j_1} \oplus 1)] = \\
&= IV_I k_{j_3}\cdots k_{j_m}k_{j_{m+1}}\cdots k_{j_{m+v}}
\end{aligned}$$

Of course the result is easily generalized for arbitrary number of terms and the degree order. For instance, if there is another term in the s_r, say $\alpha = IV_{I'}k_{j_1}k_{j_{m+u}}\cdots k_{j_{m+v}}$, thus

containing only the variable k_1 (it is covered by the variables k_1 and k_2) then computing $\bigoplus_{J\subset\{k_{j_1},k_{j_2}\}} \Delta^J\alpha$ would give,

$$\begin{aligned}\bigoplus_{J\subset\{k_{j_1},k_{j_2}\}} \Delta^J\alpha &= \Delta^{k_{j_1}}\alpha \oplus \Delta^{k_{j_2}}\alpha \oplus \Delta^{k_{j_1},k_{j_2}}\alpha = \\ &= IV_{I'}k_{j_{m+u}}\cdots k_{j_{m+v}} \oplus 0 \oplus IV_{I'}k_{j_{m+u}}\cdots k_{j_{m+v}} = 0,\end{aligned}$$

and there is no influence of the terms that are covered by the chosen subset of key variables. Thus, the only terms that remain after the differentiation are the complex expressions in the remaining key variables, the terms of the form $IV_I f_I(k_{j_{m+1}},\ldots,k_{j_\kappa})$.

It should be emphasized that in difference to differential IV attacks the differential key attacks are far more dangerous as the differentiation in the latter case affects the key variables, whereas in case of IV differentials the decrease of the degree with respect to IV variables has no effect after the IV has been set to a given value. On the other hand, the condition that the IV and key variables are first XOR-ed and then processed via the KSA might be rather restrictive.

3.3 Combining a related key and key guessing attack

In the previous section we have shown that there is a possibility to differentiate the secret state equations at those key positions for which the loading procedure XORs the IV's and key bits. In this way we could eliminate the presence of the corresponding key variables making the equation(s) depend on less number of key variables, thereby reducing its complexity. An interesting approach that can be taken now is simply to guess a subset of remaining key variables leading to a further degree reduction. Under a reasonable assumption that the secret state bits (after the initialization) do not differ significantly in terms of degree and complexity of the expressions we propose the following pseudo algorithm for attacking the ciphers whose loading of a certain portion of the key and IV bits is done by simple XOR-ing.

1. Find the key positions k_j (subset of indices) for which the given KSA injects the key and IV bits via $IV_i \oplus k_j$. Denote this subset by $J = \{k_{j_1}, k_{j_2}, \ldots, k_{j_m}\}$.

2. After loading the all key and IV bits, thus reaching the state $S^{ps} = (s_0^{ps}, s_1^{ps}, \ldots, s_{L-1}^{ps})$, run the KSA to get the finial initial state of the cipher $S = (s_0, s_1, \ldots, s_{L-1})$.

3. Depending on the complexity of the expressions $s_r = f_r(IV_1,\ldots,IV_\kappa,k_1,\ldots,k_\kappa)$ select a suitable subset of IV's for which these equations are of relatively low complexity.

4. Differentiate the key variables with respect to the subset J so that $\Delta^J s_r$ do not depend on the key variables from the indices set J for any $0 \le r \le L-1$.

5. Guess a suitable subset of key bits so that the expressions $\Delta^J s_r$ become of low degree (preferably linear).

6. Run the cipher in the output mode, i.e. producing keystream z_t, and analyze the relations $z_t = G(s_0, s_1, \ldots, s_{L-1}) = H(k_{j_{m+1}}, k_{j_{m+2}}, \ldots, k_{j_\kappa})$. Observe the suitable

part of output sequence for which the function G does not introduce complicated (high degree) equations in secret state bits. Solve the system of equations and check its consistence (if necessary guess a few more key bits and solve the system). If the solution is inconsistent guess another key value in the previous step and repeat.

Note that the complexity of the above algorithm mainly depends on the specific KSA and the way the cipher generates the keystream (complicated relations on initial state bits or not). However, since the key bits $k_{j_1}, k_{j_2}, \ldots, k_{j_m}$ have vanished from these equations the worst case complexity is estimated as,

$$C^{w.c.} = 2^{\kappa - m},$$

which correspond to guessing all the remaining bits. Once these bits are correctly set it remains to assign the unknown key bits $k_{j_1}, k_{j_2}, \ldots, k_{j_m}$ which can be easily achieved by plugging in these bits in the equations and checking the consistency with keystream bits z_t. Thus, the total worst case complexity is estimated as,

$$C^{w.c.} = 2^{\kappa - m} + 2^m,$$

thus in the case $m \ll \kappa$ we approximately have $C^{w.c.} = 2^{\kappa - m}$. We illustrate the whole idea with an example, considering a hypothetical toy cipher satisfying all necessary conditions for the attack to work.

