RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 25 = 546 (2021): 33-49
DOI: `https://doi.org/10.21857/yvjrdcl0ey`

A NEW REPRESENTATION OF S-BOXES FOR ALGEBRAIC DIFFERENTIAL CRYPTANALYSIS

Alena Bednáriková and Pavol Zajac

Abstract. Algebraic cryptanalysis can be used to break (small versions of) block ciphers with small data complexity. If we have access to a large number of P-C pairs, algebraic cryptanalysis can be combined with differential techniques. Differential characteristic produces extra linear equations, which can be used to augment the original algebraic system. In our experiments with algebraic differential cryptanalysis, we have developed a different technique to represent the system. In our new method, we model a single P-C pair based encryption, but we use the differential to restrict the equations that model active S-boxes.

An algebraic system created with our new model is smaller, and can theoretically be solved faster. Our experiments show that the advantage depends on the overall number of P-C pairs available and whether the chosen differential characteristic is correctly estimated. One of the advantages of the new method is that it can use partial information from the differential and still determine a correct solution faster than both the standard algebraic attack and the standard algebraic-differential attack.

1. Introduction

Algebraic cryptanalysis can be used to break (small versions of) block ciphers with small data complexity [3, 4]. The main principle of algebraic cryptanalysis is simple: Encryption is described by a set of equations between bits of plaintexts, ciphertexts, the unknown key, and inner states of the encryption algorithm. This set of equations is then solved by a suitable fast solver. SAT solvers can be combined with key bit guessing and massively parallel computing [7] to solve even relatively large systems.

In a recent article [2], Andrzejczak and Dudzic attack smaller versions of block ciphers SIMON and SPECK. Instead of modeling the whole cipher, they do not model the key expansion algorithm, and instead, try to find independent subkeys. This requires more plaintext-ciphertext (P-C) pairs than the standard algebraic cryptanalysis. Unfortunately, with a growing

2020 *Mathematics Subject Classification.* 94A60, 14G50.
Key words and phrases. Algebraic differential cryptanalysis, S-boxes.

number of P-C pairs, the size of the system quickly increases, which increases the solving time.

If we have access to a large number of P-C pairs, algebraic cryptanalysis can be combined with differential techniques [1,5,10]. The attack is based on a selected differential characteristic, which holds with high probability. This characteristic produces extra linear equations, which can be used to augment the original algebraic system. Based on this augmented system, the algebraic solver can detect, whether the differential characteristic holds for a particular P-C pair, and to derive information about key bits.

In our research, instead of trying to break some specific cipher, we try to understand empirically the effect of having multiple P-C pairs, and of applying differential techniques on a simple Substitution Permutation Network model. In our experiments we use SAT representation and SAT solver CryptoMiniSat [8] integrated within SAGE [9].

In our experiments with algebraic differential cryptanalysis, we have developed a different technique to represent the system. The standard model produces a system of equations for each tuple of P-C pairs (supposedly connected by a differential characteristic) as a union of equations for encryption F_1, and F_2, along with linear equations describing a chosen differential. In our new method, we model single encryption (only one of each 2 P-C pairs), but we use the differential to restrict the equations that model active S-boxes. Suppose that differential characteristic goes through some S-box with input difference Δ_x and output difference Δ_y. We replace the original S-box equation, which has the solution set $\{(x, y); S(x) = y\}$, with the new equation with the solution set $\{(x, y); S(x) = y \wedge S(x + \Delta_x) = y + \Delta_y\}$. The number of additional clauses that express new restrictions based on differences is smaller than in the standard model. The important information about the chosen differential (equations on active S-boxes) is preserved (as well as the original solution of the whole cipher, if we use enough P-C pairs to avoid false keys).

2. Preliminaries

Algebraic cryptanalysis focuses on breaking cryptographic schemes with algebraic methods. There are two main tasks involved: representing cryptanalytic problem by a set of equations and then solving the corresponding system. Although there are many types of algebraic attacks, we will focus mainly on a simple key recovery from a known set of plaintext-ciphertext pairs.

Let us have an encryption scheme with (efficiently computable) encryption function $Enc : \mathbb{Z}_2^{n_B} \times \mathbb{Z}_2^{n_K} \to \mathbb{Z}_2^{n_B}$. A key recovery problem is a problem of determining unknown key $k \in \mathbb{Z}_2^{n_K}$ from a (parametric) set of equations in the form

$$Enc(x_i, k) = y_i, \tag{2.1}$$

where $(x_i, y_i) \in \mathbb{Z}_2^{n_B} \times \mathbb{Z}_2^{n_B}$ are pairs of plaintext and ciphertext pairs (P-C pairs in short). The number of P-C pairs required to determine correct encryption key depends on the relation between n_K and n_B and properties of the encryption function. For us, however, we are mainly interested how fast a specified solver can solve system of equations (2.1): determine any solution k, or determine that no such k exists.

2.1. *Substitution-permutation network.* Instead of studying various concrete cipher designs, we focus on a simpler model. We use a substitution-permutation network (SPN). An SPN is a key alternated iterative cipher where each round consists of a layer of S-box substitution and a simple permutation of bits.

