Des. Codes Cryptogr. (2017) 82:43–56
DOI 10.1007/s10623-016-0256-x

Upper bounds on the complexity of algebraic cryptanalysis of ciphers with a low multiplicative complexity

Pavol Zajac[1]

Received: 18 August 2015 / Revised: 11 July 2016 / Accepted: 13 July 2016 /
Published online: 26 July 2016

Abstract Lightweight cipher designs try to minimize the implementation complexity of the cipher while maintaining some specified security level. Using only a small number of AND gates lowers the implementation costs, and enables easier protections against side-channel attacks. In our paper we study the connection between the number of AND gates (multiplicative complexity) and the complexity of algebraic attacks. We model the encryption with multiple right-hand sides (MRHS) equations. The resulting equation system is transformed into a syndrome decoding problem. The complexity of the decoding problem depends on the number of AND gates, and on the relative number of known output bits with respect to the number of unknown key bits. This allows us to apply results from coding theory, and to explicitly connect the complexity of the algebraic cryptanalysis to the multiplicative complexity of the cipher. This means that we can provide asymptotic upper bounds on the complexity of algebraic attacks on selected families of ciphers based on the hardness of the decoding problem.

Keywords Algebraic cryptanalysis · Multiplicative complexity · Lightweight cryptography · Decoding problem

Mathematics Subject Classification 94A60 · 68P30 · 11T71

1 Introduction

Multiplicative complexity of a Boolean function is the minimum number of two-input AND gates required to implement the circuit that computes this function. It is possible to compute

This is one of several papers published in *Designs, Codes and Cryptography* comprising the "Special Issue on Coding and Cryptography".

✉ Pavol Zajac
pavol.zajac@stuba.sk

1 Slovak University of Technology in Bratislava, Ilkovičova 3, 812-19 Bratislava, Slovakia

the multiplicative complexity for small Boolean functions [21,25]. The multiplicative complexity of a typical cipher with large block or internal state is unknown, but can easily be upper bounded by summing up the multiplicative complexities of (typically small) non-linear building blocks of the cipher (such as S-boxes). We remark that multiplicative complexity of a random Boolean function should scale exponentially in the input size [6], but for a typical cipher it scales only polynomially due to implementation constraints. This makes low multiplicative complexity a distinguishing property of a practical cipher w.r.t. idealized random model.

In the whole article we will only consider circuits given in XOR and AND gates only. Let us suppose that an encryption is realized using a circuit with a limited number of two-input AND gates. We start with a set of secret bits, and some known or chosen initialization vector. Each input of the first AND gate can be computed as a linear combination of secret bits and initialization vector bits. The inputs of the second AND gate can be computed as a linear combination of secret bits, initialization bits and the output bit of the first AND gate, and so on. The result of the encryption is some final linear combination of the secret bits, initialization bits, and outputs of all AND-gates. We would like to compute the secret bits, given the output bits and initialization bits. We can model this problem as a problem of solving a system of non-linear equations over $GF(2)$. There are various algebraic methods that can be applied to this problem.

MRHS equation [17] can represent a non-linear equation over $GF(2)$ in a way, which is particularly useful for constructing equation systems describing ciphers using an S-box as the only means for non-linearity [13]. Similarly, we can adapt this representation to model our problem directly from a AND-XOR circuit description. In a recent article [23], we have introduced a new algorithm to solve MRHS equation systems. The algorithm converts MRHS equation system to a group factorization problem, which can be solved either by a global gluing algorithm, or by finding a specific codeword in a linear code. In this article, we explore the (theoretical) application of this algorithm to the algebraic cryptanalysis of ciphers with a low multiplicative complexity.

The evaluation of the complexity of the algebraic cryptanalysis can thus be explicitly summed up in the following three steps:

1. Convert a cryptanalytic problem to a problem of solving a MRHS equation system using the circuit description of the cipher (Sects. 2 and 3).
2. Use linear algebra to translate the problem of solving the MRHS equation system to a decoding problem (Sect. 4).
3. Use known upper bounds for decoding problem to obtain upper bounds on the complexity of the algebraic attack (Sect. 5).

2 MRHS equation systems

Definition 1 [18] Let $\mathbb{K}$ be a finite field. Multiple-Right-Hand-Sides (MRHS) equation is an expression in the form

$$x\mathbf{M} \in S, \tag{1}$$

where $\mathbf{M} \in \mathbb{K}^{(k\times n)}$ is an $(k \times n)$ matrix, and $S \subset \mathbb{K}^n$ is a set of n-bit vectors. We say that $x \in \mathbb{K}^k$ is a solution of the MRHS equation (1), if $x\mathbf{M} \in S$ holds for this particular x.

We are interested in MRHS equations where the set S is a set of a small constant size (typically given explicitly). In this case, it is easy to compute all solutions of the MRHS equation by repeatedly solving a linear system for each right-hand side vector taken from S. The problem becomes more difficult if we combine more MRHS equations into a MRHS system.

A MRHS system $\mathcal{M}$ is a set of m MRHS equations, with the same dimension k, i.e.

$$\mathcal{M} = \left\{x\mathbf{M}_i \in S_i;\, i = 1, 2, \ldots, m\right\},$$

with $\mathbf{M}_i \in \mathbb{K}^{(k\times n_i)}$, and $S_i \subset \mathbb{K}^{n_i}$, respectively. Vector $x \in \mathbb{K}^k$ is a solution of the MRHS system $\mathcal{M}$, if it is a solution of all MRHS equations in $\mathcal{M}$, i.e. $x\mathbf{M}_i \in S_i$ for each $i = 1, 2, \ldots, m$. We denote the set of all solutions of a MRHS system $\mathcal{M}$ by $Sol(\mathcal{M})$. Our goal is to obtain any solution of the system, or to show that no solution of the system exists.

