Improved Agreeing-Gluing Algorithm

Igor Semaev

Department of Informatics, University of Bergen, Norway
igor@ii.uib.no

Abstract. In this paper we study the asymptotical complexity of solving a system of sparse algebraic equations over finite fields. An equation is called sparse if it depends on a bounded number of variables. Finding efficiently solutions to the system of such equations is an underlying hard problem in the cryptanalysis of modern ciphers. New deterministic Improved Agreeing-Gluing Algorithm is introduced. The expected running time of the Algorithm on uniformly random instances of the problem is rigorously estimated. The estimate is at present the best theoretical bound on the complexity of solving average instances of the problem. In particular, this is a significant improvement over those in our earlier papers [20, 21]. In sparse Boolean equations a gap between the present worst case and the average time complexity of the problem has significantly increased. Also we formulate *Average Time Complexity Conjecture*. If proved that will have far-reaching consequences in the field of cryptanalysis and in computing in general.

1 Introduction

1.1 The problem and motivation

Let (q, l, n, m) be a quadruple of natural numbers, where q is a prime power. Then F_q denotes a finite field with q elements and $X = \{x_1, x_2, \ldots, x_n\}$ is a set of variables from F_q. By X_i, $1 \le i \le m$ we denote subsets of X of size $l_i \le l$. The system of equations

$$f_1(X_1) = 0, \ldots, f_m(X_m) = 0 \tag{1}$$

is considered, where f_i are polynomials over F_q and they only depend on variables X_i. Such equations are called l-sparse. A solution to (1) over F_q is an assignment in F_q to variables X that satisfies all equations (1). That is a vector of length n over F_q provided the variables X are ordered. The main goal is to find all solutions over F_q. There may be solutions over the extensions of F_q, but we do not need them. In the most interesting case $m = n$ we will have just one solution over F_q on the average.

Deterministic Improved Agreeing-Gluing (IAG) Algorithm which solves the system is here suggested. It is presented by two variations. The expected complexity of one variation is rigorously estimated assuming uniform distribution on the problem instances; see Section 1.3. The results provide a significant improvement over earlier average time complexity estimates [20, 21]. Another variation seems require more sophisticated analysis than that presented here.

Anyway, the run-time analysis is more transparent in comparison with that in [20, 21]. It is based on the theory of random allocations [13] as earlier. However, it now requires estimating the probabilities of only few common events; see Section 9. The algorithm running time expectation is the sum of $O(n^3)$ terms, each of them is bounded by the same real-valued function in three real variables, whose values may differ for different terms. The maximum of the function is computed

with the advanced optimization package MAPLE [15] in Sections 10 and 11. The computation does not depend on n.

The approach, which exploits the sparsity of equations and doesn't depend on their algebraic degree, was studied in [28, 17, 20, 21]. These are guess-and-determine algorithms. In sparse equations the number of guesses on a big enough variable subset $Y \subseteq X$ and the time to produce them is much lower than $q^{|Y|}$ due to the Search Algorithm; see Section 8, where the algorithm is described. This is a more general and efficient method than that in [21]. No preference was previuosly made on which variables to guess. We now argue that guessing values of some particular variables leads to better asymptotic complexity bounds.

The article was motivated by applications in cryptanalysis. Modern ciphers are product ciphers, the mappings they implement are compositions of not so many functions in a low number of variables. The similar is true for underlying one-way functions in asymmetric ciphers. Any one-way function is representable by a low number of small gates as its values should be efficiently computed. Intermediate variables are introduced to simplify equations, describing the cipher, and to get a system of sparse equations like (1). More general type of sparse equations called Multiple Right Hand Side linear equations and introduced in [18], is even more convenient tool to write equations from modern ciphers like the AES. An efficient solution to the equations may break the cipher.

Equations common in cryptanalysis depend on large variable sets. Those represented by common dense polynomials are hardly manageable as it is not possible to keep them in computer memory. The equations must be sparse in one or the other sense. For instance, suitable sparsity is low degree polynomials or polynomials that admit only bounded number of terms. In this article we focus on l-sparse equations over finite fields as they are defined by (1). This definition allows more operating freedom and generally results in a more efficient solution than with Gröbner basis algorithms. Moreover bounds on the problem average complexity, which is the goal of the present research, are relatively easy to get.

The author is grateful to several anonymous referees at SCC 2010 and "Mathematics in Computer Science" for numerous suggestions on improving the presentation. This is a full paper, the extended abstract is in SCC 2010 [23].

1.2 How to write equations

Let Y be an ordered string of variables and a be an F_q-vector of the same length. We say that a is a vector in variables Y, or Y-vector, if the entries of a may be assigned to the variables Y, for instance, in case of fixation. We look for the set of all solutions to (1) over F_q. Therefore, we only consider for f_i polynomials of degree at most $q-1$ in each variable.

The main step of the present method is the Search Algorithm; see Section 8. It repeatedly checks whether the system $f_i(X_i) = 0, Y = a$, for some subsets $Y \subseteq X$ and Y-vectors a, has any solution over F_q. If there is a solution, then we say $f_i(X_i) = 0, Y = a$ consistent over F_q or simply consistent.

One may choose to work with the polynomials $f_i(X_i)$. The decision problem whether $f_i(X_i) = 0, Y = a$ is consistent over F_q may then be solved with any Gröbner basis algorithm. At least for low q, it may be more convenient in practice to deal with the local solutions V_i over F_q. That is the set of X_i-vectors, where f_i is zero. In polynomial algebra terms, V_i are common zeroes of the polynomials $f_i(X_i)$, $x^q - x$, $x \in X_i$. The sets V_i obviously determine all global solutions over F_q.

As one needs at most q^l trials to compute the set V_i and q, l are supposed to be fixed from the beginning, the computation for (1) requires $O(m)$ field operations. However, at most $l\lceil \log_2 q \rceil q^l$ bits is necessary to keep V_i and that may be expensive for larger q. The memory requirement may be

reduced to at most $q^l + q^{l-1} \ldots + q$ bits per equation with some pre-computation before applying the Search Algorithm; see Section 8.

1.3 Probabilistic model

Given q, n, m, and $l_1, \ldots, l_m \leq l$, uniform distribution on instances is assumed, that is every instance has the same probability. As any particular information on equations is beforehand assumed unknown, this looks the most fair probabilistic model to compute expected complexities. The uniformity is equivalent to

1. the equations in (1) are independently generated. Each equation $f_i(X_i) = 0$ is determined by
2. the subset X_i of size l_i taken uniformly at random from the set of all possible l_i-subsets of X, that is with the probability $\binom{n}{l_i}^{-1}$,
3. and the polynomial f_i taken uniformly at random and independently of X_i from the set of all polynomials of degree $\leq q-1$ in each of variables X_i. In other words, with the equal probability $q^{-q^{l_i}}$.

Running time of any deterministic solving algorithm is a random variable under that model. We assume that m/n tends to $d \geq 1$ as q and l are fixed and n tends to infinity.