An S-box is a (non-linear) bijective vectorial Boolean function $S : \mathbb{Z}_2^m \to \mathbb{Z}_2^m$. We will use a substitution layer $\sigma : \mathbb{Z}_2^{n_B} \to \mathbb{Z}_2^{n_B}$ constructed as a brick-layer permutation from a single S-box S. This means that the input of σ is split into m-bit blocks. The S-box function is applied independently to each of these blocks. Finally, the results are concatenated back and form a result of function σ.

Let P be a $n_B \times n_B$ permutation matrix. The permutation layer $\pi : \mathbb{Z}_2^{n_B} \to \mathbb{Z}_2^{n_B}$ is defined by $\pi(x) = P \cdot x^T$. This layer is linear, and essentially just changes the order of bits during the encryption.

Substitution-permutation network is a composition of $r - 1$ round functions defined by round transformations $\rho_i : \mathbb{Z}_2^{n_B} \to \mathbb{Z}_2^{n_B}$, $\rho_i(x) = \pi(\sigma(x \oplus k_i))$, and one final round transformation $\rho_r = \sigma(x \oplus k_r) \oplus k_{r+1}$. Constants k_i, for $i = 1, 2, \ldots, r + 1$, are computed by a key schedule $\kappa : \mathbb{Z}_2^{n_K} \to \mathbb{Z}_2^{(r+1)\cdot n_B}$.

In our experiments we use a 4-round SPN with $n_B = 16$, $m = 4$ with variable S-boxes, and a fixed permutation layer given by bit permutation $(1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15, 4, 8, 12, 16)$. As a key schedule, we use either repetition of a 16-bit key (for $n_K = 16$), 5 independent 16-bit subkeys (for $n_K = 80$), or a sequence of low 16-bits from a shifted 32-bit key (for $n_K = 32$).

2.2. *Algebraic attack on SPN.* A general equation system (2.1) can be instantiated by selecting SPN as an encryption function. Each equation $SPN(x_i, k) = y_i$ can be further rewritten as a system of simpler equations by introducing intermediate variables: input (denoted by u_i), and output (denoted by v_i) bit vectors of S-box layers in each SPN round. In each round, there are n_B/m non-linear equations of the form

$$S(u_{i,km+1}, \ldots, u_{i,km+m}) = v_{i,km+1}, \ldots, v_{i,km+m},$$

along with linear equations between bits of v_i, u_{i+1} and k_{i+1}. Specific linear equations are also added for the first and last round that connect S-box inputs/outputs with the first and last subkey and plaintext/ciphertext bits.

We can use many different representations of the algebraic problem, and then different solvers to find a solution. In our experiments, we use a CNF representation and a SAT solver, with details described in Section 3.

In general, solving the system is believed to be difficult, as it is based on the NP-hard problem of solving a system of non-linear equations over $\mathbb{Z}_2$. For random non-linear equation systems, we only have algorithms with average complexity exponential in the size of the system. However, in the case of algebraic cryptanalysis, we know that the complexity is upper bounded by 2^{n_K} verifications of a solution because we can use an exhaustive search through the keyspace to find a correct solution or to show there is no key, for which the system has a solution.

Note that experiments show that in practice the solution of the system can be found by a solver faster than by exhaustive search. We will consider an algebraic attack to be successful if it is faster than an exhaustive search on average for a random key selection.

Intuition tells us that extra information obtained by adding more equations from more P-C pairs should lead to faster attack methods. However, for a generic solver, extra equations and variables mean extra work, as the system is larger, and the solver requires more memory and computation steps to find a solution (or a conflict indicating that no solution exists). If we have a large number of P-C pairs available, we should consider statistical methods to extract the extra information provided by this amount of available data.

2.3. *Differential cryptanalysis.* Differential cryptanalysis studies attacks based on differential distinguishers. Let us consider a specific set of P-C pair tuples $\{(x_i, y_i), (x_i^*, y_i^*)\}$ with a fixed input difference Δ_x, e.i., $x_i \oplus x_i^* = \Delta_x$ for each i. A secure encryption function should be indistinguishable from a random function. This implies that the set of output differences $\Delta_{y,i} = y_i \oplus y_i^*$ should have a uniform random distribution, with $Pr(\Delta_{y,i} = \Delta_y) = 2^{-n_B}$ for each choice of Δ_y. A differential distinguisher arises, when some choice of input difference Δ_x produces some output difference Δ_y with significantly higher probability $p \gg 2^{-n_B}$. We call a pair (Δ_x, Δ_y) a differential, and denote its differential probability with $p_{\Delta_x,\Delta_y} = Pr(\Delta_y/\Delta_x)$ (for a randomly chosen P-C pairs tuples with fixed difference Δ_x).