Note that we can understand a MRHS system as a single MRHS equation $x\mathbf{M} \in S$, where

$$\mathbf{M} = \left(\mathbf{M}_1 \middle| \mathbf{M}_2 \middle| \cdots \middle| \mathbf{M}_m\right)$$

is a left-hand side matrix composed from the individual left-hand side matrices in the system, and $S = (S_1 \times S_2 \times \cdots \times S_m)$ is a Cartesian product of the right-hand side sets.

There are various methods to solve MRHS equation systems [13,14,19]. Most of these techniques involve joining individual equations into a larger one, so called Gluing [17]. Generally, joining two MRHS equations produce a larger one with a new right-hand side S that is a cartesian product of original right-hand sides. When joining two equations with linearly dependent left hand sides, some of the resultant right-had sides can be removed. Typically, during the whole Gluing algorithm the number of right-hand sides grows exponentially (with a possible time-memory trade-off), and after some point the system collapses into a single solution (if there is one). Some space can be saved by using compressed representation of right-hand sides through the use of binary decision diagrams [15,16].

In [23] we have proposed a new algorithm[1] to solve MRHS systems. This algorithm looks at solution space as an intersection between a linear code and an explicitly written set of points we get from a Cartesian product of right-hand sides (which is never evaluated). We then propose an algorithm of a different type. It uses a linear algebra and notions from group factoring [11] to solve the system. We also noted that the solution of the system can be obtained by finding a specific short vector in a solution set of a special linear system. In Sect. 4 we present the adapted algorithm to transform MRHS system into a syndrome decoding problem.

3 Using MRHS representation to model circuits of low multiplicative complexity

In this section we focus on the connection of the algebraic cryptanalysis and the multiplicative complexity of a circuit. This connection was already spotted by Courtois in [7], but only in a general sense. We quantify this connection exactly through the use of MRHS equations, and the transformation of the MRHS problem to a decoding problem. In this section we show how to efficiently transform algebraic cryptanalysis of a cipher with a low multiplicative complexity to a decoding problem instance, and estimate the complexity of attacking such a cipher. As mentioned in the introduction, we work with universal gate set (AND, XOR), which corresponds to basic operations $(\cdot, +)$ over $GF(2)$.

[1] Which in hindsight is similar to an informal proposal from [12, Sect. 5], but based on different premises.

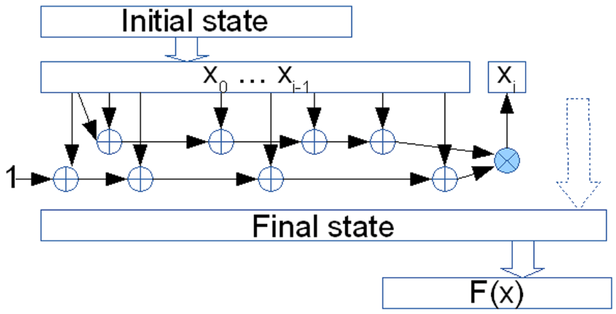

Fig. 1 Schematic view of the circuit model of the function F based on decomposition to AND gates $\otimes$ and XOR gates $\oplus$

Definition 2 Let $F : GF(2)^\nu \rightarrow GF(2)^\kappa$ be a vectorial Boolean function. Multiplicative complexity of F, denoted by $MC(F)$, is the minimum number of $GF(2)$ multiplications required to compute, using only operations from $GF(2)$, the value of the function F in an arbitrary point $x \in GF(2)^\nu$.

If function F has multiplicative complexity $MC(F) = \mu$, there exists a computational circuit composed of two-input AND gates and an arbitrary number of XOR gates that computes a value of F for any input $x \in GF(2)^\nu$. We can model the computation as a sequence of $\nu + \mu + \kappa$ bits (x_i), $x_i \in GF(2)$. The first ν bits, i.e., x_i for $i = 1, 2, \dots, \nu$ represent the input bits. The next μ bits are computed as the outputs of two-input AND-gates. Each input of the AND gate is an arbitrary affine function[2] of the previous bits. I.e., $x_{\nu+i} = (a_{i,0} + \bigoplus_{j=1}^{\nu+i-1} a_{i,j}x_j) \cdot (b_{i,0} + \bigoplus_{j=1}^{\nu+i-1} b_{i,j}x_j)$, where $a_{i,j}, b_{i,j} \in GF(2)$, $i = 1, 2, \dots, \mu$, and the $\bigoplus$ represents the sum over $GF(2)$. The two inputs of the same AND gate must be linearly independent, otherwise we would get an affine function, and the AND gate in question would not be required. Finally, the last κ bits are the outputs of F, which can be computed as any affine function of the previous bits, i.e., $x_{\nu+\mu+i} = c_{i,0} \oplus_{j=1}^{\nu+\mu} c_{i,j}x_j$. Each of the outputs of AND-gates must be used to compute at least one output bit, else the AND-gate would not be required. The output functions should be linearly independent, otherwise some of the outputs would be redundant (they could be computed as a linear combination of other outputs only). The model is schematically depicted in Fig. 1.

This model can be adapted to represent all the known stream ciphers, hash functions or block ciphers (we will call them just ciphers for the sake of simplicity), even if their multiplicative complexity (as a whole) is not known. This requires that the cipher under consideration can be written as a sequence of small operations with known multiplicative complexity (such as 4×4 bijective S-boxes analyzed in [25]), or other operations which can be realized by a limited number of two-input AND gates. We can work with a circuit description with possibly higher number of AND-gates than the minimum possible, but still low enough for the attack. It might be possible to reduce the circuit further by some optimization techniques [7].

Due to the requirement of the efficient implementation of the cipher the number of AND gates for a typical cipher would be very small in comparison to the expected multiplicative complexity of a random function. In the later text, μ will denote the number of two-input AND gates in the circuit representation of the cipher regardless of whether this number is the

[2] The bits of a possible initialization vector from the introduction part become affine constants in this model.

multiplicative complexity, or just an upper bound on the multiplicative complexity obtained from some existing implementation of the cipher.