Table 1. Algorithms' running time: $q = 2$ and $m = n$.

l	3	4	5	6
the worst case, [12]	1.324^n	1.474^n	1.569^n	1.637^n
Gluing1, expectation, [20]	1.262^n	1.355^n	1.425^n	1.479^n
Gluing2, expectation, [20]	1.238^n	1.326^n	1.393^n	1.446^n
Agreeing-Gluing, expectation, [21]	1.113^n	1.205^n	1.276^n	1.334^n
r	2	3	3	4
Weak Improved Agreeing-Gluing, expectation	1.029^n	1.107^n	1.182^n	1.239^n

2 Previous Ideas

One earlier method [20] is based on subsequent computing solutions U_k to the equation subsystems: $f_1(X_1) = 0, \ldots, f_k(X_k) = 0$ for $k = 1, \ldots, m$. Gluing procedure extends instances U_k to instances U_{k+1} by walking throughout a search tree. In the end, all system solutions are U_m. The running time is determined by the maximal of $|U_k|$. Gluing2 is a time-memory trade-off variation of the basis Gluing1 Algorithm. See Table 1 for their running time expectation in case of n Boolean equations in n variables and a variety of l. Any instance of (1) may be encoded by a CNF formula with the clause length of at most $k = \lceil \log_2 q \rceil \, l$ and in $\lceil \log_2 q \rceil \, n$ Boolean variables; see Section 6. Therefore worst case complexity bounds in [12] for k-SAT are also worst case bounds for equations (1). In Boolean case($q = 2$) they are shown in the first line of Table 1.

In Agreeing-Gluing Algorithm [21] we only extend those intermediate solutions from U_k that do not contradict with each of $f_{k+1}(X_{k+1}) = 0, \ldots, f_m(X_m) = 0$. That makes lots of search tree branches cut and implies a better average time complexity.

3 New Approach

The new method has two variations. Let Z_r denote variables that occur in at least r equations (1), and W_r denote Z_r-vectors that contradict none of (1). In Weak IAG Algorithm(WIAG), see Section 7, r is a parameter to be chosen to minimize the run-time. The vectors W_r are generated by the Search Algorithm, see Section 8. In case $r \geq 3$, the variables Z_r are substituted by the entries of $a \in W_r$. New equations in a smaller variable set $X \setminus Z_r$ are encoded by a CNF and deterministic local search algorithm, see Section 6, is applied to find all solutions. For $r = 1$, W_1 are already the system solutions. For $r = 2$, the solutions are easy to generate from W_2; see Lemma 2 below.

This variation is evaluated in Sections 10 and 11 in case $l_i = l$. Two last lines in Table 1 show the expected complexity of the Weak IAG Algorithm and the optimal value of r. The Agreeing-Gluing Algorithm [21] is a particular case of the present method for $r = 1$.

In Strong IAG Algorithm(SIAG) the largest r, where Z_r is not empty, is taken. The vectors W_r are generated by the Search Algorithm. For each $a \in W_r$ the variables Z_r are substituted by the entries of a. New l-sparse equations in a smaller variable set $X \setminus Z_r$ are to solve. For each of them assignments to $Z_{r-1} \setminus Z_r$ that contradict none of the equations are found with the Search Algorithm. Thus, one recursively computes $W_{r-1}, \ldots, W_2$. All system solutions are then easy to deduce.

The latter variation should be faster, as the Search algorithm is on the average faster in comparison with the local search. However, the estimation of the Strong IAG Algorithm should probably require more sophisticated tools. The work still in progress.

4 Related Methods

Gröbner basis algorithm was designed to work with general algebraic equation systems over any ground field; see [4, 14, 10, 11]. It may be used to solving them. The running time is bounded by a value proportional to $\binom{n+D}{D}^{\omega}$ ground field operations, where ω is the exponent in matrix multiplication complexity. The estimate simplifies to $\binom{n}{D}^{\omega}$ for any Boolean equations [3]. The parameter D, called regularity degree, is only computed for semi-regular equations as they are defined in [3]. Theoretical complexity of the Gröbner basis algorithms as F4 or F5 on general polynomial equation systems remains unknown. It is also unknown whether an average equation system behaves semi-regularly, though this seems plausible [3].

Let l be fixed while $n = m$ tending to infinity. For l-sparse Boolean equation systems each polynomial in (1) admits at most 2^l monomials. Then each row in Macaulay matrices has bounded number of nonzero terms. Wiedemann algorithm [26] may likely be used to do the linear algebra step. We then put $\omega = 2$. The regularity degree for semi-regular Boolean equations in case $n = m$ was estimated as $D = \alpha_d n + o(n)$, where α_d depends on the equations maximal algebraic degree d. So that $\alpha_2 = 0.09, \alpha_3 = 0.15, \alpha_4 = 0.2$ and so on; see [2]. By estimating the binomial coefficient, the complexity is then $2^{2H(\alpha_d)n}$ up to a polynomial factor, where $H(\alpha)$ is the binary entropy function. One can see that only for quadratic semi-regular polynomials the running time is lower than 2^n, brute force complexity, and equal to 1.832^n bit operations. By computing the exact value of the regularity degree, the conjectural running time still exceeds the brute force complexity for $n = 200$; see [1]. The best heuristic bound is of order 1.724^n [27], where the method was combined with variable guessing.

In contrast to [2, 3, 27], when it comes to average equation systems, our estimates are unconditional and rigorous mathematical statements. They are very low exponential functions themselves

even in non-quadratic case; see Table 1. Table 2 presents extended data. It shows c_l for a larger variety of l, where the expected complexity, computed at $r = 2$, on Boolean l-sparse equations is c_l^n . That compares favorably with the above estimates by Gröbner basis algorithms for quadratic sparse polynomials at least for $l \leq 19$.

Table 2. IAG Algorithm($r = 2$) base constant c_l, $q = 2$ and $m = n$.

l	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
c_l	1.029	1.118	1.191	1.252	1.304	1.347	1.385	1.418	1.448	1.474	1.497	1.518	1.538	1.555	1.571	1.586	1.600

Sparse equations may be encoded by a CNF formula and solved with a SAT-solving software. The asymptotical complexity of modern SAT-solvers, as MiniSat [9], is unknown, though they may be fast in practice [7, 25] for relatively low parameters.

5 Notation, Basic Lemma and Example

Let $Y \subseteq X$ be a subset of variables. The following statement is the basement for the IAG Algorithm complexity analysis in Section 10.

Lemma 1. *Let W be the set of Y-vectors consistent with each equation* (1)*. Let $X_1, X_2, \ldots, X_m$ be fixed and polynomials $f_1, f_2, \ldots, f_m$ be taken uniformly at random as in Section* 1.3*. Then the expectation of $|W|$ is given by*

$$\boldsymbol{E}_{f_1,\ldots,f_m}|W| = q^{|Y|} \prod_{i=1}^{m} \left(1 - (1 - \frac{1}{q})^{q^{|X_i \setminus Y|}}\right).$$

Proof. Let a be a Y-vector. We compute $\mathbf{Pr}(a \in W)$, the probability that $f_i(X_i) = 0, Y = a$ is consistent for all i. As f_i are independent,

$$\mathbf{Pr}(a \in W) = \prod_{i=1}^{m} \mathbf{Pr}(f_i(X_i) = 0, Y = a \text{ consistent}).$$

