Computing and Informatics, Vol. 37, 2018, 349–366, doi: 10.4149/cai_2018_2_349

USING LOCAL REDUCTION FOR THE EXPERIMENTAL EVALUATION OF THE CIPHER SECURITY

Pavol Zajac

Institute of Computer Science and Mathematics
Slovak University of Technology
Ilkovičova 3
812 19 Bratislava, Slovakia
e-mail: `pavol.zajac@stuba.sk`

Abstract. Evaluating the strength of block ciphers against algebraic attacks can be difficult. The attack methods often use different metrics, and experiments do not scale well in practice. We propose a methodology that splits the algebraic attack into a polynomial part (local reduction), and an exponential part (guessing), respectively. The evaluator uses instances with known solutions to estimate the complexity of the attacks, and the response to changing parameters of the problem (e.g. the number of rounds). Although the methodology does not provide a positive answer ("the cipher is secure"), it can be used to construct a negative test (reject weak ciphers), or as a tool of qualitative comparison of cipher designs. Potential applications in other areas of computer science are discussed in the concluding parts of the article.

Keywords: Algebraic cryptanalysis, local reduction, method of syllogisms, SAT solvers

Mathematics Subject Classification 2010: 94A60, 14G50

1 INTRODUCTION

The algebraic cryptanalysis is based on the idea of transforming cryptanalytic problem to a problem of solving a carefully crafted set of equations (with algebraic methods). Algebraic attacks can be executed even if the attacker has only a very small number of P-C pairs. On the other hand, it seems quite difficult to scale the attacks (using SAT solvers, or Gröbner basis solver), or to correctly compare results in assessing the strength of cryptographic primitives.

Nowadays, two main approaches are prevalent in the algebraic cryptanalysis:

1. Transform the cryptanalytic problem to CNF form, and apply fast SAT solvers.
2. Find as many linearly independent MQ equations as possible, and apply relinearization to find solutions [5], or compute solutions by computing the Gröbner basis of the ideal generated by these equations [9, 10].

These methods usually terminate in reasonable time only if some number of key bits is fixed in advance. Moreover, Gröbner basis approach can often fail due to memory constraints.

A different approach to algebraic cryptanalysis was proposed by Raddum and Semaev [15]. An encryption process can be described by a sequence of smaller operations working on some internal state. At the bottom level, the cipher can be described by a logic circuit, with logic gates, and wires. We can assign a variable for each internal value transmitted on the wires during the encryption process. Then each logic gate defines a Boolean equation in its input and output variables. It is easy to enumerate all possible (partial) solutions for Boolean equations with small number of variables (sparse equations). If we descend to a reasonably low level in cipher description, each of these equations is 3-sparse, i.e., it has at most three variables (two inputs, and one output). It might be more practical to work on higher level, e.g. on the full S-box level [13]. The equations taken together with a known P-C pair provide an equation system for the algebraic cryptanalysis. We primarily want to find the value of key bits, but we can consider all intermediate values as unknowns in the system.

We represent equation system as algebraic varieties, resp. as a list of points of these varieties projected to chosen coordinates given by variables the equation depends on. Our goal is to find the intersection of these varieties. If the point P on variety does not belong to the intersection of varieties, we can replace the original variety by a reduced one that does not contain P without influencing the final solution of the system. In some cases, we can determine the condition "does not belong to the intersection" in polynomial time, e.g., if it does not belong to the intersection of 2 varieties, it clearly cannot belong to intersection of all varieties. We call this process the (polynomial time) local reduction of the equation system [24]. If we can simplify each equation to a single solution, the system is trivially solved (it suffices to read values of variables from individual equations). In other cases we can compute the intersection by the (exponentially) difficult Gluing algorithm [14].

Let us suppose that we have a polynomial time local reduction algorithm A, but we are unable to remove any more partial solutions in the system. We suppose that the system is still unsolved, i.e., there are (unfixed) variables that can have both 0 and 1 assigned in each equation they are influencing. Let A be well behaved in a sense that it has a higher chance of reducing system, if individual equations have a lower number of partial solutions. If we take an unfixed variable and guess its value, we can remove at least one partial solution in each equation influenced by this variable. This can restart the local reduction. After guessing a finite number of variable values, we either end with the solved system, or we remove all partial solutions from some equation(s). The latter case means that the system does not have a solution compatible with the guesses (a conflict is detected), and we must backtrack our guessing sequence. The guessing process complexity is exponential in the number of variables that have to be guessed before the system can be reduced to a solution/conflict, and thus it dominates the polynomial complexity of A when we scale the problem. We silently suppose that behavior of A is not negatively influenced by expanding the system. Although the conditions on the behavior of A, and the computation of the influence of the system size on the required number of guesses can be impractical for exact determination of the complexities, it can be useful in practical experimental assessment of the strength of iterated block ciphers, and other cryptographic designs that can be scaled in a similar way.

The structure of the article is as follows: In Section 2 we provide preliminary definitions. In Section 3 we describe generic local reduction scheme, and its use in solving sparse Boolean equation system. In Section 4 we provide concrete algorithms that fit into the local reduction scheme. Finally, in Section 5 we focus on the use of the local reduction in experimental evaluation of the strength of the iterated ciphers against algebraic cryptanalysis. As an illustration, in Section 5.1 we provide experimental results from the analysis of the block cipher DES by the method of syllogisms.

2 PRELIMINARIES

Let $F : GF(2)^n \to GF(2)^m : F(x) = (f_1(x), \dots, f_m(x))$, be a Boolean function with m component functions f_i. Equation $F(x) = 0$ defines a system of m Boolean equations $f_i(x) = 0$, for $i = 1, \dots, m$ in n variables $x_1, x_2, \dots, x_n$.