For an SPN, we can find a differential with high differential probability by analysing S-box differentials, and concatenating S-box differentials through affine layers of SPN. An S-box differential is again a pair (Δ_u, Δ_v), with its probability $p_{\Delta_u,\Delta_v} = Pr(S(u \oplus \Delta_u) \oplus S(u) = \Delta_v)$, for a random choice of S-box input u. To construct SPN differential, we select suitable input differential, and follow it through affine layers of the cipher, using the fact that $\Delta_y = Mx \oplus k \oplus Mx^* \oplus k = M(x \oplus x^*) = M\Delta_x$ (with probability 1). On

S-box layer, each S-box with non-zero input difference[1] we select a suitable S-box differential with high differential probability. We call these S-boxes active S-boxes. An r-round differential trail is a combination of S-box differentials and corresponding linear transformations provided by affine layers, leading to an r-round differential (Δ_x, Δ_y). Probability p_{Δ_x,Δ_y} can be lower bounded by a product of S-box differential probabilities, if we assume S-box differentials are independent.

2.4. *Algebraic-Differential cryptanalysis.* We can associate a set of linear equations in state bits $\Delta_{u_{i,j}} = u_{i,j} \oplus u^*_{i,j}$ with each differential trail. These equations hold if the encryptions used to produce P-C pair tuple $\{(x, y), (x^*, y^*)\}$ followed exactly the selected differential trail, and can again be approximated as a product of differential probabilities on active S-boxes.

Albrecht et al. in [1] introduced a notion of algebraic-differential cryptanalysis. Basic attack (method A) can be summarized in the following steps:

1. Identify a suitable differential trail with associated differential probability $p \gg 2^{-n_B}$.
2. Construct a (parametric) system of equations for the encryption of a tuple of P-C pairs with the same key, $Enc(x, k) = y$, and $Enc(x^*, k) = y^*$. Add linear equations corresponding to an identified differential trail $\Delta_{u_{i,j}} = u_{i,j} \oplus u^*_{i,j}$.
3. For each P-C pair tuple $\{(x_i, y_i), (x^*_i, y^*_i)\}$, try to solve the equation system. With probability p, the system provides a solution k, otherwise it is rejected (due to the added linear equations).

We can limit the execution of the last step to such P-C pair tuples, which satisfy differential (Δ_x, Δ_y), in which case solution is found with the probability of the selected differential trail being used to produce the final differential. We can generalize the technique to use truncated differentials (that are not fully determined) to produce only a partial set of linear equations between bits of internal variables. In our attacks, we either use full differential trails or truncated differentials that imply linear equations only bits on the first and last layer of S-boxes (allowing any differential trail in between).

Suppose that the average time required to solve a system of algebraic equations $Enc(x, k) = y$, and $Enc(x^*, k) = y^*$ is T_0. Suppose that an average time to solve the system with added linear equations corresponding to a (truncated) differential trail with probability p is $T_s < T_0$, and an average time to reject solution of such system is T_r. Algebraic-differential cryptanalysis is preferable to a standard algebraic cryptanalysis, if $p^{-1}T_r + T_s < T_0$.

[1]A zero input difference corresponds to a zero output difference with probability 1.

3. A new representation of the equation system

Algebraic cryptanalysis requires a suitable representation of an equation system and a solver related to the chosen representation. In our experiments, we have chosen a CNF representation with SAT solvers for easy comparison with results from different authors. Note that the technique presented in this section can be easily adapted to other different representations, such as ANF, or an MRHS representation.

System of equations $Enc(x, k) = y$ is represented in a conjunctive normal form (CNF) with a standard conversion based on natural bijection between $\mathbb{Z}_2$ and Boolean ring. Variables from $\mathbb{Z}_2$ are associated with literals, with literal x_i being true if and only if variable $x_i = 1$. For the sake of simplicity we will use the same notation for Boolean variables and $\mathbb{Z}_2$ variables, and freely associate value 1 with true and 0 with false.

We transform each equation in the system into a corresponding formula, which is true if and only if each of the equation solutions corresponds to a satisfiable setting of literals for the formula. All partial formulas are then joined with a conjunction. If each equation in the system is represented by a CNF, then the final system representation is also in a CNF.

Let $S : \mathbb{Z}_2^m \to \mathbb{Z}_2^m$ be an S-box used in a substitution permutation network. We can construct a Boolean function $F_S : \mathbb{Z}_2^m \times \mathbb{Z}_2^m \to \mathbb{Z}_2$, with $F_S(u_1, \ldots, u_m, v_1, \ldots, v_m) = 1$ if and only if $S(u_1, \ldots, u_m) = (v_1, \ldots, v_m)$. Equation $S(u) = v$ is transformed into a CNF by computing a corresponding CNF formula which is true if and only if $F_S(u, v) = 1$.

Similarly, we can represent each linear equation in the system to a CNF by associating it with a similar Boolean function. However, in our chosen SPN, we only have linear equations with 2 or 3 variables. These can be transformed into CNF using $x \oplus y = z$ if and only if the following formula is true

$$(x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z).$$

In the basic Albrecht A method for each P-C pair tuple we need to join three CNF's: the first two are constructed from $Enc(x, k) = y$, and $Enc(x^*, k) = y^*$, respectively. Final CNF is constructed from linear equations corresponding to a selected differential trail. Using the previous logic formula, for each non-zero bit in the selected differentials, we need 4 clauses of 3 literals each. The size of this extra formula is smaller than each of the encryption formulas, but it still increases the input size of the system that must be processed by a SAT solver.