Upon fixation $Y = a$, the polynomial $f_i(X_i)$ produces a uniformly random polynomial in variables $X_i \setminus Y$. It follows from Lemma 4 that

$$\mathbf{Pr}(f_i(X_i) = 0, Y = a \text{ consistent}) = 1 - (1 - \frac{1}{q})^{q^{|X_i \setminus Y|}}$$

and this value doesn't depend on a. So

$$\mathbf{E}_{f_1,\ldots,f_m}|W| = \sum_a \mathbf{Pr}(a \in W) = q^{|Y|}\mathbf{Pr}(a \in W) = q^{|Y|} \prod_{i=1}^{m} \left(1 - (1 - \frac{1}{q})^{q^{|X_i \setminus Y|}}\right).$$

That proves the Lemma. □

According to Section 1.3, $X_1, X_2, \ldots, X_m$ are random subsets in X. Therefore the full expectation of $|W|$ is given by

$$\mathbf{E}|W| = \mathbf{E}_{X_1,\ldots,X_m}(\mathbf{E}_{f_1,\ldots,f_m}|W|) = \mathbf{E}_{X_1,\ldots,X_m}\left(q^{|Y|}\prod_{i=1}^{m}\left(1-(1-\frac{1}{q})^{q^{|X_i\setminus Y|}}\right)\right) \tag{2}$$

by Lemma 5 below. The set Y may depend on the subsets $X_1, X_2, \ldots, X_m$ and therefore be random. Let $Z_r(k)$ be the set of variables that appear in at least r of $X_1, \ldots, X_k$ and let $W_r(k)$ be $Z_r(k)$-vectors consistent with each equation $f_i(X_i) = 0$. We have $Z_r(r) \subseteq Z_r(r+1) \subseteq \ldots \subseteq Z_r(m) = Z_r$. The Search Algorithm extends instances $W_r(k)$ to $W_r(k+1)$, where the output is $W_r = W_r(m)$. From (2)

$$\mathbf{E}|W_r(k)| = \mathbf{E}_{X_1,\ldots,X_m}\left(q^{|Z_r(k)|}\prod_{i=1}^{m}\left(1-(1-\frac{1}{q})^{q^{|X_i\setminus Z_r(k)|}}\right)\right). \tag{3}$$

The maximal of these expectations upper bounds the complexity of first stage of the WIAG Algorithm up to a polynomial factor. It is estimated in Section 10. The second stage complexity for $r \geq 3$ is similarly estimated in Section 11. Three cases should be studied separately.

Case $r = 1$. Then $U_m = W_1$, all system solutions in variables $X_1 \cup \ldots \cup X_m$. Extending $W_1(k)$ to $W_1(k+1)$ by walking over a search tree is the Agreeing-Gluing Algorithm [21].

Case $r = 2$. Remark that variables in different $X_i \setminus Z_2$ are pairwise different. So every $a \in W_2$ is extendable to at least one solution of (1). Moreover the following statement holds.

Lemma 2. *Let $X_{i_1}, X_{i_2}, \ldots, X_{i_s}$ be all variable sets such that $X_{i_j} \nsubseteq Z_2$. After reordering of variables it holds that*

$$U_m = \bigcup_{a\in W_2} \{a\} \times V_{i_1}(a) \times V_{i_2}(a) \ldots \times V_{i_s}(a), \tag{4}$$

where $V_i(a)$ are all projections to $X_i \setminus Z_2$ of the F_q-solutions to $f_i(X_i) = 0, Z_2 = a$.

Example. Let the system of three Boolean equations be given with their local solutions:

x_1	x_2	x_3
0	0	1
1	0	0
1	1	1
1	0	1

,

x_3	x_4	x_5
0	0	0
1	0	1
1	1	1
0	0	1

,

x_5	x_6	x_7
0	0	0
0	1	1
1	1	0
1	0	1

.

We see $Z_2(2) = \{x_3\}$ and $W_2(2) = \{0, 1\}$, so $Z_2(3) = \{x_3, x_5\}$ and $W_2(3) = \{00, 01, 11\}$. The directed products (4) are:

x_3	x_5	x_1	x_2		x_4		x_6	x_7
0	0	1	0	×	0	×	0	0
							1	1
0	1	1	0	×	0	×	1	0
							0	1
1	1	0	0	×	0	×	1	0
		1	1		1		0	1
		1	0					

.

So 16 solutions to the system are represented by three strings: 00, and 01, and 11 related to variables x_3, x_5.

Case $r \geq 3$. The Search Algorithm returns some $a \in W_r$. The variables Z_r are substituted by the entries of a. The problem is represented by a k-CNF with $k = \lceil \log_2 q \rceil \, l$ and in $n_1 = \lceil \log_2 q \rceil \, |X \setminus Z_r|$ Boolean variables. Local search algorithm, described in [8], is used to find all solutions. In worst case, that takes $O((N+1)(2 - \frac{2}{k+1})^{n_1})$ bit operations up to a polynomial factor to find all N solutions; see Section 6.

6 k-CNF and Local Search

Conjunctive normal form(CNF for short) is a conjunction of disjunctions as $x_{i_1}^{(c_1)} \vee \ldots \vee x_{i_k}^{(c_k)}$ called clauses, where $x^{(0)} = x$ and $x^{(1)} = \bar{x}$. The disjunction terms x and $\bar{x}$ are called literals. If the length of each clause in a CNF is at most k, then it is called k-CNF. Let $q = 2$, and $f(x_1, \ldots, x_l) = 0$ be any Boolean equation in l Boolean variables. Let

$$(a_{11}, \ldots, a_{1l}), \ldots, (a_{s1}, \ldots, a_{sl})$$

be all binary strings such that $f(a_{i1}, \ldots, a_{il}) = 1$.

Lemma 3. *Binary string $(b_1, \ldots, b_l)$ is a solution to $f(x_1, \ldots, x_l) = 0$ iff it is a satisfying assignment for the CNF*

$$F_f = (x_1^{(a_{11})} \vee \ldots \vee x_l^{(a_{1l})}) \wedge \ldots \wedge (x_1^{(a_{s1})} \vee \ldots \vee x_l^{(a_{sl})}).$$

If all equations (1) are Boolean, one constructs l-CNF in n variables as $F = \bigwedge_i F_{f_i}$. Binary string $(b_1, \ldots, b_n)$ is a solution to (1) iff it is a satisfying assignment for F.

Generally, elements $a \in F_q$ may be encoded with the numbers $0, \ldots, q-1$. Every variable x in F_q is written with $r = \lceil \log_2 q \rceil$ Boolean variables $y_{r-1}, \ldots, y_0$, such that $x = a$ iff $(y_{r-1}, \ldots, y_0) = (b_{r-1}, \ldots, b_0)$, where $b_i \in \{0, 1\}$ and $a = b_{r-1}2^{r-1} + \ldots + b_0$. So F_q-solutions to $f(x_1, \ldots, x_l) = 0$ are written as binary lr-strings.

As in Lemma 3, for the set of binary lr-strings complimentary to those solutions one constructs a k-CNF in $k = lr$ variables. We do that for each equation in (1). Resulting k-CNF in $n_1 = nr$ variables is a conjunction of them. Solving (1) is equivalent to finding satisfying assignments to that CNF and therefore is NP-hard as it is polynomially equivalent to a k-SAT problem.

Local search is designed to solve satisfiability problems; see [16, 19]. Given a k-CNF in n variables, guess an initial assignment to all variables. Repeat $3n$ times: if the formula is satisfied, then terminate; let there be some clause not being satisfied by the current assignment, then pick one of its literals uniformly at random and flip its value in the current assignment. The probability of finding a satisfying assignment is at least $\frac{2}{3}(\frac{k}{2(k-1)})^n$; see [19], and the expected number of this procedure repetitions before the satisfying assignment if found is $(2 - \frac{2}{k})^n$ up to a polynomial factor.