Let S be the set of solutions of a system of Boolean equations $F(x) = 0$, i.e. $S = \{x \in GF(2)^n; F(x) = 0\}$. We say that the system is inconsistent, iff $S = \emptyset$. We usually suppose, that F is defined in such a way, that there is exactly one solution to the system, i.e. $|S| = 1$ (but this is not a necessary condition).

Let S_i be the set of solutions of i^{th} equation of the system, i.e. $S_i = \{x \in GF(2)^n; f_i(x) = 0\}$. If $x \in S$, then it must also be in each S_i (the converse is not true in general), thus $S \subset S_i$, and $S = \bigcap_{i=1}^{m} S_i$.

Definition 1. Let $f : GF(2)^n \to GF(2)$ be a Boolean function. Let $e^{(i)} \in GF(2)^n$,

$$\text{proj}_j(e^{(i)}) = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

We say that f depends on variable x_i iff there exists x such that $f(x) \oplus f(x \oplus e^{(i)}) = 1$.

Let X_i be the set of coordinates on which f_i depends. We say, that Boolean equation system $F(x) = 0$ is l-sparse, iff $|X_i| \leq l$, for each $i = 1, \dots, m$.

If f_i depends only on a small set of variables, we can represent S_i more effectively by storing only the values of variables on which f_i depends – vectors of length $|X_i|$, indexed by variables in X_i in the chosen order. We call these vectors partial solutions. We will denote a set of such vectors V_i. Other variables can take all possible values for each $v \in V_i$.[1] We will call (X_i, V_i) a symbol representation of the equation $f_i(x) = 0$, and the set $\mathcal{V} = \{(X_i, V_i); i = 1, \dots, m\}$ a symbol representation of the system.

Let $F(x) = 0$ define an l-sparse $m \times n$ Boolean equation system (as defined above). We can produce a symbol representation of $f_i(x) = 0$ in at most 2^l evaluations of the Boolean function f_i. Thus it is possible to compute $\mathcal{V}$ in at most $m2^l$ evaluations of simple Boolean functions. In some applications it is possible to compute $\mathcal{V}$ even faster. E.g. in algebraic cryptanalysis, we can describe individual S-boxes with equations $y = S(x)$ by symbols (X_i, V_i), where $X_i = \{x_1, \dots, x_k, y_1, \dots, y_j\}$, and V_i contains 2^k vectors in the form $(x, S(x))$.

The problem of solving Boolean equation system in symbol representation is as follows: Given $\mathcal{V}$, compute S. We can alternatively have two simpler goals: compute at least one element of S, or show that $S = \emptyset$.

3 SOLVING SPARSE BOOLEAN SYSTEMS BY LOCAL REDUCTION

Let $\mathcal{V} = \{(X_i, V_i)\}$ denote a system of Boolean equations in symbol representation, and let S denote a set of all solutions of the equation system. Let $s \in S$ be a solution, and let s_i be its projection to X_i. Clearly $s_i \in V_i$ for $i = 1, 2, \dots, m$. We call such vectors true (partial) solutions.

Let us suppose that there exists $v \in V_i$, which is not a projection of any of $s \in S$ – a false (partial) solution. A false solution is in a conflict with the rest of the equation system. It is possible to replace the symbol (X_i, V_i) by the symbol $(X_i, V_i \setminus \{v\})$. The new symbol represents a different Boolean equation $f'_i(x) = 0$ (we factor out the part of f_i corresponding to the root v), but the new system of equations has the same solution set S. We say that we have locally reduced the system. We may remove some partial solutions in such a way that the new equation does not depend on some $x \in X_i$. In this case, we should also remove x from X_i (and corresponding coordinates from V_i).

The primary goal of a local reduction method is to remove all false partial solutions from sets V_i. The reduced system allows us to easily identify a conflict, or a solution (of the original system). Let us suppose that $|S| > 0$, and $\mathcal{V}$ contains a symbol with $|V_i| = 1$. Then exact values of variables from X_i are known to us without the need to further examine the system. Similarly, if we get some $V_i = \emptyset$, then clearly $S = \emptyset$. We say that the equation is solved, if its symbol representation

[1] V_i is a projection of variety $V(f_i)$ into coordinates given by X_i.

does not contain any false partial solutions. The system is solved, if every equation in the system is solved.

Let $\mathcal{P}$ denote a finite set of predicates (in practice, we can use any suitable form of representation). Let us define three basic operations:

1. Apply:

Input: $\mathcal{P}$, (X_i, V_i)
Output: (X_i', V_i')
Description: Let $V_i' = \emptyset$. For each vector $v \in V_i$, check whether it is in conflict with $\mathcal{P}$. If not, add it to V_i'. Finally, construct symbol (X_i', V_i') by removing variables on which the equation no longer depends (if any).

2. Collect:

Input: (X_i, V_i)
Output: Set of predicates P
Description: Computes a set of predicates P that represent (local) information about the symbol (X_i, V_i).

3. Join:

Input: Sets of predicates P, $\mathcal{P}$
Output: Updated $\mathcal{P}$
Description: Merges (local) information P into (global) $\mathcal{P}$.

Generic local reduction scheme works as follows (with input $\mathcal{V}$, $\mathcal{P}$):

1. Let $i = 1$, $j = 0$.
2. Let $(X_i', V_i') = \mathit{Apply}(\mathcal{P}, (X_i, V_i))$. If $V_i' = \emptyset$, STOP, return CONFLICT.
3. Let $P = \mathit{Collect}(X_i', V_i')$.
4. $\mathcal{P}' = \mathit{Join}(P, \mathcal{P})$. If $\mathcal{P}$ is inconsistent, STOP, return CONFLICT.
5. If $(X_i, V_i) = (X_i', V_i')$ and $\mathcal{P}' = \mathcal{P}$, increment j. Else $j = 0$.
6. If $j = m$, STOP, return REDUCED.
7. $\mathcal{V} = (\mathcal{V} \setminus (X_i, V_i)) \cup (X_i', V_i')$. Cyclically increment i, and GOTO step 2.