The (traditional) main problem of the cryptanalysis is to compute the input bits $x_1, x_2, \ldots, x_\nu$, if we are given the output bits $x_{\nu+\mu+1}, x_{\nu+\mu+2}, \ldots, x_{\nu+\mu+\kappa}$ (inverting the one way function). In practice, some of the input bits might be known, e.g., the input block of a block cipher, the initialization vector of a stream cipher, etc. We can model the known inputs by adding additional outputs that are identically equal to the input values, or add the known bits to the affine constants a_0, b_0, and simplify the circuit where possible.

The circuit representation leads to direct translation of the cryptanalytic problem to a problem of solving a set of non-linear (degree 2) equations over $GF(2)$, which is the domain of the algebraic cryptanalysis. There are many techniques how to solve such a system, such as using some of the Gröbner basis techniques [9], translation to SAT problem [2], and others.

To translate the problem to MRHS representation we do the following:

1. For each of the μ AND-gates, write an MRHS equation using $(\nu+\mu) \times 3$ left-hand side matrix M_i. The first column of M_i contains coefficients $a_{i,j}$, $j = 1, \ldots \nu + i - 1$ (others are zero), the second column of M_i contains coefficients $b_{i,j}$, $j = 1, \ldots \nu + i - 1$, and the third column contains just a single non-zero coefficient at position $\nu + i$ (representing the output bit x_i).
2. The right-hand side of each M_i encodes the AND-gate operation, and the effect of affine constants. If affine constants $a_{i,0} = 0$ and $b_{i,0} = 0$, the right hand side contains four 3-bit vectors $S_i = \{(0,0,0), (0,1,0), (1,0,0), (1,1,1)\}$. The third bit in each vector is the output of the AND gate when the first two bits are its inputs, for all 4 possible input bit combinations. Affine constants are ”added” to the input bit prior to using the AND-gate. Thus, the right-hand side with affine constants taken into account is given as $S_i = \{(a_{i,0}, b_{i,0}, 0), (a_{i,0}, b_{i,0}+1, 0), (a_{i,0}+1, b_{i,0}, 0), (a_{i,0}+1, b_{i,0}+1, 1)\}$. We provide Example 1 to clarify the process.
3. The original equation system has $\nu + \mu$ unknowns. The m known output bits are right hand sides of κ (independent) linear equations (given by coefficients $c_{i,j}$) in these $\nu + \mu$ unknowns. By linear algebra we can express κ of the $\nu + \mu$ unknowns as affine functions of just $\nu + \mu - \kappa$ unknowns.
4. We can simplify a system of μ MRHS equations over $GF(2)^{(\nu+\mu)}$ to a smaller system over $GF(2)^{(\nu+\mu-\kappa)}$ by applying the affine substitutions from the previous step. We add the linear part to the left hand sides of MRHS equations (so we cancel out the substituted variables), and add the constant to the corresponding coordinate of all vectors in the right-hand side sets.

It is possible that some of the new MRHS equations after the last step contain linearly dependent columns. These equations can be simplified by Agreeing algorithm [13], so the final system is even simpler. However, in the following we will suppose the worst case system that cannot be further simplified after the substitution step.

Example 1 Consider an example with $x \in GF(2)^5$. We model an AND gate which computes x_4 from x_1, x_2, x_3:

$$x_4 = (x_1 \oplus x_2 \oplus 1) \otimes (x_2 \oplus x_3)$$

This equation can also be written in vector form as

$$(x \cdot (11000)^T \oplus 1) \otimes (x \cdot (01100)^T \oplus 0) = x \cdot (00010)^T$$

This leads to an MRHS equation in the form:

$$(x_1, x_2, x_3, x_4, x_5) \cdot \begin{pmatrix} 1\ 0\ 0 \\ 1\ 1\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \\ 0\ 0\ 0 \end{pmatrix} \in \begin{Bmatrix} 0\ 0\ 0, \\ 0\ 1\ 1, \\ 1\ 0\ 0, \\ 1\ 1\ 0, \end{Bmatrix}$$

On the right hand side, we have vectors representing all four combinations of inputs of the AND gate, and the corresponding output bit. Note that x_5 is unused in this example, and if the system does not have any more MRHS equations, we can omit this variable. We can also note that the second row of M is a sum of the first and the third one. This means that we can choose any x_2, and get four corresponding solutions of the MRHS equation (depending on the choice of x_2), giving a total of 8 solutions. In a larger MRHS system, this linear dependency might disappear by additional restrictions on x_2 from another MRHS equations in the system.

4 How to solve MRHS systems with syndrome decoding

Let $x\mathbb{M} \in S$ denote a MRHS system with

$$\mathbf{M} = \left(\mathbf{M}_1 \middle| \mathbf{M}_2 \middle| \cdots \middle| \mathbf{M}_m\right),$$

and $S = S_1 \times S_2 \times \cdots \times S_m$, respectively. Submatrices $\mathbf{M}_m$ have dimensions $(k \times n_i)$, while matrix $\mathbf{M}$ has dimension $k \times n$ with $n = \sum n_i$.

We can restrict ourselves to systems with full row rank k (no linearly dependent variables), and with $k < n$ (otherwise the system can be solved by linear algebra for any choice of the right-hand side $s \in S$). Thus, each solution x of the MRHS system produces a corresponding codeword $x\mathbf{M}$ in (n, k)-code over $\mathbb{K}$ generated by $\mathbf{M}$. Let $\mathbf{H}$ denote the $(n-k) \times n$ parity check matrix of this code. For any $c \in \mathbb{K}^n$, such that $c\mathbf{H}^T = 0$, there is a unique solution $x \in \mathbb{K}^k$ such that $x\mathbf{M} = c$. Moreover, vector x is a solution of the MRHS system if and only if $c \in S$.