A deterministic version is described in [8]. Within $\text{poly}(n)(2 - \frac{2}{k+1})^n$ operations the algorithm finds a satisfying assignments or returns the CNF is unsatisfiable. Therefore local search may be used to compute all N solutions in time at most $(N+1)(2 - \frac{2}{k+1})^n$ up to a polynomial factor.

7 Weak IAG Algorithm

Let $r \geq 1$ be such that $Z_r \neq \emptyset$. The sequence of subsets $Z_r(r) \subseteq Z_r(r+1) \subseteq \ldots \subseteq Z_r(m) = Z_r$ defines a rooted search tree. The tree has at most $m - r + 2$ levels numbered $0, r, r+1 \ldots, m$. The

root at level 0 is labeled by $\emptyset$. Vertices at level $k \geq r$ are labeled by $\emptyset$ if $Z_r(k) = \emptyset$ and by vectors $W_r(k)$ if this set is not empty. If $W_r(k) = \emptyset$, then there is no any vertices at level k, so the whole system of equations is inconsistent.

Let $a \in W_r(k)$ be a level k vertex label. It is connected to a level $k+1$ vertex labeled by $b \in W_r(k+1)$ whenever a is a sub-vector of b. Remark that $Z_r(k) \subseteq Z_r(k+1)$.

The search tree for the example system is presented in Fig. 1, where $r = 2$. Level 2 vertices are labeled by $W_2(2) = \{0, 1\}$, vectors in variables $Z_2(2) = \{x_3\}$. The vertices at level 3 are labeled by $W_2(3) = \{00, 01, 11\}$, vectors in variables $Z_2(3) = \{x_3, x_5\}$. We now describe the Algorithm. More formal description of its first stage is in the next Section.

Stage 1(Search Algorithm) It starts at the root. Let the Algorithm be at a level k vertex labeled a. If $k = 0$, then we extend a to all possible $Z_r(r)$-vectors. If $k \geq r$, then we extend a to $Z_r(k+1)$-vectors by trying all possible assignments to variables $Z_r(k+1) \setminus Z_r(k)$.
Let b be one such extension. If $Z_r(k+1) = b$(or $Z_r(r) = b$ if $k = 0$) is consistent with every equation in (1), then $b \in W_r(k+1)$. Return b if $k + 1 = m$. The Algorithm walks to the vertex labeled b. Otherwise, another assignment is taken to extend a. If all the assignments are exhausted and $k = 0$, then stop. If $k > 0$, then the Algorithm backtracks to the level $k-1$. This stage output is $W_r = W_r(m)$. If no vertex at level m is hit , then the system has no solution.

Stage 2 Let the Algorithm achieve a vertex at level m labeled by $a \in W_r$. If $r = 1$, then a is a system solution. If $r = 2$, then the system solutions are deduced with (4). If $r \geq 3$, a system of l-sparse equations in variables $X \setminus Z_r$ after substituting Z_r by constants a is solved with deterministic local search; see Section 6.

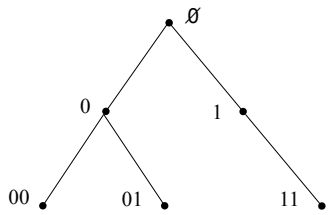

Fig. 1. The search tree.

Remark that instead of all possible assignments to variables $Z_r(k+1) \setminus Z_r(k)$ one may only take the projections of local solutions $f_{k+1}(X_{k+1}) = 0$ to those variables. That slightly accelerates the first stage.

Theorem 1. *Let N be the number of the system solutions.*

1. *The complexity of the first stage is $O\left(m + m\sum_{k=r}^{m} |W_r(k)|\right)$ bit operations. If $r = 1$, then this is the Algorithm's run-time. If $r = 2$, then the Algorithm run-time is the sum of the first stage complexity and $O(N)$.*

2. *The complexity of the second stage is at most* $(N + |W_r|)\ c^{n-|Z_r|}$ *bit operations up to a polynomial factor, where* $c = (2 - \frac{2}{l\lceil \log_2 q \rceil + 1})^{\lceil \log_2 q \rceil}$. *If* $r \geq 3$, *then the Algorithm run-time is the sum of its stages complexities.*

Proof. The complexity of the first stage is

$$mq^{|Z_r(r)|} + m \sum_{k=r}^{m-1} |W_r(k)| q^{|Z_r(k+1) \setminus Z_r(k)|}$$

decisions whether $f_i(X_i) = 0, Y = b$ is consistent for some Y. The latter costs $O(1)$ bit operations. Because $|Z_r(r)|$ and $|Z_r(k+1) \setminus Z_r(k)|$ are at most l, the first statement is then true.

Let $a \in W_r$ and N_a be the number of the system solutions after the variables Z_r being substituted by the entries of a. The complexity of the second stage is

$$\sum_{a \in W_r} (N_a + 1) c^{n-|Z_r|} = (N + |W_r|)\ c^{n-|Z_r|}$$

bit operations up to a polynomial factor. That implies the second statement. □

We realize that $\mathbf{E}\, N = q^{n-m}$ as $\mathbf{E}_{f_1,\ldots,f_m} N = q^{n-m}$ for any fixed variable sets X_i. Under the probabilistic model the values $|W_r(k)|$ are random. The expected complexity of the first stage(and of the whole algorithm for $r = 1, 2$) is proportional to $m + m^2 \max_k \mathbf{E}|W_r(k)|$, where $\mathbf{E}\,|W_r(k)|$ is represented by (3) and estimated in Section 10. For the second stage complexity we have

$$\mathbf{E}\,(N + |W_r|)\ c^{n-|Z_r|} = \mathbf{E}\, q^{n-m} c^{n-|Z_r|} + \mathbf{E}\,|W_r|\ c^{n-|Z_r|}.$$

The expectations of $q^{n-m} c^{n-|Z_r|}$ and $|W_r|\, c^{n-|Z_r|}$ are estimated in Section 11, where the latter may be shown dominates the sum.

For a range of r the estimates are computed with an optimization software like MAPLE; see [15]. One then finds r to minimize the running time expectation. Remark that the computation does not depend on n.

8 General Search Algorithm

Given $Y \subseteq X$, this general Algorithm finds all Y-vectors over F_q that consistent with each of (1). A subset sequence $Y_1 \subseteq Y_2 \subseteq \ldots \subseteq Y_s = Y$ is taken. That defines a search tree. The root is labeled by $\emptyset$, the vertices at levels $1 \leq k \leq s$ are labeled by Y_k-vectors that do not contradict any of (1). We denote them $W(k)$. Vertices a and b at subsequent levels are connected if a is a sub-vector of b. The algorithm walks with backtracking throughout the tree by constructing instances $W(k)$. There are $q^{|Y_{k+1} \setminus Y_k|}$ extensions to any of $W(k)$. Each of them should be checked for consistency with m equations. We represent the Algorithm with a pseudocode. Let, by agreement, $Y_0 = \emptyset$ and Y_0-vector $a = \emptyset$ is consistent with every equation (1). The extension to $a = \emptyset$ with an assignment c to some variables is c.