Example 2 *Let us consider a toy cipher that use the key $K = (k_1, k_2, k_3, k_4) = (0, 1, 1, 1)$, and assume that IV difference in two bits is translated in the key difference at positions k_1 and k_2. For some specified IV let the first three (secret) initial state bits be given as,*

$$\begin{aligned} s_0 &= k_1k_2k_3 + k_3k_4 + k_3 + k_4 = 0, \\ s_1 &= k_1k_3k_4 + k_2 + k_4 = 0, \\ s_2 &= k_1k_2k_4 + k_2k_3k_4 + k_1 + k_2 + k_3 = 1. \end{aligned}$$

Obviously we cannot get rid of the degree 3 terms by considering linear combinations of the above equations. Let us compute the differentials for the 2 keystream bits $z_1 = s_0 + s_1$ and $z_2 = s_0 + s_2$,

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

The attacker observes outputs z_1 and z_2 under different IV's (translated into the key difference), thus he is able to compute the above differentials. For instance, to compute $\Delta^{k_1} z_1$ he would simply observe the first keystream bit for a specified IV and fixed key K and then the same keystream bit for a single change of one bit in the IV corresponding to the key position k_1. For the given values of K we may check that $\bigoplus_{J \subset \{k_1,k_2\}} \Delta^J z_1 = k_3 = 1$ and $\bigoplus_{J \subset \{k_1,k_2\}} \Delta^J z_2 = k_3 + k_4 = 0$. Hence, the correct values of the key bits $k_3 = 1$ and $k_4 = 1$ are found.

4 Adjusting attack to different scenarios

When applicable, the key related attack as discussed above is a powerful cryptanalytic tool. Identifying some subset of key bits that only appears through linear relations with some IV bits at certain stage of initialization process, would give a complexity that is always beyond the exhaustive key search for the family of ciphers susceptible to this kind of algebraic cryptanalysis (excluding for instance ciphers that use irregular clocking). However, one needs to be careful when deriving the key differentials through the IV change. The reason is that the KSA of a given cipher starts to process the pre-setup state $S^{ps} = (s_0^{ps}, s_1^{ps}, \ldots, s_{L-1}^{ps})$ where the IV and key bits are initially loaded. Though some of these s_r^{ps} are expressible as $s_r^{ps} = IV_i \oplus k_j$ this may not be sufficient. For instance, if the loading procedure injects $s_0^{ps} = k_1$ and $s_1^{ps} = IV_1 \oplus k_1$ there is obviously ambiguity when translating the IV difference at position IV_1 into the key difference. Remember that the main idea is to transform the modification of IV bits into key difference is not valid in this case, as we simply cannot differentiate the state bit $s_0^{ps} = k_1$.

Nevertheless, the attack is still applicable in cases when the same subset of key bits is both linearly combined with some IV bits and at the same time appear in another part of the cipher in another form (but not obeying a simple linear relation with IV bits). The following example illustrates that the key differentiation does not guarantee the elimination of the considered subset of key bits, but the degree and complexity of equations is significantly decreased.

Example 3 *Let us consider a toy cipher where the subset of key bits, say k_2, k_3, appears as linear combination with two IV bits, but also on a stand-alone basis. More precisely, let $k_1, \ldots, k_6$ and $IV_1, \ldots, IV_6$ after some t_1 rounds appear in the state of the cipher as follows,*

$$S^{t_1} = (s_1^{t_1}, \ldots, s_{12}^{t_1}) = (k_1, k_2, k_3, k_4, k_2 + IV_2, k_3 + IV_3, IV_1, IV_4, k_5, k_6, IV_5, IV_6).$$

Let also the following relations be valid after the KSA,

$$\begin{aligned} s_1^{(FIN)} &= s_1 s_3 s_5 s_6 + s_2 s_6 s_{10} s_{12} + s_3 s_5 + s_1 s_9 \\ s_2^{(FIN)} &= s_1 s_5 s_6 s_7 + s_2 s_3 + s_3 s_5 s_8 \\ &\vdots \\ s_{12}^{(FIN)} &= \ldots \end{aligned}$$