The algorithm to solve MRHS equation system works by trying to obtain $c \in S$, such that $c\mathbf{H}^T = 0$. Given c, we can compute x by linear algebra. Vector c can be written as a concatenation of m parts $(c_1, c_2, \ldots, c_m)$, such that $c_i \in S_i$. Let

$$\mathbf{H} = \left(\mathbf{H}_1 \middle| \mathbf{H}_2 \middle| \cdots \middle| \mathbf{H}_m\right),$$

where each $\mathbf{H}_i$ is $(n-k) \times n_i$ matrix. The condition $c\mathbf{H}^T = 0$ can be rewritten as

$$(c_1, c_2, \ldots, c_m) \cdot \begin{pmatrix} \mathbf{H}_1^T \\ \mathbf{H}_2^T \\ \vdots \\ \mathbf{H}_m^T \end{pmatrix} = c_1\mathbf{H}_1^T + c_2\mathbf{H}_2^T + \cdots c_m\mathbf{H}_m^T = 0.$$

Let $S_i\mathbf{H}_i^T$ denote a set of vectors from $\mathbb{K}^{(n-k)}$ such that $S_i\mathbf{H}_i^T = \{c_i\mathbf{H}_i^T; c_i \in S_i\}$. We want to choose exactly one vector from each $S_i\mathbf{H}_i^T$ in such a way that all the selected vectors sum to zero, or to show that it is not possible to find such a set of vectors. From the corresponding c_i's we can construct the desired vector c by concatenation, and then obtain x by linear algebra. This corresponds to a group factorization problem (see [23]), or in a coding theory to a 1-regular decoding problem if the sizes of each S_i are fixed.

Let us reformulate the problem in a different form. Let $r = \sum |S_i|$, and let $\mathbf{R}$ denote a $r \times (n-k)$ matrix with rows composed of vectors from $S_i \mathbf{H}_i^T$. We want to find a solution u of $u\mathbf{R} = 0$, such that $w_H(u) = m$, and if we split u to parts $u = (u_1, u_2, \ldots, u_m)$ of sizes n_i, then each part has $w_H(u_i) = 1$. I.e., we are looking for a vector of a very specific type in the $(r, r-n+k)$-code with parity check matrix $\mathbf{R}^T$. In [23], we append each row of R by m bits that denote to which group $S_i \mathbf{H}_i^T$ the row belongs (the i-th of the additional bits is set to one, and the other $m-1$ bits are set to zero). We get a new system

$$u\mathbf{R}' = u \left(\begin{array}{c|c} & 1, 0, \ldots, 0 \\ & 1, 0, \ldots, 0 \\ & \vdots \\ \mathbf{R} & 1, 0, \ldots, 0 \\ & 0, 1, \ldots, 0 \\ & \vdots \\ & 0, 0, \ldots, 1 \end{array} \right) = (0, 0, \ldots, 0, 1, 1, \ldots, 1).$$

Each solution of this system with $w_H(u) = m$ is a solution of the original MRHS system. To find such u, we must solve a syndrome decoding problem for code with parity check matrix $(\mathbf{R}')^T$.

Let us now go in a different direction than in [23]. Let $r_{i,j}$ denote j-th vector in the i-th block of (row) vectors from $\mathbf{R}$. Recall that i-th block of $\mathbf{R}$ corresponds to $S_i \mathbf{H}_i^T$. Let $q = \sum r_{i,n_i}$, and let $\mathbf{Q}$ be a $(r-m) \times (n-k)$ matrix with rows in m blocks $q_{i,1} = r_{i,1} - r_{i,n_i}$, $\ldots$, $q_{i,n_i-1} = r_{i,n_i-1} - r_{i,n_i}$. Let $v\mathbf{Q} = q$. Suppose that $w_H(v_i) \leq 1$ for each part of v (v now consists of blocks with lengths $n_i - 1$). Then $u_i = (v_i, 1)$, if $w_H(v_i) = 0$, and $u_i = (v_i, 0)$, if $w_H(v_i) = 1$.

The reformulated problem can be seen as a classical syndrome decoding problem: Given syndrome q and parity check matrix $\mathbf{Q}^T$, find an error vector v of the weight at most m such that $v\mathbf{Q} = q$. We are working with a smaller binary $(r-m, r-m-n+k)$-code. There is a unique solution, if the code can repair up to m errors, i.e., it has a code distance at least $2m+1$. Otherwise, we must search for a specific solution with $w_H(v_i) \leq 1$ for each part of v (if the original MRHS system has a solution, such a vector must exist). The code under consideration is smaller than the one given by $\mathbf{R}'$, so we expect that the corresponding decoding problem can be easier to solve.

5 On the complexity of algebraic attacks on ciphers with a low multiplicative complexity

In Sect. 3 we have shown how to construct an MRHS equation system, whose solution is the solution of the problem of inverting a one way function $F : GF(2)^\nu \to GF(2)^\kappa$ with multiplicative complexity at most μ. The final MRHS system has $\nu + \mu - \kappa$ unknowns, with μ MRHS equations with $|S_i| = 4$ solutions each. The system matrix M has the size $(\nu + \mu - \kappa) \times (3\mu)$. I.e., $n = 3\mu$, $k = \mu + \nu - \kappa$, $m = \mu$, $r = 4\mu$ in the notation from Sect. 4.

Applying the algorithm in Sect. 4, we transform the problem of solving the MRHS system into a specific syndrome decoding problem $v\mathbf{Q} = q$. We are working with a binary $(r-m, r-m-n+k)$-code. Translating back the values, this means that code words have length $r-m = 3\mu$, and the code dimension is $r-m-n+k = 4\mu-\mu-3\mu+\mu+\nu-\kappa = \mu+\nu-\kappa$.