Procedure EXTEND(k, a).
input: $0 \leq k \leq s - 1$ and a Y_k-vector a consistent with each of (1).
output: all Y-vectors, extensions to a and consistent with each of (1).

1. for every assignment to variables $Y_{k+1} \setminus Y_k$ do
2. extend a to a Y_{k+1}-vector b. Let b be consistent with every equation in (1). If $k+1 < s$, then call recursively EXTEND(k+1,b). If $k+1 = s$, then return b. If b is inconsistent with at least one equation, then take another assignment.
3. If all assignment are exhausted and $k = 0$, then stop. If $k > 1$, then return.

The Search Algorithm is EXTEND$(0, \emptyset)$ and its running time is proportional to

$$m(q^{|Y_1|} + |W(1)|q^{|Y_2 \setminus Y_1|} + |W(2)|q^{|Y_3 \setminus Y_2|} + \ldots + |W(s-1)|q^{|Y_s \setminus Y_{s-1}|})$$

operations. A sequence of subsets that minimizes the running time may be taken. How to do that is generally an interesting open problem. In IAG Algorithms the sequence is $Z_r(r) \subseteq Z_r(r+1) \subseteq \ldots \subseteq Z_r(m) = Z_r$.

To save time and memory, one may solve the decision problems in advance and keep the tracks. For each equation one keeps the decision(1 or 0) on whether $f_i(X_i) = 0, X_i \cap Y_k = a$ is consistent, where a has $q^{|X_i \cap Y_k|}$ possible values and $k = 1, \ldots, s$. As there are at most l different $X_i \cap Y_k$, then that requires at most $q^l + q^{l-1} + \ldots + q$ bits of memory. It is not necessary to keep any local solutions.

In practice, one may want to find weather a Y_k-vector a contradicts the whole system (1) but not only each of the equations taken separately. One then runs the Agreeing Algorithm [18, 24] after the variables Y_k get substituted by constants a. Even if no contradiction is found, one may learn values of some new variables. That improves the method efficiency. However such a variation seems difficult to evaluate.

9 Tools

In this Section we collect miscellaneous auxiliary statements.

Let H be the set of all polynomials over F_q in $l \geq 1$ variables, whose degree in each of its variables is at most $q-1$. Let H_1 be the subset of polynomials $f \in H$, where the equation $f = 0$ has no solutions over F_q.

Lemma 4. *Every polynomial $f \in H$ defines a mapping $f : F_q^l \to F_q$ and vice versa. That is a one-to-one correspondence. So $|H| = q^{q^l}$ and $|H_1| = (q-1)^{q^l}$.*

Proof. The number of polynomials in H and the number of mappings $F_q^l \to F_q$ is q^{q^l}. One proves if a polynomial $f \in H$ defines an identically zero mapping, then all its coefficients are 0. Really, $f(x_1, x_2, \ldots, x_l) = f_{q-1}x_1^{q-1} + \ldots + f_1 x_1 + f_0$, where f_i are polynomials of degree at most $q-1$ in each of $x_2, \ldots, x_l$, they are constants if $l = 1$. Let f be 0 on F_q^l. After fixation of $x_2, \ldots, x_l$ by any constants in F_q, we get a polynomial in x_1 which is 0 on F_q. Therefore, its coefficients are zeros as, otherwise, it can not have more than $q-1$ different roots. That proves the statement for $l = 1$. If $l > 1$, then for the same reason the polynomials f_i are 0 on F_q^{l-1} and by induction they all have zero coefficients. We conclude f has zero coefficients.

If $f_1, f_2 \in H$ define the same mapping, then $f_1 - f_2$ defines an identically zero mapping and, therefore, $f_1 = f_2$. That proves the above correspondence is one-to-one. The polynomials from H_1 correspond to the mappings without zero-values. Their number is $(q-1)^{q^l}$. □

Let $\eta = \eta(x, y)$ be any variable that depends on two independent random variables x and y with finite number of values. Then $\mathbf{E}_y \eta$ denotes the expectation of η, where y is generated to its initial distribution. So $\mathbf{E}_y \eta$ is a function in x.

Lemma 5. [21] *For the full expectation of $\eta = \eta(x, y)$ we have*

$$\boldsymbol{E}_{x,y}\,\eta = \boldsymbol{E}_x(\boldsymbol{E}_y(\eta)).$$

Random Allocations Theory studies random allocations of particles(balls, shots) into boxes, see [13]. Let k complexes of particles be independently and uniformly allocated into $n \le 1$ boxes, $l_i \le n$ particles at the i-th allocation. This means that at the i-th allocation any l_i boxes are occupied with the equal probability $\binom{n}{l_i}^{-1}$. This is how variable sets $X_1, \ldots, X_m$ are generated according to Section 1.3. We will need to upper bound the probability of several events defined by such allocations. There is some useful theory in [13] developed mostly for allocations of particles one after the other, that is by complexes of size 1. The following Lemma relates the probability of the same event under the two types of allocation.

Let $\nu_1, \ldots, \nu_n$ be the string of box frequencies, that is ν_i is the number of particles in the i-th box. Let $A = A(\nu_1, \ldots, \nu_n)$ be any event depending on ν_i. Let also $\mathbf{Pr}(A|\ l_1, \ldots, l_k)$ denote the probability of the event A under the allocation by complexes of $l_1, \ldots, l_k$ particles.

Lemma 6.

$$\boldsymbol{Pr}(A|\ l_1, \ldots, l_k) \le \frac{\boldsymbol{Pr}(A|\ 1, \ldots, 1)}{\prod_{i=1}^{k} (1 - 1/n) \ldots (1 - (l_i - 1)/n)},$$

where $\boldsymbol{Pr}(A|\ 1, \ldots, 1)$ is the probability of A under $L = l_1 + \ldots + l_k$ particles are allocated one after the other.

Proof. Let L particles be independently and uniformly allocated into n boxes one after the other. Let B denote the event that the first l_1 particles were allocated into different boxes, the following l_2 were allocated into different boxes and etc, until the last l_k particles were allocated into different boxes. In other words, the event B occurs if the particles are allocated by complexes of size $l_1, l_2, \ldots, l_k$. Then $\mathbf{Pr}(B) = \prod_{i=1}^{k} (1 - 1/n) \ldots (1 - (l_i - 1)/n)$ as the particles were allocated independently. By the complete probability formula we get

$$\begin{aligned}\mathbf{Pr}(A|\, 1, \ldots, 1) &= \mathbf{Pr}(B)\ \mathbf{Pr}(A|\ B) + \mathbf{Pr}(\bar{B})\ \mathbf{Pr}(A|\ \bar{B}) \\ &\ge \mathbf{Pr}(B)\ \mathbf{Pr}(A|\ B) = \mathbf{Pr}(B)\ \mathbf{Pr}(A|\ l_1, \ldots, l_k)\end{aligned}$$

as $\mathbf{Pr}(A|\, B) = \mathbf{Pr}(A|\ l_1, \ldots, l_k)$. That proves the Lemma. □

Let $f(z) = \sum_{k=0}^{\infty} a_k\, z^k$, where real $a_k \ge 0$, be a non-zero analytic function. We denote $f^n(z) = \sum_{k=0}^{\infty} a_{n,k}\, z^k$ for any natural n.