Then as before, let us assume that $z_1 = s_1^{(FIN)} + s_2^{(FIN)}$. Note that in the expressions for $s_1^{(FIN)}$ and $s_2^{(FIN)}$ all the terms that are important for the purpose of illustration of differentiation w.r.t. k_2 and k_3 do appear. Then we compute,

$$\begin{aligned} \Delta^{k_2} z_1 = z_1^{IV_2} + z_1^{IV_2+1} = \Delta^{s_5} z_1 &= s_1 s_3 s_6 + s_3 + s_1 s_6 s_7 + s_3 s_8 \\ \Delta^{k_3} z_1 = z_1^{IV_3} + z_1^{IV_3+1} = \Delta^{s_6} z_1 &= s_1 s_3 s_5 + s_2 s_{10} s_{12} + s_1 s_5 s_7 \\ \Delta^{k_2,k_3} z_1 = z_1^{IV_2,IV_3} + z_1^{IV_2+1,IV_3+1} = \Delta^{s_5,s_6} z_1 &= s_1 s_3 s_6 + s_1 s_3 s_5 + s_1 s_3 + s_2 s_{10} s_{12} + \\ &\quad s_3 + s_1 s_7 + s_1 s_6 s_7 + s_1 s_5 s_7 + s_3 s_8 \\ \bigoplus_{J \subset \{k_1, k_2\}} \Delta^J z_1 &= s_1 s_7 + s_1 s_3 \end{aligned}$$

What is interesting to observe is that we have successfully eliminated the presence of s_5 and s_6 as expected, but the state bits $s_1, \ldots, s_4$ (involving the key bits $k_1, \ldots, k_4$) may still be present in the expressions. Anyway the degree of equations is decreased and more importantly only the terms that contain s_5 and s_6 survive after differentiation. Thus in case there are no terms in z_1 that contain both s_5, s_6 and s_2, s_3 the effect of differentiation would be the same as previously discussed; the case when some key bits only appear as an XOR sum with some IV bits. Also the terms such as $s_3 s_5$ that only contain a subset of key bits used in the differentiation simply vanish.

In general, given a subset of key bits $k_{i_1}, \ldots, k_{i_m}$ that relate linearly to some IV bits $s_{j_1} = k_{i_1} + IV_{r_1}, \ldots, s_{j_m} = k_{i_m} + IV_{r_1}$ the differentiation of the keystream z_i is performed by identifying the terms that contain the largest subset of these state bits $s_{j_1}, \ldots, s_{j_m}$. Assuming the product $s_{j_1} s_{j_2} \cdots s_{j_m}$ is contained in monomials that represent keystream bit z_i, then after the differentiation the degree of z_i is reduced by m and the number of terms is significantly decreased. Then one may apply key guessing for further reduction of the degree, and guessing sufficiently many key bits we would again end up with linear equations. The only problem is that this technique is design dependent and in contrast to the scenario in the previous section, we cannot always guarantee the complexity beyond the exhaustive key search attack.

5 Some practical applications of key differentiation attack

In this section we analyze the possible application of our technique to those designs that satisfy the necessary assumption. To successfully apply the differential analysis our main requirement was a linear mixing of certain subset of IV and key bits. It turns out that many ciphers actually use such a linear mixing making them vulnerable to the key related attacks.

5.1 Case study: TRIVIUM

TRIVIUM [8] is a hardware oriented stream cipher that reached the final third phase of the eSTREAM project. Due to its simplicity and compact algebraic description it has been a popular target for cryptanalysis by a broad crypto community. However, TRIVIUM has resisted all the cryptanalytic attempts so far and the most efficient attacks proposed recently [18] is a key recovery attack on a reduced round TRIVIUM in the chosen IV scenario and reduced round TRIVIUM (the original KSA ibeing halved, consiting of 576 cycles instead of 1152 cycles), and an application of so-called cube attacks [9] that succeeds to recover the key for TRIVIUM reduced to 735 initialization rounds. In the original paper [8] the designers claim that 2 full cycles for the key/IV setup are sufficient to properly mix the key and IV bits so that each state bit after 576 iterations depends on each key and IV bit in a highly nonlinear manner.

TRIVIUM cipher 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 pre-setup state S^{ps} is completed. The key/IV setup is performed by clocking the cipher $4 \cdot 288 = 1152$ times without producing the output.