Code rate is thus

$$R = \frac{\mu + \nu - \kappa}{3\mu} = 1/3 + \frac{\nu - \kappa}{3\mu}.$$

Furthermore, we want to decode at most $m = \mu$ errors, so we would like the code with distance at least $2\mu + 1$. The Singleton bound $k + d \leq n + 1$ says that the code distance is less than $n + 1 - k = 2\mu + 1 + (\nu - \kappa)$. If $\nu > \kappa$, there are more unknown bits than the number of restrictions (known bits), so we expect that there are more solutions of the inversion problem.

If $\nu = \kappa$, we need an MDS code to provide a unique solution, i.e., every 2μ rows of Q must be linearly independent. This is not possible in the case of binary codes. However, we look for a specific error vector, which is unique, if the original solution of the system is unique. Other (incorrect solutions) must contain some linear combinations of more rows of Q in a single block, corresponding to picking a linear combination of right-hand sides, instead of a single right-hand side. We need a special decoding algorithm that does not accept such solutions. Furthermore, due to the construction of the matrix Q, the expected number of errors is lower than the upper bound μ. If there is an independent and equal chance to pick each right hand for each MRHS equation, we expect on average to only get $3/4\mu$ non-zero error bits.

If $\nu < \kappa$, we have more additional information bits than unknowns. Thus, we have a higher chance that the code can decode as much as μ (or the expected $3/4\mu$) errors. The code rate R is at most $1/3$, and goes to zero as $\kappa - \nu$ grows relative to the number of AND gates. Thus, the excess bits also lower the code rate, which means the complexity of the syndrome decoding approach is lower. If the number of excess known bits is higher than the number of AND gates, the problem degenerates to a simple linear algebra. On the other hand, if $\kappa - \nu = 0$, the number of AND gates does not influence the code rate, but it still increases the dimension of the problem, so the complexity of the algebraic attack grows exponentially in the multiplicative complexity.

The new results on general decoding algorithms [4] give some (asymptotic) upper bounds on decoding complexity for random linear codes. They estimate the worst case time complexity to be $2^{0.1019n} = 2^{0.3057\mu}$ with the space required bounded by $2^{0.0769n} = 2^{0.2307\mu}$. This means that to keep the attack complexity above the expected cost of the brute-force attack 2^{ν}, we need $\nu < 0.3057\mu$, or equivalently $\mu > 3.27\nu$, i.e., at least 3.27 times more AND gates than the number of unknowns/key-bits.

5.1 Upper bounds for complexity of selected cipher families

The analysis provided in this section shows that complexity of the algebraic attacks on ciphers with a low number of AND gates can be upper bounded by the complexity of solving a decoding problem. Let us consider some examples of well known lightweight cipher families. In the examples, the attack complexity derivation for block ciphers uses parameter N_b, the block size in bits. Parameter N_b directly influences the number of AND gates we need to implement the cipher. Furthermore, if the number of key bits $N_k = N_b$, we can use an algebraic attack with data complexity 1 (a single P-C pair should yield a unique key with high probability). It should be easy to adapt the examples to situations where $N_k \neq N_b$.

PRESENT [5] is an example of a cipher based on a simple substitution-permutation network (SPN). In each round, after key addition, bijective 4-bit S-boxes are applied to the state of the cipher, and finally the state bits are permuted. PRESENT has a specific S-box with known multiplicative complexity 4 out of maximum 5 [25]. One S-box is also applied in each round in the key schedule. Let N_r denote the number of rounds of PRESENT-like SPN cipher, and N_b the block size in bits. The number of AND gates used during the encryption can be bounded by $N_r \cdot (N_b/4+1) \cdot M$, where M is the multiplicative complexity of the S-box (the +1 comes from the key-schedule, but it could be ignored in an asymptotic analysis). If each S-box has multiplicative complexity 4, we need at least four rounds to get the estimated attack complexity higher than 2^{N_b} (for a large enough block size). When using S-boxes with multiplicative complexity 5, the estimated complexity of the attack (for large enough N_b) becomes

$$O(2^{0.3057\mu}) = O(2^{0.3057 N_r \cdot (N_b/4+1)\cdot 5}) \approx O(2^{0.3844 N_r \cdot N_b}).$$

In this case, we get a higher complexity estimate than 2^{N_b} for three rounds already.

National Security Agency proposed two lightweight ciphers SIMON and SPECK [3], both of which have simple rounds with a low multiplicative complexity. SIMON is a hardware oriented cipher based on Feistel structure. SIMON applies single AND gate per each bit of the half state in each (Feistel) round. This means that we can express its multiplicative complexity as $N_r \cdot (N_b/2)$. This means that we need at least seven rounds of SIMON to reach the desired security level of at least 2^{N_b}. SPECK is based on ARX[3] design. It is known that modular addition of n-bit words has multiplicative complexity $n-1$ [7]. This is the only non-linear operation in SPECK, and it operates on half-words. Thus, if we ignore the key schedule, we get a similar result as for SIMON: at least seven rounds required to protect against algebraic attacks. On the other hand, key schedule uses a single modular addition after each round. If we have to take the key schedule into account, we can only break four rounds of SPECK with a generic algebraic attack.

A prominent example of a stream cipher with a low multiplicative complexity is Trivium [8]. In each clock of Trivium, only 3 AND gates in each clock are applied to the cipher state. Suppose the attacker wants to reconstruct the internal state of the generator from the known keystream. Here, $\nu = \kappa = 288$ (internal state size), and the multiplicative complexity μ is at most 3ν. This leads to an estimated complexity bounds: 2^{264} time and 2^{199} memory. This still does not represent a break of Trivium, as Trivium only provides an 80-bit security level.