Lemma 7. 1. *For any real $z > 0$*

$$a_{n,k} \le \frac{f^n(z)}{z^k} = e^{n \ln f(z) - k \ln z}. \tag{5}$$

2. *Let $a_i, a_j > 0$ for some $i \ne j$. Then at any real $z > 0$ the derivative of $\frac{z f'(z)}{f(z)}$ is positive.*

Proof. The expansion of f^n has only nonnegative coefficients, so $a_{n,k}\, z^k \le f^n(z)$. That proves the first statement. To prove the second statement one represents

$$\left(\frac{zf'}{f}\right)' = \frac{zf''f - zf'^2 + f'f}{f^2} = \frac{\sum_{l=0}^{\infty} b_l z^l}{f^2},$$

where $b_u = \sum_{k=0}^{u} (u - k + 1)(u - 2k + 1) a_k a_{u-k+1} = \sum_{k=0}^{\lfloor \frac{u+1}{2} \rfloor} (u - 2k + 1)^2 a_k a_{u-k+1}$. Therefore $b_{i+j-1} > 0$. That proves the statement. □

To minimize the bound (5) one may take a positive root z_0 to $(n \ln f(z) - k \ \ln z)' = 0$ or, equivalently,

$$n\frac{zf'(z)}{f(z)} = k.$$

if there exist any. In case there is only one root, the Lemma estimate is proportional to the main term of the asymptotic expansion for $a_{n,k}$ with the saddle point method as n and k tend to infinity; see [5]. Lemma 7 estimate is then asymptotically close to the real value of $a_{n,k}$. We use rather (5) than the saddle point method in Lemmas 8, 9 and 13.

Let $\boldsymbol{\mu}_r = \boldsymbol{\mu}_r(t,n)$ be the number of boxes with just r particle after uniform allocation of t particles into n boxes one after the other. Let $\boldsymbol{\mu}'_r(l_1,\ldots,l_k,n)$ be the number of boxes with just r particle after uniform allocation of k complexes by $l_1,\ldots,l_k$ particles into n boxes. The probability of some events related to $\boldsymbol{\mu}'_r$ is required in what follows. We here estimate the probability of them for the allocation of particles one after the other, that is in case of variables $\boldsymbol{\mu}_r$. Then Lemma 6 is used in Section 10.

Let $\mathbf{E}\,(x_1^{\boldsymbol{\mu}_{r_1}}\ldots x_s^{\boldsymbol{\mu}_{r_s}})$ be the expectation of the random variable $x_1^{\boldsymbol{\mu}_{r_1}}\ldots x_s^{\boldsymbol{\mu}_{r_s}}$, where $x_1,\ldots,x_s$ are any variables. By definition,

$$\mathbf{E}\,(x_1^{\boldsymbol{\mu}_{r_1}}\ldots x_s^{\boldsymbol{\mu}_{r_s}}) = \sum_{k_1,\ldots,k_s} \mathbf{Pr}(\boldsymbol{\mu}_{r_1} = k_1,\ldots,\boldsymbol{\mu}_{r_s} = k_s)\; x_1^{k_1}\ldots x_s^{k_s}.$$

Theorem 2 in Chapter 2, Section 1 of [13] states

$$\sum_{t=0}^{\infty}\frac{n^t\,z^t}{t!}\mathbf{E}\,(x_1^{\boldsymbol{\mu}_{r_1}}\ldots x_s^{\boldsymbol{\mu}_{r_s}}) = \left(e^z + \frac{z^{r_1}}{r_1!}(x_1-1) + \ldots + \frac{z^{r_s}}{r_s!}\,(x_s-1)\right)^n. \tag{6}$$

In particular, we get

$$\sum_{t=0}^{\infty}\frac{n^t\,z^t}{t!}\mathbf{E}\,(x_0^{\boldsymbol{\mu}_0}\ldots x_{r-1}^{\boldsymbol{\mu}_{r-1}}) = \left(e^z + (x_0-1) + \ldots + \frac{z^{r-1}}{(r-1)!}\,(x_{r-1}-1)\right)^n.$$

We there put $x_0 = \ldots = x_{r-1} = 0$ and get

$$\left(e^z - 1 - z\ldots - \frac{z^{r-1}}{(r-1)!}\right)^n = \sum_{t=nr}^{\infty}\frac{n^t z^t}{t!}\,\mathbf{Pr}(\boldsymbol{\mu}_0 = 0,\ldots,\boldsymbol{\mu}_{r-1} = 0)$$

as $\mathbf{Pr}(\boldsymbol{\mu}_0 = 0,\ldots,\boldsymbol{\mu}_{r-1} = 0) = 0$ for $t < nr$. Let $g(x) = e^x - 1 - x\ldots - \frac{x^{r-1}}{(r-1)!}$.

Lemma 8. *Let $r \geq 1$. For any natural number $t \geq nr$*

$$\boldsymbol{Pr}(\boldsymbol{\mu}_0(t,n) = 0,\ldots,\boldsymbol{\mu}_{r-1}(t,n) = 0) \leq \frac{g^n(x)\;t!}{x^t\;n^t},$$

where x is the only nonnegative root of the equation

$$n\frac{xg'(x)}{g(x)} = t. \tag{7}$$

Proof. We have $g(x) = \frac{x^r}{r!} + \frac{x^{r+1}}{(r+1)!} + \ldots$. So $\frac{xg'(x)}{g(x)}$ tends to r as $x \to 0^+$. Also it tends to ∞ as $x \to \infty$. By Lemma 7, the derivative of $\frac{xg'(x)}{g(x)}$ is positive at positive x. Therefore, the equation (7) has just one nonnegative root for $t \geq nr$.

For $t > nr$ the statement is true by Lemma 7. Let $t = nr$, then the root $x = 0$. One sees that $\frac{g^n(x)\, t!}{x^t\, n^t}$ is defined at $x \to 0^+$ and equal to $\frac{(nr)!}{(r!)^n\, n^{nr}}$. One directly computes

$$\mathbf{Pr}(\boldsymbol{\mu}_0(nr, n) = 0, \ldots, \boldsymbol{\mu}_{r-1}(nr, n) = 0) = \frac{(nr)!}{(r!)^n\, n^{nr}}.$$

The statement is true for any $t \geq nr$. That proves the Lemma. □

Let $r \geq 2$. It follows from (6) that

$$\sum_{t=0}^{\infty} \frac{n^t\, z^t}{t!} \mathbf{E}\left(x_1^{\boldsymbol{\mu}_1} \ldots x_{r-1}^{\boldsymbol{\mu}_{r-1}}\right) = \left(e^z + z(x_1 - 1) + \ldots + \frac{z^{r-1}}{(r-1)!}(x_{r-1} - 1)\right)^n. \tag{8}$$

Substitute $x_i = x^i$ for $i = 1, \ldots, r-1$. Then

$$\begin{aligned} &\sum_{t=0}^{\infty} \frac{n^t\, z^t}{t!} \mathbf{E}\left(x^{\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1}}\right) \\ &= \left[e^z - \left(z + \ldots + \frac{z^{r-1}}{(r-1)!}\right) + \left(zx + \ldots + \frac{(zx)^{r-1}}{(r-1)!}\right)\right]^n. \end{aligned}$$

By the definition of expectation,

$$\mathbf{E}\left(x^{\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1}}\right) = \sum_{k=0}^{t} x^k\, \mathbf{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = k\right)$$

because $\mathbf{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = k\right) = 0$ if $k > t$. We denote zx by x and get from the last two identities that

$$\begin{aligned} &\sum_{t \geq k} \frac{n^t\, z^{t-k}\, x^k}{t!} \mathbf{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = k\right) \\ &= \left[e^z - \left(z + \ldots + \frac{z^{r-1}}{(r-1)!}\right) + \left(x + \ldots + \frac{x^{r-1}}{(r-1)!}\right)\right]^n, \end{aligned}$$

where the left hand side sum is over t and k such that $t \geq k \geq 0$. We now put $z = 0$ and get

$$\left(1 + x + \ldots + \frac{x^{r-1}}{(r-1)!}\right)^n = \sum_{t=0}^{(r-1)n} \frac{n^t\, x^t}{t!} \mathbf{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = t\right).$$

We remark that the probability is zero if $t > (r-1)n$. Let $h(y) = 1 + y \ldots + \frac{y^{r-1}}{(r-1)!}$.