This description is general, and not necessarily optimal. If no information about the system is known a priori, initial $\mathcal{P} = \emptyset$. Scheme returns the state (CONFLICT/REDUCED), and the final $\mathcal{V}, \mathcal{P}$, respectively.

It is easy to show that the algorithm stops. In each step either we learn something new about the system, remove a false partial solution, or increment the counter j. We suppose that possible $\mathcal{P}$ is finite. Then there exists a saturation point, i.e., we cannot learn anything more about the system. There is a limited number of partial solutions, thus we must stop removing solutions at some point.

Counter j reaches m, if we were unable to add new information, or remove any partial solution from each of the equations.

Let $m = O(n)$, and $\mathcal{V}$ be l sparse. Let $|\mathcal{P}|$, as well as running times of operations *Apply*, *Collect*, and *Join* be polynomially bounded in n. Then the running time of the *Local Reduction* algorithm is also polynomially bounded in n. This is easy to see: Each repetition of the algorithm uses 1 application of *Apply*, *Collect*, and *Join*, so we get $O(n^{k_1})$ complexity of each repetition, where k_1 is determined by the most difficult of the three operations. The number of repetitions is lower than $|\mathcal{P}| + m2^l + m = O(n^{k_2})$. The total running time is thus bounded by $O(n^{k_1+k_2})$. The similar holds for the memory requirements. In the remainder of this paper we will only consider polynomial *Local Reduction* algorithms.

If the algorithm stops with conflict, we know that the system is inconsistent, and thus it has no solution. Otherwise, the algorithm stops when nothing more can be learned about the system by the selected local reduction method. The algorithm then outputs the reduced system (and optionally also the information about the system $\mathcal{P}$). In some cases, the system is solved, and thus we can find at least one solution $s \in S$ in polynomial time (the trivially detected case is when each V_i contains a single solution). Otherwise we can find the solution using guessing and backtracking:

1. Let $G = \{p_1, p_2, \ldots, p_k\}$ be a set of (mutually exclusive) statements about the solution of the system $\mathcal{V}$ not already contained in $\mathcal{P}$. Furthermore, let G have a property, that if each of p_i is false, then $\mathcal{V}$ has no solution.
2. If $G = \emptyset$, return CONFLICT.
3. Guess: Choose $p \in G$.
4. Reduce: Let $(\mathit{State}, \mathcal{V}', \mathcal{P}') = \mathit{LocalReduction}(\mathcal{V}, \mathcal{P} \cup \{p\})$.
5. If *State* is CONFLICT, remove p from G, GOTO Step 2.
6. Unless solution is found, recursively apply this procedure on $(\mathcal{V}', \mathcal{P}')$. If CONFLICT is returned, remove p from G, GOTO Step 2.

If there was a polynomial instantiation of *Local Reduction* scheme which produces directly (without guessing) a solution for each $\mathcal{V}$, it would mean P = NP. However, we are skeptical that this is the case. We are more interested in determining the bounds for polynomially solvable cases, and the resulting global complexity of the guessing for systems, which are not polynomially solvable.

Due to the recursive nature of the guessing algorithm, its expected complexity is exponential (it depends on the depth and branching of the search tree). We call the sequence of sets G in the recursion the guessing strategy. The guessing strategy significantly influences the final complexity of the attack [21]. We believe that for each polynomial local reduction method (with fixed sparsity), there can be some optimal generic strategy, which leads to the lowest possible complexity. However, we cannot prove this hypothesis.

A typical set of statements G is a choice of a value of a single variable, e.g. $G = \{x_1 = 0; x_1 = 1\}$. If we use this method for algebraic cryptanalysis, the number of values we need to guess (depth of recursion) corresponds to a bit complexity of the attack.

Another type of G can be constructed from a single symbol as a choice of a partial solution, e.g. $G = \{v_1 \text{ is true solution}; v_2 \text{ is true solution}, \ldots\}$. This is especially useful, if we know a priori that there is exactly one solution of the system. If the symbol with $|X_i| = l$ variables contains $|V_i| = k_i$ partial solutions, we can examine all possible values of l variables with only k_i choices. The bit complexity corresponding to a single symbol is given by $\log_2 k_i$, and the bit complexity of the whole attack is given by $\sum \log_2 k_i$, with the sum taken over symbols we use in the attack.

If we want to assess the strength of the cipher against the attack based on local reduction, we can generate the problem instance with a known solution. In each step, when we have to guess some value, we can provide the correct answer to the algorithm, so no backtracking is necessary. We know that finally the process will converge to a (known) solution. Meanwhile, we keep track of the expected bit complexity of the (uninformed) attack. To estimate the complexity of the attack on a random unknown instance, we can randomize the process by trying different random instances (e.g., for the block cipher testing we can use different plaintexts, and keys, respectively), and we can also randomize the guessing strategy. Under the condition that the local reduction method is polynomial, we can expect the evaluation of the complexity can be relatively fast even for large systems.

4 LOCAL REDUCTION ALGORITHMS

We present a collection of polynomial local reduction algorithms that can be used in conjunction with guessing to solve equation systems in symbol forms. The list is not exhaustive, moreover the individual methods can be combined to create a more efficient version of the algorithm. The method "Spreading of constants" is the common part of each of the presented methods, otherwise these methods are distinct (i.e., there are examples of systems that can be solved by one of the methods but not the others). In our algebraic cryptanalysis experiments we use the method of syllogisms, for which we have implemented the experimental software solver [20].