TRIVIUM seems to be a very suitable target for the application of key related attacks due to the following arguments.
– In the first place the first 66 output bits are linear combinations of six state bits and therefore the degree of these equations cannot be larger than the maximum degree of individual terms. Thus, if we can efficiently turn the state bit equations into linear by applying the key differentiation together with guessing some key bits, then 66 linear equations of the keystream are easily solved.
– To apply the above technique we must ensure the existence of linear relations among certain IV and key bits. Obviously this is not the case for the pre-setup state S^{ps} as there is no mixture between the key and IV bits. But 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 IV and key bits in a linear manner. For convenience we denote the state at time i as $S^i = (s_1^{(i)}, \ldots, s_{288}^{(i)})$ 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 this would give a rise to one linear relation. It is easy to verify that such a linear relations are obtained in the first 12 clocks,

$$s_{94}^{(i)} = k_{67-i} \oplus IV_{79-i}, \quad i = 1, \ldots, 12, \tag{4}$$

which leads to the following internal state after 12 clocks:

$$(s_1^{(12)}, \ldots, s_{93}^{(12)}, k_{55} \oplus IV_{67}, k_{56} \oplus IV_{68}, \ldots, k_{66} \oplus IV_{78}, s_{106}^{(12)}, \ldots, s_{288}^{(12)}). \tag{5}$$

5.1.1 Performing the attack

Assume at the moment that the key bits $k_{55}, \ldots, k_{66}$ and IV bits $IV_{67}, \ldots, IV_{78}$ are later mixed with remaining variables but their internal linear relation is always preserved, and in addition the key variables $k_{55}, \ldots, k_{67}$ only appear in this form during the initialization process. Then, through the differentiation of the IV bits at positions $67, \ldots, 78$ we get the key differential at positions $55, \ldots, 66$. Technically, the IV bits at positions $1, \ldots, 66$ and $79, 80$ are kept fixed, whereas the IV bits at positions $67, \ldots, 78$ are varied so that

the key differentiation is performed as explained before. Assuming we are able to handle the initial state equations after the KSA (which we can by specifying the IV and guessing certain number of key bits) we can get rid of the key bits $k_{55}, \ldots, k_{66}$ therefore reducing the complexity of the exhaustive search to $C^{w.c.} = 2^{80-12} = 2^{68}$.

Then, under the same assumption, we would successfully combine the key differentiation with the key guessing attack. Assume that we guess 60 key variables, thus leaving only 8 key variables in the algebraic expressions for the secret state bits (after the key variables $k_{55}, \ldots, k_{66}$ has been eliminated) to be determined. Most likely the algebraic expressions are only quadratic in remaining key variables, and since we have 66 linear keystream equations we would get a system of 66 equations with the maximum number of terms equal to $\binom{8}{2} + \binom{8}{1} = 36$. The system is heavily overdefined and we easily find a correct solution. In the case there are cubic terms in the expressions we would then guess 62 bits and the number of terms would be $\binom{6}{3} + \binom{6}{2} + \binom{6}{1} = 41$ which again results in an overdetermined system. Finally, by guessing 63 bits of the key we are assured that the system is overdefined. In this case there are only 5 key variables left and the total number of terms is at most 32, thus 66 equations from the keystream generation should suffice.

Nevertheless, TRIVIUM implementation **does not allow** this scenario due to the presence of the key bits $k_{55}, \ldots, k_{66}$ at positions $s_{67}^{(12)} = k_{55}, \ldots, s_{78}^{(12)} = k_{66}$. This fact has been overlooked by the authors in the original version of this paper. Thus, we can only apply the modified scenario of the attack and therefore only the degree/complexity reduction is guaranteed. Anyway, we describe the attack that has been implemented on TRIVIUM (due to extensive memory requirements only reduced round TRIVIUM 576 rounds has been considered). To further reduce the complexity of computation we have decided to formally use key guessing in our attack. This means that we select 60 bits of the key that we would "guess", and fix $IV_1, \ldots, IV_{66}, IV_{79}, IV_{80}$ to some arbitrary value (preferably all zero value for reduced complexity of expressions). Then, it is easily verified that the only active state positions after 12 clocks are $s_{67}, \ldots, s_{78}$, $s_{94}, \ldots, s_{105}$ and some $s_{i_1}, \ldots, s_{i_8}$, where $i_1, \ldots, i_8 \not\subset \{67, \ldots, 78\} \cup \{94, \ldots, 105\}$, depending on which 8 key variables have not been fixed (apart from $k_{55}, \ldots, k_{66}$).