Lemma 9. *Let $r \geq 2$. For any integer number t such that $0 \leq t \leq (r-1)n$ we have*

$$\boldsymbol{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = t\right) \leq \frac{h^n(y)}{y^t} \frac{t!}{n^t},$$

where y is the only nonnegative root (including ∞) of the equation

$$n\frac{yh'(y)}{h(y)} = t. \tag{9}$$

Proof. $\frac{yh'(y)}{h(y)}$ tends to 0 as $y \to 0^+$. Also it tends to $r-1$ as $y \to \infty$. By Lemma 7, the derivative of $\frac{yh'(y)}{h(y)}$ is positive at positive y. Therefore, the equation (9) has just one nonnegative root for $0 \le t \le (r-1)n$ including $y = 0$ for $t = 0$ and $y = \infty$ for $t = (r-1)n$.

Let $0 < t < (r-1)n$. The equation (9) has the only positive root. The estimate is true by the first statement of Lemma 7. Let $t = 0$, then $y = 0$ and the Lemma is true as both the sides of the inequality are 1. Let $t = (r-1)n$, then $y = \infty$. The right hand side of the inequality is defined at $y = \infty$ and equal to $\frac{t!}{((r-1)!)^n\, n^t}$. By direct calculation,

$$\mathbf{Pr}\left(\boldsymbol{\mu}_1 + 2\boldsymbol{\mu}_2 + \ldots + (r-1)\boldsymbol{\mu}_{r-1} = t\right) = \mathbf{Pr}\left(\boldsymbol{\mu}_{r-1}(t,n) = n\right) = \frac{t!}{((r-1)!)^n\, n^t}.$$

That proves the Lemma. □

From the Stirling approximation to $k!$, see [6], we get

Lemma 10. *For every integer number $k \ge 0$ it holds that*

$$k^k e^{-k} \le k! \le k^k e^{-k}\sqrt{2\pi(k+1)}.$$

10 Complexity Estimate. Stage 1

Let $r \ge 2$ and $l_i = l$ for all $i = 1, \ldots, m$. We now estimate the expectation of $|W_r(k)|$ with (3). Its maximum in k will be estimated with (17). According to the probabilistic model, $X_1, \ldots, X_m$ are uniformly allocated into the whole variable set X of size n. So we use the language of particle allocation into n boxes from now. In particular, $Z_r(k)$ is the set of boxes with at least r particles after uniform allocation by k complexes of size l. We split the product in (3):

$$\mathbf{E}|W_r(k)| = \mathbf{E}_{X_1,\ldots,X_m}\left(q^{|Z_r(k)|}\prod_{i=1}^{k}\left(1-(1-\frac{1}{q})^{q^{|X_i\setminus Z_r(k)|}}\right)\prod_{j=k+1}^{m}\left(1-(1-\frac{1}{q})^{q^{|X_j\setminus Z_r(k)|}}\right)\right).$$

We say the event $A = A(U, t_1, \ldots, t_k)$ occurs if $Z_r(k) = U$ and $|X_i \setminus U| = t_i$, where $i = 1, \ldots, k$; see Fig. 2. With the conditional expectation formula we get

$$\mathbf{E}|W_r(k)| = \sum_{U}\sum_{t_1,\ldots,t_k} q^{|U|}\prod_{i=1}^{k}\left(1-(1-\frac{1}{q})^{q^{t_i}}\right)\mathbf{E}(A)\,\mathbf{Pr}(A), \tag{10}$$

where U runs over all subsets of X and $0 \le t_i \le l$ and we denoted

$$\mathbf{E}(A) = \mathbf{E}_{X_1,\ldots,X_m}\left(\prod_{j=k+1}^{m}\left(1-(1-\frac{1}{q})^{q^{|X_j\setminus Z_r(k)|}}\right)\middle| A\right).$$

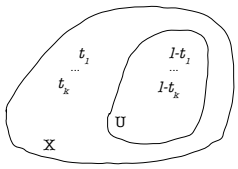

Fig. 2. The event A.

So

$$\mathbf{E}(A) = \mathbf{E}_{X_{j+1},\ldots,X_m} \prod_{j=k+1}^{m} \left(1 - (1-\frac{1}{q})^{q^{|X_j \setminus U|}}\right)$$
$$= \prod_{j=k+1}^{m} \mathbf{E}_{X_j} \left(1 - (1-\frac{1}{q})^{q^{|X_j \setminus U|}}\right) = \left(\mathbf{E}_{X_j} \left(1 - (1-\frac{1}{q})^{q^{|X_j \setminus U|}}\right)\right)^{m-k}.$$

We remark that $\mathbf{E}(A)$ only depends on the size u of the set U, and not on the set itself. Let $u = \beta n$, where $0 \le \beta \le 1$, then

$$\mathbf{E}_{X_j}(1 - (1-\frac{1}{q})^{q^{|X_j \setminus U|}}) = 1 - \sum_{t=0}^{l} \frac{\binom{u}{l-t}\binom{n-u}{t}}{\binom{n}{l}}(1-\frac{1}{q})^{q^t} = 1 - \sum_{t=0}^{l} \frac{\binom{\beta n}{l-t}\binom{n-\beta n}{t}}{\binom{n}{l}}(1-\frac{1}{q})^{q^t}.$$

By taking $\lim_{n\to\infty}$, we get

Lemma 11. *As n tends to ∞*

$$\boldsymbol{E}_{X_j}(1 - (1-\frac{1}{q})^{q^{|X_j \setminus U|}}) = F(\beta) + O(\frac{1}{n}),$$

where $F(\beta) = 1 - \sum_{t=0}^{l} \binom{l}{t} \beta^{l-t}(1-\beta)^t(1-\frac{1}{q})^{q^t}$ and $O(\frac{1}{n})$ is uniformly bounded in β.

Lemma 11 implies $\mathbf{E}(A) \le (F(\beta)+\epsilon)^{m-k}$, where ϵ is any positive number and n is big enough. Let $L = lk = \alpha n$, where $0 \le \alpha \le dl$. So $\frac{m-k}{n} = \frac{m}{n} - \frac{\alpha}{l}$. As $\frac{m}{n}$ tends to d, then

$$\mathbf{E}(A) \le (F(\beta)+\epsilon)^{(d-\frac{\alpha}{l})n} \tag{11}$$

for any positive ϵ and for all big enough n. We now estimate the probability of the event A. Let as above $|U| = u$.