It would be beneficial to optimize the formulas before we attempt the solution. Note, however, that differential formulas are connected to both of the encryption formulas. We have instead chosen a different approach. We do not use both encryption formulas, along with differential equations, but instead, try to provide extra information from differential trail directly to the encryption formula. Thus, we only use a single ”enhanced” encryption formula for a single P-C pair, with extra information from differential trail contributing extra constraints on possible S-box values.

X	Y	$\Delta X = 1011$	$\Delta Y = 0010$
		X^*	Y^*
0000	1110	1011	1100
0001	0100	1010	0110
0010	1101	1001	1111
0011	0001	1000	0011
0100	0010	1111	0000
0101	1111	1110	1101
0110	1011	1101	1001
0111	1000	1100	1010
1000	0011	0011	0001
1001	1010	0010	1000
1010	0110	0001	0100
1011	1100	0000	1110
1100	0101	0111	0111
1101	1001	0110	1011
1110	0000	0101	0010
1111	0111	0100	0101

FIGURE 1. New S-box representation construction. Lines with green background show an example of S-box entries that remain in the reduced table of valid input-output pairs (their I/O differences are fulfilled). Red background denotes an example of input-output pairs that are removed due to an I/O difference mismatch.

Suppose that we have selected a differential trail with some active S-box S and the corresponding S-box differential (Δ_u, Δ_v), with $\Delta_u \neq 0$. Equation $S(u \oplus \Delta u) \oplus S(u) = \Delta_v$ holds for only a limited number of values u (depending on S-box differential probability). Consider an S-box with a truth table depicted in Fig. 1. If we select differential $(1011, 0010)$, input values $0000, 1011$ lead to a required input and output difference combination. On the other hand inputs $0010, 1001$ do not lead to a required output difference, given the input difference. In standard S-box representation we use function F_S, which is true on each combination of input-output values. In the new representation, we instead construct a Boolean function $G_{S,\Delta_u,\Delta_v} : \mathbb{Z}_2^m \times \mathbb{Z}_2^m \to \mathbb{Z}_2$ (or simply G), such that $G_{S,\Delta_u,\Delta_v}(u_1,\dots,u_m,v_1,\dots,v_m)=1$ if and only if

$$S(u)=v, \text{ and } S(u\oplus\Delta_u)\oplus S(u)=\Delta_v.$$

Now we construct an equation system for $Enc(x,k)=y$ and associated differential trail together by replacing standard (F) CNF representation of active S-boxes with the restricted (G) CNF representation of active S-boxes.

If we select any of the two P-C pairs that correspond to a satisfied differential trail, the associated CNF will be satisfiable (as each S-box differential is covered correctly). On the other hand, if we select incorrect P-C pair (one that does not fulfill the differential trail exactly), the associated restricted CNF can still have a solution. This is because we do not use the second P-C pair and exact linear equations for differentials. The probability of obtaining a solution given an incorrect pair depends on the correspondence between intersection of supports for G_{S,Δ_u,Δ_v} for different differentials (Δ_u,Δ_v). With an increasing number of active S-boxes, this probability quickly decreases. In practice, this means that with differential trails with a small number of active S-boxes we have a higher probability of solving the system than the one predicted by differential trail probability, and the data complexity of the algebraic-differential cryptanalysis decreases. The exact probabilities must be computed for each concrete attack, as they depend on selected S-boxes and S-box differentials.

4. Experimental results

We have experimentally verified the effectiveness of algebraic and algebraic-differential cryptanalysis on a 4-round S-P network using mathematical software SAGE [9]. Equation systems were uniformly generated by our software. We have measured the solution time as an average of the time required to solve a specified number of randomly generated instances. However, we were not interested in solving time itself. Instead, we were interested in the overall ratio of the solving time in a specified set of scenarios.

4.1. *Influence of the differential information on solving time.*

In the first set of experiments, we examined several factors. We were interested in the question, whether adding differential information can affect the complexity of the solution. Furthermore, we investigated how the change of differential trajectories, and S-boxes, affects the solving time of the system.

For each S-box mapping, we created three different systems of clauses with two P-C pairs computed for 20 randomly generated keys:

1. A system with randomly generated, unrelated P-C pairs for a pure algebraic attack.
2. A system that contained specific P-C pairs which all satisfied a chosen differential characteristic. This system did not contain differential clauses.

3. A system of clauses with specific P-C pairs which all satisfied a chosen differential characteristic. This system contained a different representation of S-box clauses with additional differential information based on our proposed technique of algebraic-differential attack.

In each run of the experiment, we considered a different differential characteristic. Each characteristic had a differential probability $p > 2^{2-n_B}$, so the probability of a differential characteristic was large enough to break the cipher by differential cryptanalysis.