4.1 Local Reduction with Linearization

One of the basic local reduction methods is based on the linearization of the symbols. We try to find linear equations that hold for each partial solution in each symbol individually, and remove incorrect solutions using global linear algebra.

Let $\mathcal{P}$ be represented as a set of linearly independent equations in all n variables. Maximum cardinality of $\mathcal{P}$ is n. We can instantiate the Local reduction procedures as follows:

1. **Apply:** For each partial solution $v \in V_i$ substitute values of variables from X_i within equations in $\mathcal{P}$, and check whether the resulting system of linear equations has at least one solution. If not, v can be removed.

2. Collect: Find the largest possible set P of independent linear equations in variables from X_i, such that each $v \in V_i$ is in a solution space of P.

3. Join: Add equations from P to $\mathcal{P}$, and remove all linearly dependent equations (e.g. by triangularization).

Theorem 1. For each symbol (X_i, V_i), with $0 < |V_i| \leq |X_i|$ there exists at least one affine Boolean function a, such that each $v \in V_i$ is a solution of $a(v) = 0$.

Proof. Let (X_i, V_i) be a symbol with $|X_i| = l$ variables, and $|V_i| = k$ partial solutions, respectively. Let A be $l \times k$ matrix with columns corresponding to the partial solutions. Let I be $l \times l$ identity matrix, with columns corresponding to variables from X_i. Let $M = (B|C)$ be a reduced row echelon form of matrix $(A|I)$. Each row with all zeros in the first k columns, represents a linear equation that holds for each solution (if the row is all zero, it is the trivial equation $0 = 0$, otherwise the coefficients are given by the part of the row in C). If $B = I$, we can construct affine equation by summing all rows of C (the right-hand side becomes one in each solution). $\square$

Sometimes it is possible to find linear equations also when $|V_i| > |X_i|$, but these cases are rare. However, if we guess value of some variable in the symbol, we expect that $|V_i|$ is halved (in average). Thus with each guess we are getting higher chance of finding linear equations, until the system can completely be linearized.

4.2 Spreading of Constants

The spreading of constants can be considered a special case of linearization, when we limit $\mathcal{P}$ to contain only equations in the form $x_i = a_i$. Finding the equations in the individual symbols is very easy. We remark that spreading of constants is also a special case of the method of syllogisms, and Agreeing, respectively.

4.3 Method of Syllogisms

The method of syllogisms was proposed by [24], and we have further been able to adapt it for algebraic cryptanalytic purposes [19, 21]. We provide a simple scheme as follows.

Let $\mathcal{P}$ contain logic formulas (implications) in the form $(x_j = a) \Rightarrow (x_k = b)$.[2] We have $|\mathcal{P}| = (2n)^2$.

1. Apply: For each partial solution $v \in V_i$ check whether all implications in $\mathcal{P}$ are true. If not, remove v.

2. Collect: For each pair of variables $x_j, x_k \in X_i$ make projection of V_i onto $\{x_j, x_k\}$. Each missing tuple (from the set $\{00, 01, 10, 11\}$) defines two new implications of the required form.

[2] In practice we store these formulas in the form of the implication graph.

3. Join: Add implications from P to $\mathcal{P}$, and compute the transitive closure of the corresponding implication graph.

The complexity of operations Collect and Apply is dominated by the number of partial solutions and variables in the symbol. If we consider system to be l-sparse with fixed l, this complexity factors becomes "constant" as n grows. The transitive closure of the implication graph can be computed in $O(n^3)$ (e.g. by Warshall's algorithm). Thus the whole Local reduction based on the method of syllogisms is polynomial as required.

Let $\mathcal{V}$ be an l-sparse equation system with randomly chosen X_i's. Let each l-tuple become a partial solutions in V_i with probability p. In [18] we show that if p is low enough, then the system can be solved by the method of syllogisms without any guessing with very high probability. The actual threshold depends on l (for smaller l's it is higher) but does not seem to depend on n. This means that if the average number of solutions per symbol in the system falls below some threshold, the system can be solved with no (more) guessing. If the system cannot be reduced by the method of syllogisms, with guessing we can remove some partial solutions (incompatible with the guess), thus lowering the average number of partial solutions per symbol. We can thus expect that the guessing process will terminate with the collapse of the system by the method of the syllogisms.

4.4 Agreeing

We can also map the method Agreeing from [14] into the scope of Local reduction methods. The set $\mathcal{P}$ is represented directly by $\mathcal{V}$ (can be augmented by more effective representations).

1. Apply: For each partial solution $v \in V_i$ check whether it is possible to find a matching projection on $X_i \cap X_j$ in V_j, with $X_i \cap X_j \neq \emptyset$. If the projection is missing in some of the connected equations, remove v.

2. Collect: Nothing to be done.

3. Join: Nothing to be done.

In [18] we show that Agreeing has similar behavior as the method of syllogisms when working with random equation systems, however it tends to be more effective when the sparsity l is larger (the threshold is at $l = 7$). We should note that there are now more efficient realizations of the Agreeing algorithm [17]. However, this new algorithm already combines basic Agreeing with guessing, and learning new information during the guessing process, so it is not possible anymore to directly incorporate it into our local reduction scheme.

5 EXPERIMENTAL EVALUATION OF THE SECURITY OF ITERATED CIPHER DESIGNS

The local reduction framework splits the question of the complexity of the algebraic cryptanalysis into two parts: polynomial local reduction algorithm, and exponential guessing. The practical realization of the local reduction algorithm can be quite complex, especially in comparison with the speed of the encryption routine. However, this complexity factor can be omitted when we are scaling the problem (exponential guessing part dominates the polynomial one). The strength of the cipher is then given by the minimum number of intermediate bits the attacker needs to guess before the local reduction method can solve the problem.