Recently, a new design LowMC was introduced [1]. It is based on SPN with partial S-box layer, and thus uses a very low number of AND gates per output bit (6-9 ANDs per bit). Although this number of bits is higher than our asymptotic bound 3.27 AND gates per bit, it leaves a very low security margin. Furthermore, our estimate is based on the upper bounds for the generic decoding problem based on randomly generated linear codes. In practice, codes obtained by our algorithm from real-world ciphers are far from random linear codes. The generating matrix of the code can be very sparse and contain some regular structure that might be exploited by more sophisticated attacks.

Although we have prepared this article with an intention to provide theoretical results, on a suggestion of an anonymous reviewer we have also prepared a proof-of-concept attack on a simple 4-round SIMON. A simple decoding-based attack is approximately 2000-times faster in average than an exhaustive search. The details of the attack are provided in Appendix.

6 Concluding remarks

The MRHS equations and simple linear algebra can be used to transform an instance of algebraic cryptanalysis to an instance of the decoding problem, either a 1-regular decoding

[3] Cipher design where only operations modular Addition, bit Rotation, and XOR are used.

problem, or a specific syndrome decoding problem. We can then apply new algorithms from the decoding area to decrypt standard block and stream ciphers, or to invert hash functions. This is especially useful for ciphers with (very) low multiplicative complexity. Moreover, the new results in syndrome decoding (and generalized birthday attacks) can provide us strong bounds on the required multiplicative complexity of ciphers.

In this article we use the transformation of the algebraic cryptanalysis to the decoding problem to derive upper bounds on the complexity of algebraic attacks relative to the multiplicative complexity of the cipher. As the analysis in Sect. 5.1 shows, the generic decoding attacks can break only very small instances of ciphers. See also Appendix for an example attack on a small version of SIMON.

As far as we know, there is no known cipher that can be broken by this generic technique. However, we believe that in practice the attack techniques can be further improved. The first type of improvements can be derived from the special structure of the decoding problem derived from the algebraic cryptanalysis. In our analysis we use generic decoding bounds derived for random linear codes, but the codes we get from real ciphers have specific structure (repeated rounds, symmetries in design) that might be exploited in decoding algorithm. Other types of improvements might be derived from multiple instances of the decoding problem.

Sendrier [20] analyzes the situation when attacker needs to decode only one out of many syndromes in the same code. The algorithm gives a significant advantage to the attacker when the number of errors is small. When applying the algebraic cryptanalysis to obtain a symmetric encryption key we have many possible instances with the same key as unknown, and we need to decode just one of them. However, the instances have different (but related) parity-check matrices. We might ask, whether it is possible to adapt some decoding methods to decode one out of many instances in this specific form, and profit from many possible plaintext-ciphertext pairs in the attack.

Furthermore, the construction of the matrix Q removes last vectors from the groups of vectors in R, sums them up and produces the instance to decode. With a single instance of the algebraic cryptanalysis, we might construct many instances of the decoding problem by a random selection of vectors that are taken out of the matrix R. Again, we might ask, whether this can speed-up or simplify the decoding algorithm.

Acknowledgements This research was supported by Grant VEGA 1/0173/13 and by project Cryptography brings security and freedom SK06-IV-01-001 funded by EEA Scholarship Programme Slovakia.

Appendix: Experimental attack on a small version of Simon

Attack description and expected complexity

To support the theoretical results, we have prepared an example of a decoding attack on a 4-round version of a block cipher SIMON-32. SIMON is a cipher with Feistel structure, which uses only bit rotations and bitwise AND and XOR in its round function. For the purposes of the attack, we have restricted the key to 32 bits, and adapted the key schedule to reflect this change (we use $m = 2$ instead of original $m = 4$). For the full description of the cipher, please refer to the original paper [3].

Given a plaintext-ciphertext pair $(x_L|x_R, y_L|y_R)$ we can compute the key $k = (k_0, k_1)$ from a set of equations

$$\begin{aligned} F(k_0 + F(x_L) + y_R) &= y_L + F(y_L) + k_1 \\ F(y_L + k_3 + F(y_R)) &= y_R + k_2 + k_0 \end{aligned} \tag{2}$$

Bits of the subkeys k_2 and k_3 can be expressed as affine functions of the bits of k. Function $F(x)$ can be computed as $F(x) = (ROT(x, 8)$ AND $ROT(x, 1))$ XOR $ROT(x, 2)$, where AND, XOR, ROT are bitwise vector operations on 16 bit vectors (in our example).

We can rewrite the set of equations (2) directly on a bit level as a set of 32 non-linear equations over $GF(2)$ of a form:

$$\left(k \cdot a_{i,0}^T + c_{i,0}\right) \cdot \left(k \cdot a_{i,1}^T + c_{i,1}\right) = k \cdot a_{i,2}^T + c_{i,2}. \tag{3}$$

Vectors $a_{i,j} \in GF(2)^3 2$ and constants $c_{i,j} \in GF(2)$ are determined from (2) and the key schedule.

Finally, we transform the equation to a MRHS system $k \cdot \mathbf{M} \in S$:

$$k\left(a_{0,0}^T a_{0,1}^T a_{0,2}^T | \cdots\right) \in \left\{\begin{array}{c} (0,0,0) + (c_{0,0}, c_{0,1}, c_{0,2}), \\ (0,1,0) + (c_{0,0}, c_{0,1}, c_{0,2}), \\ (1,0,0) + (c_{0,0}, c_{0,1}, c_{0,2}), \\ (1,1,1) + (c_{0,0}, c_{0,1}, c_{0,2}) \end{array}\right\} \times \cdots \tag{4}$$

Thus, in our example matrix $\mathbf{M}$ has dimensions 32×96, and $|S| = 4^{32}$. We expect that there is a unique vector $c \in S$, such that $k \cdot \mathbf{M} = c$ has a solution. This vector is composed of parts that correspond to inputs and outputs of the AND-gates used in the inner two Feistel rounds during the encryption.