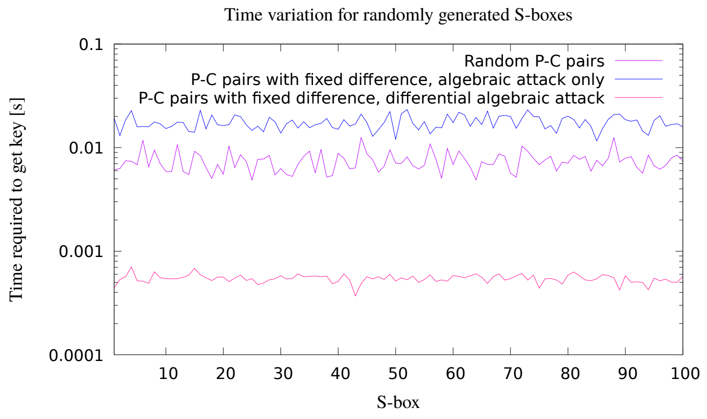

FIGURE 2. Comparison of algebraic-only and algebraic-differential attack depending on S-box selection.

The average time required to obtain an 80-bit key depending on randomly generated S-boxes for three modeled systems is shown in Figure 2.

We can see that the fastest way to obtain a correct solution is solving the third system of clauses, which includes additional differential information. Basic algebraic cryptanalysis takes a longer time, regardless of whether P-C pairs are random, or whether they are related by a given differential.

The graph in Figure 3 shows the computation time of an 80-bit key for a system of clauses that uses information from differential cryptanalysis depending on S-box mappings. We can see, that the solving time for individual S-box choices is very similar. Systems with arbitrarily chosen differential characteristics, that satisfy a certain differential probability can be solved at about the same time.

4.2. *A comparison with standard algebraic differential cryptanalysis.* As a demonstration of our results, we have prepared a concrete comparison for a selected SPN setting and differential characteristic. We base our settings and

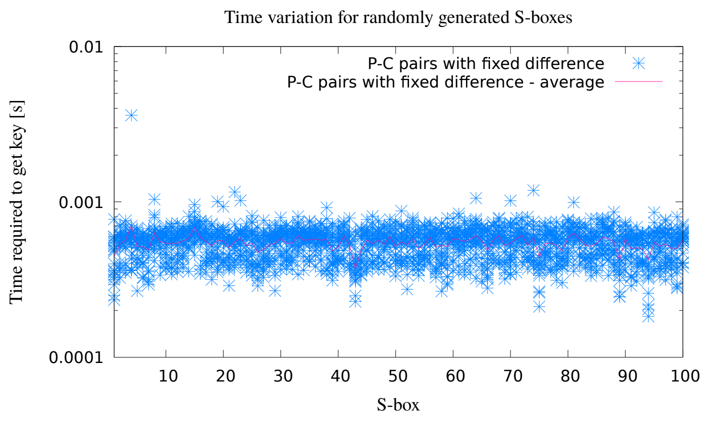

FIGURE 3. Comparison of algebraic-differential attack.

trajectories on a well-known tutorial on linear and differential cryptanalysis [6]. The S-box setting is presented in Table 1.

TABLE 1. The S-box selected for the experiment.

Input	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

The full differential characteristic and truncated characteristic used in experiments are presented in Figure 4. The truncated characteristic uses active S-boxes only from the first and the last rounds. The selected differential trail has an estimated differential probability $p_1 = 0.00165$, and the truncated version has differential probability $p_2 = 0.03125$.

We have compared the effectiveness of our system representation with the standard one presented by Albrecht and Cid in [1]. Specifically, we compared the time complexity of finding the key (or rejecting the system) depending on the number of P-C pairs. We have separately tested cases when the system of clauses contains correctly or incorrectly selected P-C pairs related to the preselected full or truncated differential trail.

The basic system covering a tuple of P-C pairs for 4 round SPN consisted of 2368 clauses. Our representation reduced this to 1216 in case of truncated difference and 1246 in case of full difference. Albrecht's representation required 2392 (truncated), and 2416 clauses (full), respectively.

For all examined cases, we have randomly generated 20 keys of 16-bits, 32-bits, and 80-bits, respectively. For each of them, we have constructed a system of clauses with the appropriate number of suitably selected P-C pairs.

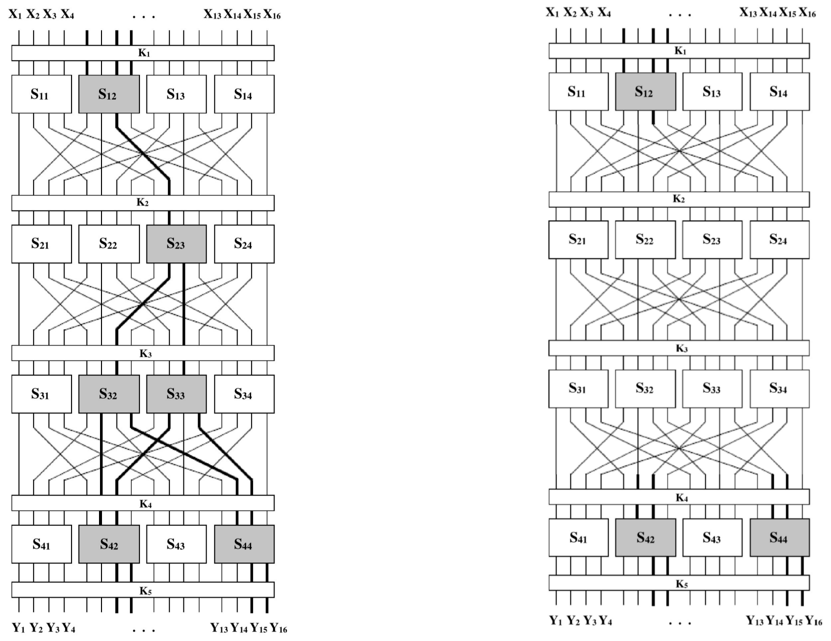

FIGURE 4. Full (left) and truncated (right) differential characteristic used in the demonstration experiments.