Most block ciphers are designed in such a way that it is relatively straightforward to construct an l-sparse Boolean equation system in symbol representation describing the encryption process. In the basic form, all input, output and key bits can also be represented by corresponding unknowns in the system. When analyzing the cipher we fix the input and output bits according to a known P-C pair.

The goal is to solve the equation system in the remaining unknowns with the minimum complexity. When we want to evaluate the complexity of the attack, we can work with instances for which we know the whole solution. We can thus provide the correct solution for each guess incrementally until the system collapses into a solved state. We estimate the complexity of the attack by summing expected bit complexity in each step of the guessing. We measure the final complexity in bits: If we are guessing directly variables, we count the number of guessed variables. If we are guessing whole vectors $v \in V_i$ (which fixes $|X_i|$ variables at once), we add $\log_2 |V_i|$ for each guess (instead of $|X_i| > \log_2 |V_i|$).

We remark, that if we "guess" input and key bits, and use the spreading of constants, the system will collapse into a single solution (spreading of constants emulates the working of the corresponding encryption logic circuit). Thus the upper limit for the minimal complexity of the local reduction methods is the key size (in the single P-C pair scenario).

Let us suppose that we use iterated cipher, and increase the number of rounds of the evaluated cipher. The size of the system increases with each round (due to new intermediate variables). The complexity of the local reduction part scales polynomially (so the experiments are not significantly slower). The bit complexity of the attack depends on the guessing strategy. If we guess values of randomly chosen variables (or partial solutions of randomly chosen symbols), we can expect that the bit complexity will grow with the size of the system. In this case, we simulate completely uninformed attacker, and the result can be considered the upper bound on the bit-complexity of the attack. If the attack with random guessing has lower bit-complexity than the number of key-bits, then the cipher is fundamentally insecure. It does not make sense to use (or even attack) such a cipher, as even the uninformed attacker can use the selected local reduction method to break it.

The guessing strategy is suitable for evaluation if it converges to some fixed upper value as we increase the number of rounds of the evaluated cipher. The bound for the secure cipher should be the key size n_K. In our experiments the strategy having this effect is the "maximum impact" strategy, where we guess the value of variable that occurs in the highest number of equations (or when we guess the partial solution of the equation, that has the highest total occurrence of variables in other equations).

Attack on a cipher with r rounds with expected lower average complexity than $n_K - 1$ can be considered as a shortcut attack. If the complexity of the attack is higher than $n_K - 1$ for each round above t, we can consider $r - t$ as a security margin (or $(r - t)/r$ as a relative security margin). The main problem is that the margin can depend on the chosen local reduction method, and the guessing strategy, respectively. It does not inform us whether the cipher is secure, only that it is at most as secure as given by the worst margin from all known attacks.

The evaluation framework can be used to either compare different attacks (defined by the local reduction method and the guessing strategy) using the bit complexity measure. Furthermore, it can provide us with the expected dependency of the complexity of algebraic attacks on the number of rounds of the selected cipher. Moreover, two ciphers can be compared by the means of their security margin (computed with the strongest known attack on each of the ciphers).

5.1 Experimental Results

As an illustration of the method, we provide experimental evaluation of the block cipher DES, and its security against the algebraic attacks based on the method of syllogisms. The system of equations was constructed from the blocks describing 2 rounds of DES. In each 2-rounds we use 64-bit input and 64-bit output bits of the round as variables as well as 48-bit input and 32-bit output bits of the S-boxes for each round. Thus each additional 2-rounds add 2(48+32)+64 new unknowns (inputs to next 2-rounds are outputs from the previous one). The nonlinear equations for the S-boxes are 10-sparse (6+4 bits, 2^6 partial solutions each), the rest of the system consists of linear equations for the XOR-s of individual bits (Feistel scheme, and key addition, respectively).

We use the following guessing strategies [23]:

1. RANDOM: Choose random symbol, guess partial solution.
2. MAXINFO: Local strategy, choose symbol with the best ratio k/l (minimal number of guesses to find the value of the maximum number of variables).
3. IMPACTs: Choose symbol with variables that influence the highest number of equations (in total).
4. IMPACTv: Choose variable that influences the highest number of equations.

The results are summarized in Figure 1. Random strategy can be used for up-to 6-round DES, for larger number of rounds the bit-complexity is higher than key-