In a naive way, we can find this vector either by trying all 2^{64} vectors in S, or more efficiently by trying all 2^{32} options for k (which is equivalent to a brute force attack). We will instead employ the decoding approach. By using the algorithm presented in Sect. 4, we change the problem to a syndrome decoding form: find $v \in GF(2)^{96}$, such that $v\mathbf{Q} = q$, $w_H(v) \leq 32$, and $w_H(v_{3i..3i+2}) \leq 1$ for $i = 0, 1, \ldots 31$. Recall that $\mathbf{Q}$ is a block matrix which should have 32 rows and 32 blocks of 3 columns each.

To find v we employ a modified version of a Lee-Brickel decoding algorithm [10]. In the original algorithm, it is sufficient to find a vector of a prescribed weight. In our modification, we must also ensure the weight in each block. The modified algorithm can be summarized as follows :

1. Apply a random permutation to 3-bit blocks of columns in $\mathbf{Q}$ to obtain $\mathbf{Q}'$.
2. Construct 96×64 matrix $\mathbf{H} = (\mathbf{H}_1|\mathbf{H}_2|\mathbf{H}_3)$ by randomly assigning columns from each block of $\mathbf{Q}'$ into distinct parts $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3$.
3. Use linear algebra on $(\mathbf{H} \mid q)$ to obtain

   $$\left(\mathbf{I}_{64}\ \mathbf{U} \middle| \ \hat{q}\right).$$

   If this is not possible, re-randomize matrix $\mathbf{H}$.
4. For each pair of vectors $u_0, u_1 \in cols(\mathbf{U})$ compute $u = u_0 + u_1 + \hat{q}$. For each u:
   (a) Test whether $w_H(u) \leq \omega$, where $\omega = 32 - 2$. If not, continue with different u.
   (b) Non-zero bits in u can now cancelled by adding at most ω columns u_i from $\mathbf{I}_{64}$ part of $\mathbf{H}$.
   (c) Check that no two vectors u_i are from the same original block of $\mathbf{Q}$. If not, continue with different u.

(d) Construct $v' \in GF(2)^{96}$ that has 1's exactly in positions corresponding to u_i's. Reconstruct v by applying the inverses of the column distribution and block permutation to v'.

5. If no suitable u was found, re-randomize matrix $\mathbf{H}$.

The result v from this algorithm can be further used to reconstruct c: if $v_{3i..3i+2} = (0, 0, 0)$, the first vector in S_i is used, is $v_{3i..3i+2} = (1, 0, 0)$, the second vector in S_i is used, etc. Finally, by using a simple linear algebra, we decode c to a target key k.

Note that there is no guarantee that the algorithm stops. If the system does not have a solution, we will try to re-randomize $\mathbf{H}$ forever. In practice, we stop the algorithm after some fixed amount of retries. We can estimate the expected number of retries by computing the probability of success in each try. Originally, we have 32 non-zero "error" positions. We expect that 1/4 of these is summed into the syndrome q. The remaining 24 "errors" are distributed between $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3$ uniformly. Our algorithm can only succeed if at most 2 "errors" fall into the $\mathbf{H}_3$ part, that is with probability 0.53%. This probability is further modified by the probability that $(\mathbf{H}_1, \mathbf{H}_2)$ is invertible. This leads to an expected 650 number of retries before a solution is found.

The algorithm can be generalized to different matrix sizes. E.g., for SIMON-32 with 5 rounds we would get 144×96 matrix $\mathbf{Q}$, with a condition $w_H(v) \leq 48$. If we also allow re-randomization of $\mathbf{Q}$ and q (i.e., not just sum up the first rows of $\mathbf{R}$), we can expect approximately 37 times more retries than in the 4-round case.

In general, for SIMON-32 with $r + 2$ rounds, the matrix size is $48r \times 32r$, and target weight is $16r$. E.g., for 22 rounds we get parity check matrix $\mathbf{Q}$ with dimensions 1056×704, $w = 352$. The asymptotic upper bound estimate leads to an expected work factor of 2^{108}. However, this bound was computed for a full decoding, see [4]. If we could somehow apply their techniques from half distance decoding and lower the complexity exponent to $0.04934n$, this would lead to a work factor 2^{52}. There is a huge gap between these estimates, and we see a lot of potential for further practical research in this area.

Experimental results

We have implemented the algorithm in SAGE [22]. The source code is available at https://github.com/zajacpa/DecodingAttack [24]. We have tested the attack software from a command-line interface of SAGE on a PC with `Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz` with 8 cores, 64 GB RAM and Lubuntu OS.

As a target instance we have selected $(x_L|x_R) = (0, 0)$, and $(y_L|y_R) = (1, 1)$, because the obvious first choice $y = (0, 0)$ did not lead to a solution. The running time for the attack repeated 100 times was 9 min 32 s, that is an average attack time took 5.73 s. For a limited comparison, we have also executed a modified `'simon.py'` script [26] in the same SAGE instance. The modified script runs 10,000 encryptions and 10,000 decryptions of SIMON-32 in 0.840 s. From this we can extrapolate the expected brute-force attack time to be 11,247 s, which is 1968-times more than the attack via decoding takes on average.

In our example the matrix $\mathbf{Q}$ had dimensions 96×65, because the row rank of $\mathbf{M}$ was only 31 instead of 32. We have slightly modified the algorithm to consider only vectors u with last bit equal to zero. Thus we can still cancel the remaining ones with the first 64-bit reduced columns.