A comparison of our method with Albrecht method for correctly selected P-C pairs is presented in Figures 5, 6 and 7. We can observe that the system of clauses for each technique can be solved on average faster if we know the whole differential characteristic, followed by truncated differential, and then a system with no extra differential knowledge.

Our method outperforms the Albrecht A method in all cases with a full differential trail, but the distinction is more pronounced for a small 16-bit key. When using 80-bit keys, Albrecht's method is slightly faster for a system with an average number of P-C pairs. This might be caused by the fact that our system can produce more false solutions for a given tuple of P-C pairs, and thus the solver follows false trails for a longer time.

The case of incorrectly selected P-C pairs is shown in Figures 8, 9 and 10. For large systems, the solver can disprove the existence of the solution for the system of clauses with truncated differences significantly faster than finding a solution when it exists. With the Albrecht method, the solution of the system is an empty set in all cases even for smaller systems.

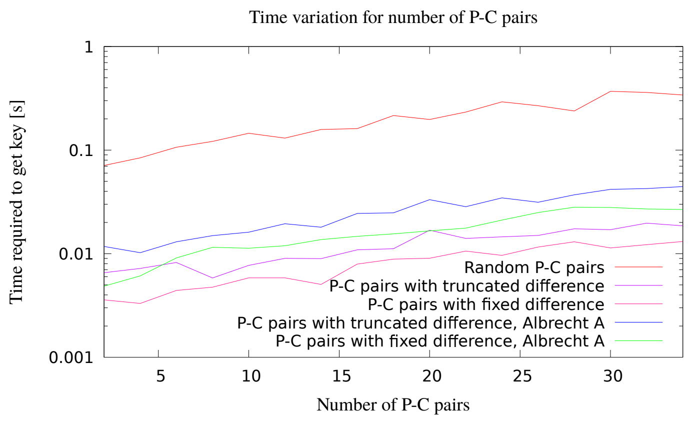

FIGURE 5. Comparison of our new method with Albrecht method, correctly selected P-C pairs, 16-bit key.

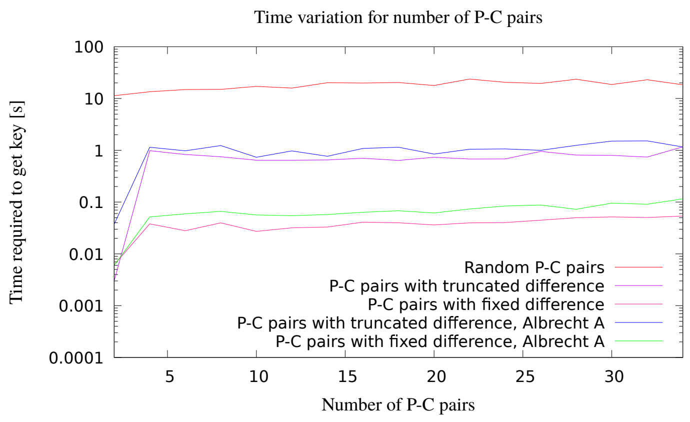

FIGURE 6. Comparison of our new method with Albrecht method, correctly selected P-C pairs, 32-bit key.

With our model, it is possible to find a (possibly false) solution with a small number of P-C pairs, so the average time complexity is higher in these cases (because we average a possibly shorter time needed to find a conflict in some instances, with a longer time needed to find a solution in other instances). With the increasing number of P-C pairs, even our method can only find an empty solution. When no solutions can be found, we can see that the computation of the system in case of incorrect P-C pairs can get

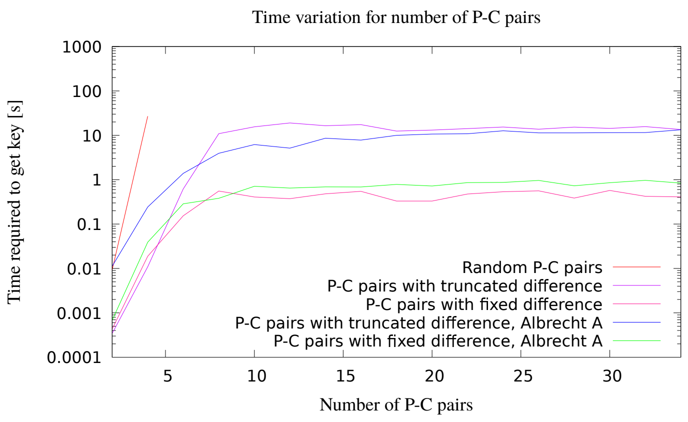

Figure 7. Comparison of our new method with Albrecht method, correctly selected P-C pairs, 80-bit key.