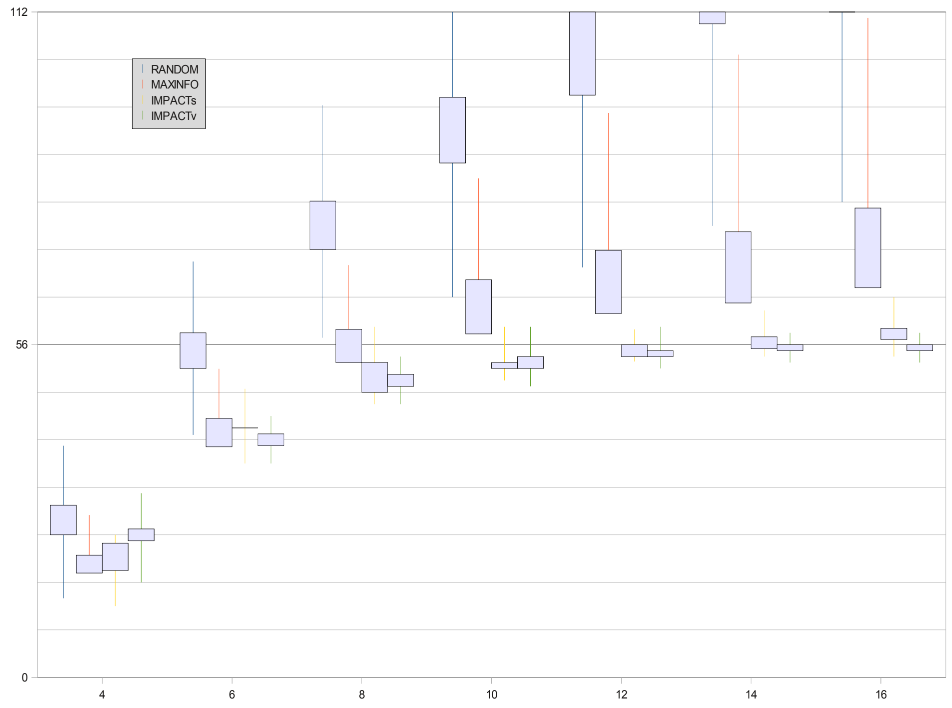

Figure 1. The bit-security of r-rounds DES against the attack with the method of syllogisms using different guessing strategies. Results are based on 1000 experiments, vertical lines connect minimum and maximum values, the thicker bars are bounded by the 1st, and 3rd quartile, respectively.

size. The MAXINFO strategy[3] that was proposed in [21] is better than uninformed guessing, but it does not converge to the key-size bound (but grows with the number of rounds). Both maximum impact strategies converge (statistically) to a key-size bound. In 8-round DES the attacker only needs to guess in average 50-bits of information (minimum was 46 bits). Although the brute-force attack might still be faster in practice, it depends on how fast can we implement the method of syllogisms in comparison with the DES encryption. Still the 8-round DES should not be considered secure. The expected complexity for the full DES is higher than 55 bits. We can see, that both "impact" strategies have lower minimum complexity (53 bits). It might be worth for the attacker to explore the experiments with lower than 55-bit complexity to construct a more effective attack on DES than brute-force, but it is usually cheaper to buy 4-times faster hardware.

[3] Originally this strategy was named GUESS, which is unfortunately not very descriptive.

We stress that all our guessing strategies are generic, i.e., they can be used for any sparse Boolean equation system. There might exist a strategy exploiting some specific properties of DES that can break more rounds, or provide better attack complexity for the given number of rounds. Moreover, some attacks on reduced round ciphers can be extended due to some specific properties of ciphers, as was demonstrated very recently by Courtois [6] in the case of the cipher GOST.

5.2 A Comparison with SAT Solvers

Although the approach provided in this paper is mostly experimental, it might be possible base for better understanding of the complexity of the algebraic crypto-analysis. We believe, that the results obtained by experiments with local reduction apply (in a qualitative way) to other methods used in algebraic cryptanalysis as well. To support this hypothesis, we have performed a series of experiments with SAT solvers. Namely, we compare our results on DES from Section 5.1 with estimated complexity of solving the (randomly) selected (round-reduced) DES instances with MiniSat 2.2.0 [7].

We have not run the whole experiments (which are quite costly in computational power). Instead we have estimated the required complexity in a way similar to [5]. Curtois and Bard provide a time (68 s) required to solve the instance of 6-round DES, when 20 bits of the key are fixed (and known). Given number of fixed bits g, and the solution time t_g respectively, we can estimate the time to solve the whole instance as $t_g 2^g$. We remark, that this estimate can be misleading. The SAT-solver in its basic setting will report the solution as soon as it is found, thus if we are "lucky", we get a solution "too fast", and the estimate is significantly skewed. The expected distribution of running times t_g is usually lognormal [4], so it is better to compute average estimate in logarithmic terms (instead of absolute times).

A different estimate can be obtained by fixing g *incorrect* bits of the key (randomly chosen), and measure the time to reject the value T_g. A local reduction method is more likely to reject the incorrect guess sooner (due to collisions) than confirm a correct one. on the other hand SAT solver is more likely to find the correct solution sooner than to reject the incorrect one. If we have to check all 2^g guesses of the fixed bits, the expected running time also depend on the architecture of the experiment. If the guesses are verified in series, we should base the estimate on the average time of the rejection. If guesses can be verified in parallel, it is possible to stop the verification of incorrect guesses as soon as the correct solution is found (so the minimal time applies).

To evaluate the complexity using SAT solvers, we run the local reduction experiment, and after each guess and reduction we store the corresponding (partially) reduced system. After the solution is found (or rejected), we convert the stored systems to CNF, and try to solve them with MiniSat. We start from the system, which was missing only one bit to be immediately solved by the method of syllogisms, then continue with the system missing 2 bits, etc. As the parameter g decreases, the running times t_g, T_g of the SAT solver increase. We stop the "unguessing" after the running times t_g, T_g are too long (MiniSat takes more than a day to complete).

There is a large variance in the number of decisions and the correspoding guessing times, see Table 1. Moreover, the estimated brute-force time (last column) is not monotone. This behaviour (along with more SAT solver based results that are out of scope of this paper) is explored in more details in [11].

g	Decisions $[10^3]$	t_g [s]	$t_g 2^g$ $[10^6$ Years]
55	70.0 ± 11.0	0.2 ± 0.1	173.49
54	94.0 ± 12.3	0.2 ± 0.1	121.91
53	154.5 ± 16.0	0.5 ± 0.1	138.24
52	199.3 ± 24.3	0.8 ± 0.3	117.63
51	243.7 ± 23.4	1.4 ± 0.5	101.36
50	294.6 ± 37.6	2.5 ± 1.0	87.69
49	415.5 ± 81.1	6.9 ± 4.0	123.34
48	577.8 ± 165.6	13.5 ± 8.3	120.55
47	800.2 ± 275.5	24.8 ± 14.5	110.67
46	$2\,511.4 \pm 1\,444.2$	113.9 ± 87.7	253.90
45	$6\,228.8 \pm 4\,960.0$	371.9 ± 369.4	414.64

Table 1. Number of decisions made by MiniSat 2.2.0, along with running times t_g (on Intel i7-3820, 3.60 GHz) reported when solving CNF's constructed from 14-round DES system reduced by the method of syllogisms after g-bits (suggested by IMPACTv strategy) were guessed. Results are averaged from 100 runs, reported along with the experimental standard deviation.