Thus after the initialization process (reduced or not) the state bits are given as,

$$\begin{aligned} s_1^{(FIN)} &= f_1(s_{i_1}^{(12)}, \ldots, s_{i_8}^{(12)}, s_1^{(12)}, \ldots, s_{66}^{(12)}, s_{79}^{(12)}, \ldots, s_{288}^{(12)}, IV_{67}, \ldots, IV_{78}) \\ &\vdots \\ s_{288}^{(FIN)} &= f_{288}(s_{i_1}^{(12)}, \ldots, s_{i_8}^{(12)}, s_1^{(12)}, \ldots, s_{66}^{(12)}, s_{79}^{(12)}, \ldots, s_{288}^{(12)}, IV_{67}, \ldots, IV_{78}), \end{aligned}$$

Thus the TRIVIUM state bits after KSA are some complex function of 32 state bits after 12 clock cycles, and 12 IV bits that are later varied to perform differentiation with respect to $s_{94}^{(12)}, \ldots, s_{105}^{(12)}$. As previously remarked the differentiation is performed by finding the term that contains most of the state variables $s_{94}^{(12)}, \ldots, s_{105}^{(12)}$. Then everything depends on the way the complexity of expressions evolve during the initialization process. It is important to notice that the attack is general in the sense that only the ciphers (for which the attack is applicable) that perfectly mix the state variables in the Shannon's sense are then resistant to this kind of attack. For instance, if for some keystream bit we can collect the terms of the form,

$$s_{94}^{(12)} s_{95}^{(12)} \cdots, s_{105}^{(12)} f(s_{i_1}^{(12)}, \ldots, s_{i_8}^{(12)}, IV_{67}, \ldots, IV_{78}),$$

then we completely eliminate the presence of the key variables $k_{55}, \ldots, k_{66}$ through differentiation, thus only only f remains. In any case, we reduce the complexity of expressions and it might be of benefits to fix a few more bits to arrive at linear equations. This however requires a careful investigation, though a practical implementation of the attack is both time and memory demanding.

5.2 Case study: DECIM

DECIM-128 [2] was submitted as a hardware efficient cipher to the eSTREAM project. Though it was not successfully cryptanalyzed, as the originally proposed version called DECIM, it did not pass to the final phase of the project, but still it represents a strong design that is based on irregular decimation and nonlinear filtering. Nevertheless the initial loading of 128-bit key and 128-bit IV into the 288 bit register $S = (s_0, s_1, \ldots, s_{287})$ is performed as follows,

$$s_i = \begin{cases} k_i, & 0 \leq i \leq 127 \\ k_i \oplus IV_i, & 128 \leq i \leq 255. \end{cases}$$

For such an initial loading we are free to choose any subset of IV bits and to reduce the degree of equations by eliminating the dependency on the corresponding state variables. Since no decimation is involved during the initialization process which takes 1152 cycles the initial state of the cipher after the KSA can be determined (note that this might be totally infeasible as the state bits might contain as many as 2^{256} monomials - in such a case we would be forced to specify IV bits in advanced and even to guess some key bits). Then assuming that no decimation is involved in the keystream generation the attacker would proceed as in case of TRIVIUM, by simply observing the output keystream bits for different choices of IV bits.

Since the keystream is generated by passing the output of the nonlinear filter through the ABSG decimation mechanism we cannot determine for sure which state bits constitute the output. Thus additional guessing would be necessary and this would probably lead to the complexity above the exhaustive search.

6 Conclusions

In this paper we have shown that a linear mixing of the IV and key bits might lead to an efficient key recovery attack. The computer simulation concerning TRIVIUM and application of the attack to other designs apart from TRIVIUM and DECIM-128 are currently in progress. To circumvent this type of attack either one has to ensure that the cipher either optimally mixes the key and IV bits or the initial loading of the IV and key bits into the cipher should be performed in a nonlinear manner, thus slightly increasing the amount of clock cycles for the KSA procedure.

References

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

[2] C. et al. Berbain. DECIM-128. Avaliable on ECRYPT Stream Cipher Project page, 2005. http://www.ecrypt.eu.org/stream/decim.html.

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

[4] 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.

[5] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, vol. 4(1):3–72, 1991.

[6] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97*, volume LNCS 1294, pages 513–525. Springer-Verlag, 1997.

[7] 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.

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

[9] I. Dinur and A. Shamir. Cube attacks on tweakable black box polynomials. Cryptology ePrint Archive, Report 2008/385, 2008. http://eprint.iacr.org/.

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

[11] 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.

[12] 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.

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

[14] 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.

[15] A. Joux and F. Muller. A chosen IV attack against Turing. In *Selected Areas in Cryptography*, volume LNCS 3006, 2003.

[16] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—EUROCRYPT'93*, volume LNCS 765, pages 386–397. Springer-Verlag, 1993.

[17] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, Vol. 27:656–715, 1949.

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