During the calculation we have counted the number of required re-randomisations of $\mathbf{H}$. The average number was 389 ± 383. The histogram of the distribution is depicted in Fig. 2. The number of retries is lower than expected, mostly due to the fact that matrix $(\mathbf{H}_1\mathbf{H}_2)$

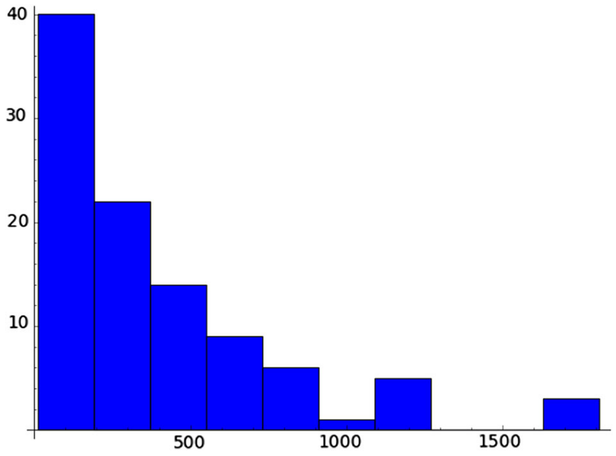

Fig. 2 Histogram of the number of re-randomisations required to decode $v\mathbf{Q} = q$ (from 100 experiments with the same P-C pair)

created from random columns of **Q** has a higher chance to be invertible than a random binary matrix.

Finally, in a second experiment we have tested the attack with fixed $(x_L|x_R) = (0, 0)$ and 100 randomly generated pairs $(y_L|y_R)$. The stop condition was set to 1500 iterations (based on a previous experiment). The experiment took 23 min 35 s, or 14.15 s on average. The number of successfully decoded keys was 35. The average number of iterations for successful attacks was 284 ± 345.

References

1. Albrecht M.R., Rechberger C., Schneider T., Tiessen T., Zohner M.: Ciphers for MPC and FHE. In: Advances in Cryptology—EUROCRYPT 2015, pp. 430–454. Springer, Berlin (2015).
2. Bard G.V., Courtois N.T., Jefferson C.: Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. Cryptology ePrint Archive, Report 2007/024 (2007). http://eprint.iacr.org/.
3. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptology ePrint Archive (2013).
4. Becker A., Joux A., May A., Meurer A.: Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In: Advances in Cryptology—EUROCRYPT 2012, pp. 520–536. Springer, Berlin (2012).
5. Bogdanov A., Knudsen L.R., Leander G., Paar C., Poschmann A., Robshaw M.J., Seurin Y., Vikkelsoe C.: PRESENT: An Ultra-lightweight Block Cipher. Springer, Berlin (2007).
6. Boyar J., Peralta R., Pochuev D.: On the multiplicative complexity of boolean functions over the basis $(\wedge, \oplus, 1)$. Theor. Comput. Sci. **235**(1), 43–57 (2000).
7. Courtois N., Hulme D., Mourouzis T.: Solving circuit optimisation problems in cryptography and cryptanalysis. Cryptology ePrint Archive, Report 2011/475 (2011).
8. De Cannière C.: Trivium: A stream cipher construction inspired by block cipher design principles. In: Information Security, pp. 171–186. Springer, Berlin (2006).
9. Faugère J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Workshop on Applications of Commutative Algebra, Catania, Italy, 3–6 Apr 2002. ACM Press, New York (2002).
10. Lee P.J., Brickell E.F.: An observation on the security of McEliece's public-key cryptosystem. In: Workshop on the Theory and Application of of Cryptographic Techniques, pp. 275–280. Springer, Berlin (1988).

11. Magliveras S.S., Stinson D.R., van Trung T.: New approaches to designing public key cryptosystems using one-way functions and trapdoors in finite groups. J. Cryptol. **15**(4), 285–297 (2002).
12. Raddum H.: MRHS equation systems. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, pp. 232–245. Springer, Berlin (2007).
13. Raddum H., Semaev I.: Solving multiple right hand sides linear equations. Des. Codes Cryptogr. **49**(1–3), 147–160 (2008).
14. Schilling T., Raddum H.: Solving equation systems by agreeing and learning. In: Hasan M., Helleseth T. (eds.) Arithmetic of Finite Fields. Lecture Notes in Computer Science, vol. 6087, pp. 151–165. Springer, Berlin (2010). doi:10.1007/978-3-642-13797-6_11.
15. Schilling T.E., Raddum H.: Analysis of Trivium using compressed right hand side equations. In: Information Security and Cryptology-ICISC 2011, pp. 18–32. Springer, Berlin (2012).
16. Schilling T.E., Raddum H.: Solving compressed right hand side equation systems with linear absorption. In: Sequences and Their Applications—SETA 2012, pp. 291–302. Springer, Berlin (2012).
17. Semaev I.: On solving sparse algebraic equations over finite fields. Department of Informatics, University of Bergen, Tech. Rep. (2005).
18. Semaev I.: On solving sparse algebraic equations over finite fields. Des. Codes Cryptogr. **49**(1–3), 47–60 (2008).
19. Semaev I., Mikuš M.: Methods to solve algebraic equations in cryptanalysis. Tatra Mountains Math. Publ. **45**, 107–136 (2010).
20. Sendrier N.: Decoding one out of many. In: Post-Quantum Cryptography, pp. 51–67. Springer, Berlin (2011).
21. Sönmez M.T., Peralta R.: The multiplicative complexity of boolean functions on four and five variables. In: Lightweight Cryptography for Security and Privacy, pp. 21–33. Springer, Berlin (2015).
22. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 7.2) (2016). http://www.sagemath.org.
23. Zajac P.: A new method to solve MRHS equation systems and its connection to group factorization. J. Math. Cryptol. **7**(4), 367–381 (2013).
24. Zajac P.: GitHub - zajacpa/DecodingAttack: demonstration of the decoding attack on SIMON-32. https://github.com/zajacpa/DecodingAttack (2016).
25. Zajac P., Jókay, M.: Multiplicative complexity of bijective 4×4 S-boxes. Cryptogr. Commun. **6**(3), 255–277 (2014).
26. Zhu B.: GitHub - bozhu/NSA-ciphers: SIMON and SPECK, the two lightweight block ciphers dedesign by the researchers from NSA. https://github.com/bozhu/NSA-ciphers (2016).