The results of our MiniSat experiments are summarized in Figure 2. As expected, the minimal times to verify the correct guess by MiniSat is significantly lower than the minimal time to reject incorrect guess. Moreover there is a higher difference between average and minimal estimates in these cases. Most notably, the minimal expected complexity to solve the system when using the correct guess is faster than brute force.[4] However, the average expected complexity, as well as the minimal and average complexity of rejecting incorrect guess is lower than brute-force complexity.

Figure 3 compares our estimates obtained by using the method of syllogisms (as implemented in the tool called sylog), and the results obtained by using MiniSat, respectively. Unlike Figure 1, we took into account also the (polynomial) running time required for a reduction (a dashed line shows the estimate without considering the growing cost of reduction). The running time of the sylog tool is inferior to highly optimized MiniSat. However, the dependence of the expected running times on the number of rounds for DES is very similar.

[4] Two different P-C pairs/keys gave different minimums: 2.5-times faster than brute force, and 11-times faster than brute-force, respectively. In both of these cases 16 bits of the key are correctly guessed, guessing 20 bits leads to a higher estimate.

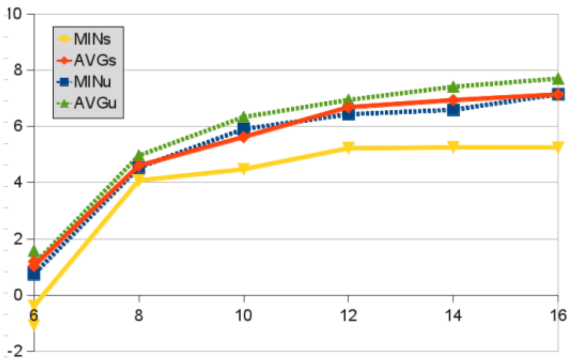

Figure 2. Expected minimal and average times required (on Intel E8200, 2.66 GHz) to solve round-reduced DES with MiniSat, in case of correct partial guess (solid lines), and incorrect partial guess (dotted lines). Results are in logarithmic scale with base 0 corresponding to the estimated brute-force effort on the same computer.

6 DISCUSSION AND CONCLUSIONS

In this paper we have examined a methodology that can be used for a fast evaluation of the complexity of generic algebraic attacks against ciphers. We use symbol representation of the cipher structure, and a polynomial time local reduction algo-

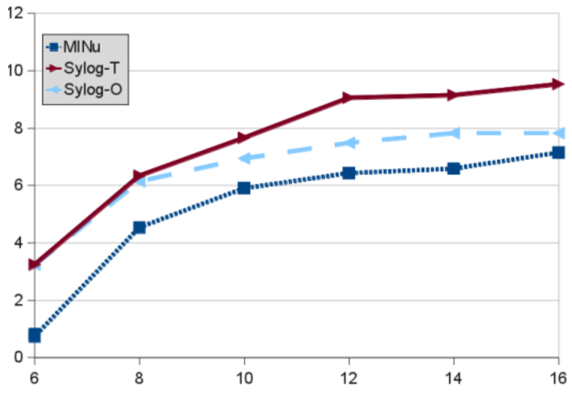

Figure 3. Expected minimal time required (on Intel E8200, 2.66 GHz) to reject a partial guess for round-reduced DES with MiniSat (dotted line) compared with the expected running time of the sylog tool (solid line). Dashed line corresponds to the estimated sylog complexity effort, if we ignore the polynomial factor in the complexity growth. Results are in logarithmic scale with base 0 corresponding to the estimated brute-force effort on the same computer.

rithm combined with informed or uninformed guessing to estimate the dimensions of the search tree. This allows us to predict the exponential part of the complexity of the whole algebraic attack, not only when using symbol based algorithms (such as gluing), but also when using SAT solver based attacks. As such, our method can be adapted to other applications where SAT solvers are used.

Although the expected complexity of attacks is exponential, there are specific instances where algebraic cryptanalysis can perform better then brute-force attacks on ciphers. This is intensified in cases of various experimental lightweight cipher proposals [8, 2, 3], which try to sacrifice some security margins to the speed or hardware resource consumption.

Symbol representation can be extended to MRHS form [13], which is especially suitable to model ciphers with low multiplicative complexity such as recently proposed LowMC [1]. We have proposed a new algorithm that can solve [22] the systems in MRHS form. The exponent in the complexity of this algorithm depends on the total number of right-hand sides (RHS) in the system. Some of the local reduction methods (such as agreeing) can be combined with MRHS representation to reduce the number of RHS in polynomial time, and thus reduce the complexity exponent for the whole system. It is an open question, whether any method that can be applied to symbol representation can be generalised also to MRHS equations. Furthermore, one can ask for each system what is the lowest possible number of RHS one can get with any local reduction method.

Finally, we would like to advise of some practical applications of local reduction methods. Algebraic attacks on ciphers can be mitigated by increasing the key space or other cipher parameters (such as number of rounds). On the other hand, there are various side channel attacks such as DPA [16], or fault attacks [12], that can break ciphers by measuring physical leakage from the cipher implementation. These side-channel attacks can be improved by combining them with algebraic attacks. Here symbol representation provides an advantage because we can capture the equation system as a collection of most probable hypotheses. When the attacker measures the physical leakage, he can try the attack based on local reduction, or just estimate its complexity with our methodology. If the attack complexity is too high, he can try further measurements, otherwise he can find the secret parameters by the algebraic attack.

Acknowledgement

This material is based upon work supported under the grant VEGA 1/0173/13, and VEGA 1/0159/17.