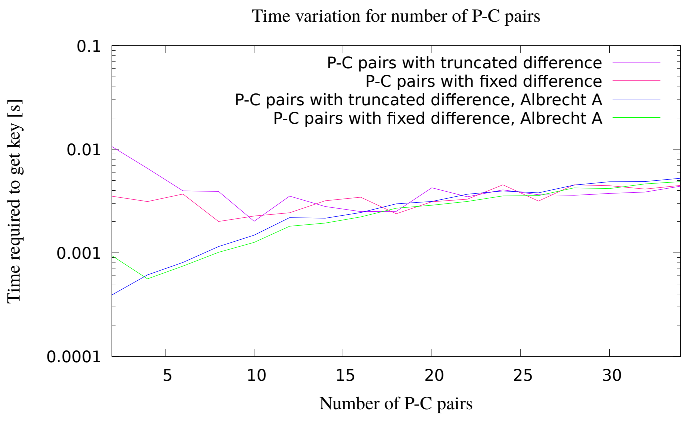

Figure 8. Comparison of our new method with Albrecht method, incorrectly selected P-C pairs, 16-bit key.

faster than the computation by the Albrecht method. However, the inability to quickly distinguish between correct and incorrect P-C pairs is detrimental to the overall performance.

As an example let us compare concrete numbers for the attack on 4 round SPN with a 16-bit key. The solver can find a key on average in 71 ms without differential information. The system with added truncated differentials

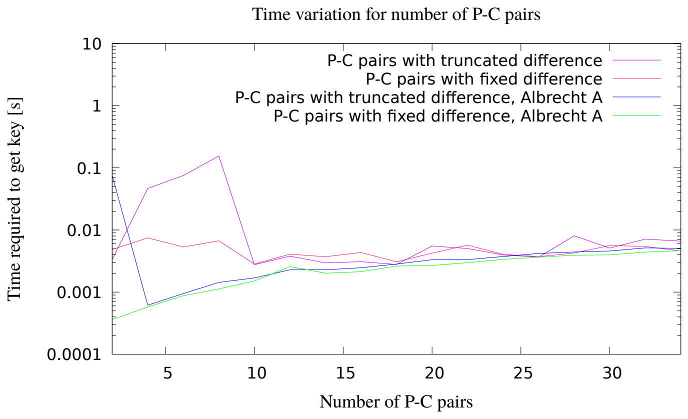

FIGURE 9. Comparison of our new method with Albrecht method, incorrectly selected P-C pairs, 32-bit key.

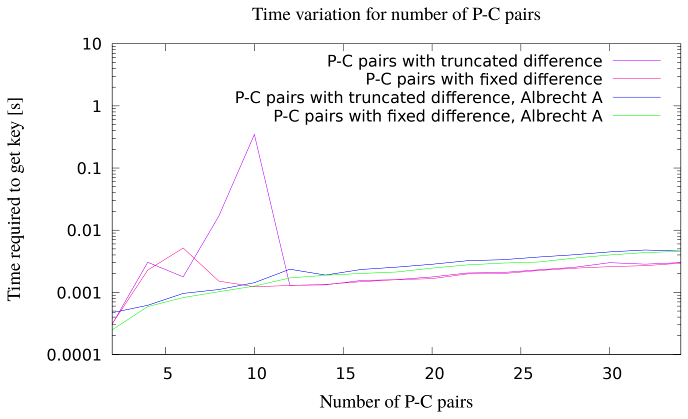

FIGURE 10. Comparison of our new method with Albrecht method, incorrectly selected P-C pairs, 80-bit key.

requires 8.2 ms with our representation and 11.7 ms with Albrecht's representation. Adding full differentials reduces time to 3.6 ms with our representation and 4.9 with Albrecht's representation. These comparisons scale to larger systems as well.

However, if we switch to an incorrect input P-C pair tuple, our representation leads to rejection in 10.6 ms for reduced, and 3.5 ms for full differential information. This is essentially the same as the time required to solve the system for the correct P-C pair tuple. Note that in 1 of 20 cases, the correct key was found even from the incorrect P-C pair tuple. On the other hand, with Albrecht's representation, we can reject incorrect P-C pair tuple faster, 3.5 ms for the truncated case, and 0.9 for the full case. In Albrecht's case, the overall time to reject incorrect P-C pairs (based on expected differential probability) is then faster than the full solution required in the truncated case, 12.5 ms. In other examined cases of small systems with short keys, the full time expected to reject all incorrect P-C pairs makes standard algebraic cryptanalysis, without differential information, faster overall.

This however changes for instances with larger keys. The differential information speeds the key search significantly. On the other hand, if we use a single P-C pair tuple, we will likely get just a false solution. When planning an attack on a concrete cipher, some trade-offs must be carefully considered. We suggest that Albrecht's method should be used to quickly reject incorrect pairs, running in parallel with our method. If the P-C pair tuple is not rejected in some short time, a (potential) solution can be found faster with our method. Once a system for a single P-C pair tuple is shown to be solvable (with any potential key), we can save the P-C pair tuple as a potentially correct P-C pair tuple. We can then incrementally append information from other P-C pairs and extend the system until a correct key is found.