REFERENCES

[1] Albrecht, M. R.—Rechberger, C.—Schneider, T.—Tiessen, T.—Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (Eds.): Advances in Cryptology – EUROCRYPT 2015. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 9056, 2015, pp. 430–454.

[2] Antal, E.—Hromada, V.: A New Stream Cipher Based on Fialka M-125. Tatra Mountains Mathematical Publications, Vol. 57, 2013, No. 1, pp. 101–118, doi: 10.2478/tmmp-2013-0038.

[3] Antal, E.—Hromada, V.: A Micro-Controller Implementation of a Fialka M-125 Based Stream Cipher. Tatra Mountains Mathematical Publications, Vol. 60, 2014, No. 1, pp. 101–116, doi: 10.2478/tmmp-2014-0027.

[4] Bard, G.: Algebraic Cryptanalysis. Springer, 2009, doi: 10.1007/978-0-387-88757-9.

[5] Courtois, N.—Bard, G.: Algebraic Cryptanalysis of the Data Encryption Standard. In: Galbraith, S. (Ed.): Cryptography and Coding (Cryptography and Coding 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4887, 2007, pp. 152–169, doi: 10.1007/978-3-540-77272-9_10, doi: 10.1007/978-3-540-77272-9_10.

[6] Courtois, N. T.: Security Evaluation of GOST 28147-89 in View of International Standardisation. Cryptology ePrint Archive, Report 2011/211, 2011, `http://eprint.iacr.org/`.

[7] Een, N.—Sörensson, N.: The MiniSat Page. `http://minisat.se`.

[8] Eisenbarth, T.—Kumar, S.—Paar, C.—Poschmann, A.—Uhsadel, L.: A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test of Computers, Vol. 6, 2007, pp. 522–533, doi: 10.1109/MDT.2007.178.

[9] Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases (F4). Journal of Pure and Applied Algebra, Vol. 139, 1999, No. 1-3, pp. 61–88, doi: 10.1016/S0022-4049(99)00005-5.

[10] Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). Workshop on Applications of Commutative Algebra, Catania, Italy, April 3–6, 2002, ACM Press.

[11] Hromada, V.—Öllös, L.—Zajac, P.: Using SAT Solvers in Large Scale Distributed Algebraic Attacks Against Low Entropy Keys. Tatra Mountains Mathematical Publications, Vol. 64, 2015, No. 1, pp. 187–203, doi: 10.1515/tmmp-2015-0048.

[12] Hromada, V.—Varga, J.: Phase-Shift Fault Analysis of Trivium. Studia Scientiarum Mathematicarum Hungarica, Vol. 52, 2015, No. 2, pp. 205–220, doi: 10.1556/012.2015.52.2.1308.

[13] Raddum, H.: MRHS Equation Systems. In: Adams, C., Miri, A., Wiener, M. (Eds.): Selected Areas in Cryptography (SAC 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4876, 2007, pp. 232–245, doi: 10.1007/978-3-540-77360-3_15.

[14] Raddum, H.—Semaev, I.: New Technique for Solving Sparse Equation Systems. Cryptology ePrint Archive: Report 475/2006, `http://eprint.iacr.org/2006/475`, 2006.

[15] Raddum, H.—Semaev, I.: Solving Multiple Right Hand Sides Linear Equations. Designes, Codes and Cryptography, Vol. 49, 2008, No. 1-3, pp. 147–160, doi: 10.1007/s10623-008-9180-z.

[16] Repka, M.—Varchola, M.—Drutarovský, M.: Improving CPA Attack Against DSA and ECDSA. Journal of Electrical Engineering, Vol. 66, 2015, No. 3, pp. 159–163.

[17] Schilling, T. E.—Raddum, H.: Solving Equation Systems by Agreeing and Learning. In: Hasan, M. A., Helleseth, T. (Eds.): Arithmetic of Finite Fields (WAIFI 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6087, 2010, pp. 151–165, doi: 10.1007/978-3-642-13797-6_11.

[18] Schilling, T.—Zajac, P.: Phase Transition in a System of Random Sparse Boolean Equations. Tatra Mountains Mathematical Publications, Vol. 45, 2010, pp. 93–105, doi: 10.2478/v10127-010-0008-7.

[19] Zajac, P.: On the Use of the Method of Syllogisms in Algebraic Cryptanalysis. Proceedings of the 1st Plenary Conference of the NIL-I-004, 2009, University of Bergen, pp. 21–30.

[20] Zajac, P.: Implementation of the Method of Syllogisms. Preprint, 2010.

[21] Zajac, P.: Solving Trivium-Based Boolean Equations Using the Method of Syllogisms. Fundamenta Informaticae, Vol. 114, 2012, No. 3–4, pp. 359–373.

[22] Zajac, P.: A New Method to Solve MRHS Equation Systems and Its Connection to Group Factorization. Journal of Mathematical Cryptology, Vol. 7, 2013, No. 4, pp. 367–381.

[23] Zajac, P.—Čagala, R.: Local Reduction and the Algebraic Cryptanalysis of the Block Cipher GOST. Periodica Mathematica Hungarica, Vol. 65, 2012, No. 2, pp. 239–255.

[24] Zakrevskij, A.—Vasilkova, I.: Reducing Large Systems of Boolean Equations. 4th International Workshop on Boolean Problems, Freiberg University, 2000, pp. 21–22.

Pavol Zajac is Associate Professor at the Institute of Computer Science and Mathematics, FEI STU in Bratislava, where he also obtained his Ph.D. in applied mathematics in 2008. His main research is cryptography. He currently works on the design and cryptanalysis of lightweight ciphers which can be used to protect security and privacy on mobile devices. Furthermore, he works on practical algorithms for post-quantum cryptography, which are also resistant to side-channel attacks.