5. Conclusions

We have presented a new representation of an algebraic system suitable for algebraic differential cryptanalysis. The system created with our new model is smaller, and can theoretically be solved faster. Our experiments show that the advantage depends on the overall number of P-C pairs available and whether the chosen differential characteristic is correctly estimated. One of the advantages of the new method is that it can use partial information from the differential, and still determine a correct solution faster than both standard algebraic attack, and standard algebraic-differential attack. On the other hand, our method requires more time to identify incorrect P-C pair tuple. Thus, in an experimental setup, we recommend running it in parallel with Albrecht's method: we expect that for correct P-C pair candidates our method provides a solution faster, while incorrect P-C pair candidates are quickly filtered by a solver using Albrecht's representation.

Note that in this article we have examined only applications to basic algebraic differential attacks. However, the new representation can easily be adapted for other types of attacks that provide additional information about the S-box inputs or outputs. It can also be used to model extra information obtained from the side-channel analysis, to combine side-channel attacks with algebraic cryptanalysis.

Acknowledgements.

This research was sponsored by Slovak Republic under grant VEGA 1/0159/17.

References

[1] M. Albrecht and C. Cid, *Algebraic techniques in differential cryptanalysis*, in: International Workshop on Fast Software Encryption, Lecture Notes in Comput. Sci. **5665**, Springer, Berlin, 2009, pp. 193–208.

[2] M. Andrzejczak and W. Dudzic, *SAT attacks on ARX ciphers with automated equations generation*, Infocommunications **9(4)** (2019), 2–7.

[3] N. T. Courtois and G. V. Bard, *Algebraic cryptanalysis of the data encryption standard*, in: Cryptography and coding, Lecture Notes in Comput. Sci. **4887**, Springer, Berlin, 2007, pp. 152–169.

[4] N. T. Courtois and J. Pieprzyk, *Cryptanalysis of block ciphers with overdefined systems of equations*, in: Advances in Cryptology – ASIACRYPT 2002, Lecture Notes in Comput. Sci. **2501**, Springer, Berlin, pp. 267–287.

[5] J.-C. Faugère, L. Perret and P.-J. Spaenlehauer, *Algebraic-differential cryptanalysis of DES*, in: Western European Workshop on Research in Cryptology-WEWoRC, Graz, 2009, pp. 1–5.

[6] H. M. Heys, *A tutorial on linear and differential cryptanalysis*, Cryptologia **26** (2002), 189–221.

[7] V. Hromada, L. Öllős and P. Zajac, *Using SAT solvers in large scale distributed algebraic attacks against low entropy keys*, Tatra Mt. Math. Publ. **64** (2015), 187–203.

[8] M. Soos, K. Nohl and C. Castelluccia, *Extending SAT solvers to cryptographic problems*, in: Theory and Applications of Satisfiability Testing – SAT 2009, Lecture Notes in Comput. Sci. **5584**, Springer, Berlin, 2009, pp. 244–257.

[9] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 9.0), 2020, `https://www.sagemath.org`.

[10] M. Wang, Y. Sun, N. Mouha and B. Preneel, *Algebraic techniques in differential cryptanalysis revisited*, in: Australasian Conference on Information Security and Privacy, Lecture Notes in Comput. Sci. **6812**, Springer, Berlin, 2011, pp. 120–141.

Novi prikaz S-kutija za algebarsku diferencijalnu kriptoanalizu

Alena Bednáriková i Pavol Zajac

Sažetak. Algebarska kriptoanaliza može se koristiti za razbijanje blokovne šifre male složenosti podataka. Ako imamo pristup velikom broju P-C parova, može se kombinirati algebarska kriptoanaliza i diferencijalne tehnike. Diferencijalna karakteristika daje dodatne linearne jednadžbe, koje se mogu koristiti za uvećanje izvornog algebarskog sustava. U našim eksperimentima s algebarskom diferencijalnom kriptalizom razvili smo drugačiju tehniku predstavljanja sustava. U našoj novoj metodi, modeliramo šifriranje zasnovano na paru P-C, ali koristimo diferencijal za ograničavanje jednadžbi koje modeliraju aktivne S-kutije. Algebarski sustav stvoren s našim novim modelom je manji i može se teoretski riješiti brže. Naši eksperimenti pokazuju da prednost ovisi o ukupnom broju dostupnih P-C parova i o tome je li odabrana diferencijalna karakteristika ispravno procijenjena. Jedna od prednosti nove metode je da može koristiti djelomične informacije iz diferencijala i odrediti ispravno rješenje brže od standardnog algebarskog napada i standardnog algebarsko-diferencijalnog napada.

Alena Bednáriková
Slovak University of Technology in Bratislava
Bratislava, Slovakia
E-mail: `albednarikova@gmail.com`

Pavol Zajac
Slovak University of Technology in Bratislava
Bratislava, Slovakia
E-mail: `pavol.zajac@stuba.sk`

Received: 7.10.2020.
Revised: 15.12.2020.
Accepted: 19.1.2021.