Lemma 12. *Let $L = lk$ and $T = t_1 + \ldots + t_k$. Then*

$$\boldsymbol{Pr}(A) \le \frac{\left(\frac{u}{n}\right)^{L-T}\left(\frac{n-u}{n}\right)^{T} P_1(L-T,u)\; P_2(T,n-u)\; \prod_{i=1}^{k}\binom{l}{t_i}}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^k},$$

where

$$P_1(L-T,u) = \boldsymbol{Pr}(\boldsymbol{\mu}_0(L-T,u) = 0, \ldots, \boldsymbol{\mu}_{r-1}(L-T,u) = 0),$$
$$P_2(T,n-u) = \boldsymbol{Pr}(\boldsymbol{\mu}_1(T,n-u) + 2\,\boldsymbol{\mu}_2(T,n-u) + \ldots + (r-1)\,\boldsymbol{\mu}_{r-1}(T,n-u) = T).$$

Proof. Let $u = 0$, then $T = L$ and A occurs if in the allocation of $X_1, \ldots, X_k$ every variable is hit at most $r-1$ times. So by Lemma 6,

$$\mathbf{Pr}(A) \le \frac{P_2(L,n)}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^k},$$

and the statement is true. Let $u = n$, then $T = 0$ and A occurs if in the allocation of $X_1, \ldots, X_k$ every variable is hit at least r times. So by Lemma 6,

$$\mathbf{Pr}(A) \le \frac{P_1(L,n)}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^k},$$

and the statement is true. So we can assume $0 < u < n$. We say the event B occurs if $|X_i \setminus U| = t_i$ for $i = 1, \ldots, k$. Then $\mathbf{Pr}(A) = \mathbf{Pr}(B)\mathbf{Pr}(A|B)$.

$$\mathbf{Pr}(B) = \prod_{i=1}^{k} \mathbf{Pr}(|X_i \setminus U| = t_i) = \prod_{i=1}^{k} \frac{\binom{u}{l-t_i}\binom{n-u}{t_i}}{\binom{n}{l}} =$$

$$= \prod_{i=1}^{k} \binom{l}{t_i} \left(\frac{u}{n}\right)^{l-t_i} \left(\frac{n-u}{n}\right)^{t_i} \frac{(1-\frac{1}{u})\ldots(1-\frac{l-t_i-1}{u})(1-\frac{1}{n-u})\ldots(1-\frac{t_i-1}{n-u})}{(1-\frac{1}{n})\ldots(1-\frac{l-1}{n})}$$

$$= \frac{\left(\frac{u}{n}\right)^{L-T}\left(\frac{n-u}{n}\right)^{T}\prod_{i=1}^{k}\binom{l}{t_i}\prod_{i=1}^{k}(1-\frac{1}{u})\ldots(1-\frac{l-t_i-1}{u})\prod_{i=1}^{k}(1-\frac{1}{n-u})\ldots(1-\frac{t_i-1}{n-u})}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^k}.$$

The event $A|B$ occurs if and only if the following two events A_1 and A_2 occur simultaneously. First, the complexes of $l-t_1, \ldots, l-t_k$ particles are allocated into $|U| = u$ boxes, where each box is occupied by at least r particles. Second, the complexes of $t_1, \ldots, t_k$ particles are allocated into $|X \setminus U| = n-u$ boxes, where each box is occupied by at most $r-1$ particles; see Fig. 2. These are independent events. Therefore $\mathbf{Pr}(A|B) = \mathbf{Pr}(A_1)\mathbf{Pr}(A_2)$.

The event A_1 occurs if and only if $\boldsymbol{\mu}'_i(l-t_1, \ldots, l-t_k, u) = 0$ for $i = 0, \ldots, r-1$. The event A_2 occurs if and only if $\boldsymbol{\mu}'_i(t_1, \ldots, t_k, n-u) = 0$ for $i \ge r$. The latter is equivalent to

$$\boldsymbol{\mu}'_1(t_1, \ldots, t_k, n-u) + 2\,\boldsymbol{\mu}'_2(t_1, \ldots, t_k, n-u) + \ldots + (r-1)\,\boldsymbol{\mu}'_{r-1}(t_1, \ldots, t_k, n-u) = T.$$

See the definition of $\boldsymbol{\mu}'_s$ in Section 9. By Lemma 6,

$$\mathbf{Pr}(A_1) \le \frac{P_1(L-T,u)}{\prod_{i=1}^{k}(1-\frac{1}{u})\ldots(1-\frac{l-t_i-1}{u})}$$

and

$$\mathbf{Pr}(A_2) \le \frac{P_2(T,n-u)}{\prod_{i=1}^{k}(1-\frac{1}{n-u})\ldots(1-\frac{t_i-1}{n-u})}$$

So $\mathbf{Pr}(A) = \mathbf{Pr}(B)\mathbf{Pr}(A|B) =$

$$= \mathbf{Pr}(B)\mathbf{Pr}(A_1)\mathbf{Pr}(A_2) \le \frac{\left(\frac{u}{n}\right)^{L-T}\left(\frac{n-u}{n}\right)^{T} P_1(L-T,u)\; P_2(T,n-u)\; \prod_{i=1}^{k}\binom{l}{t_i}}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^k}.$$

That proves the Lemma. □

From (10), as $\mathbf{E}(A)$ only depends on u, and $\mathbf{Pr}(A)$ only depends on $u, t_1, \ldots, t_k$ we get

$$\mathbf{E}|W_r(k)| = \sum_{u=0}^{n} \binom{n}{u} q^u \, \mathbf{E}(A) \sum_{t_1,\ldots,t_k} \prod_{i=1}^{k} \left(1 - (1 - \frac{1}{q})^{q^{t_i}}\right) \, \mathbf{Pr}(A). \tag{12}$$

From (12) by Lemma 12,

$$\begin{aligned} \mathbf{E}|W_r(k)| \le \; & \frac{1}{\prod_{i=1}^{l-1}(1 - \frac{i}{n})^k} \sum_{u=0}^{n} \binom{n}{u} q^u \, \mathbf{E}(A) \\ \times \sum_{T=0}^{L} C_T \; & \left(\frac{u}{n}\right)^{L-T} \left(\frac{n-u}{n}\right)^{T} P_1(L-T, u) \; P_2(T, n-u), \end{aligned} \tag{13}$$

where

$$C_T = \sum_{t_1+\ldots+t_k=T} \; \prod_{i=1}^{k} \binom{l}{t_i} \left(1 - (1 - \frac{1}{q})^{q^{t_i}}\right).$$

Let

$$f(z) = \sum_{t=0}^{l} \binom{l}{t} \left(1 - (1 - \frac{1}{q})^{q^t}\right) z^t.$$

It is obvious that $f^k(z) = \sum_{T=0}^{lk} C_T \, z^T$.

Lemma 13. *For every $0 \le T \le l\,k$ we have $C_T \le \frac{f^k(z)}{z^T}$, where z is the only nonnegative root(including ∞ for $T = lk$) to the equation $k\frac{zf'(z)}{f(z)} = T$.*

Proof. It is similar to the proofs of Lemmas 8 and 9. □

Let $T = \gamma n$, where $0 \le \gamma \le \alpha$. By Lemma 8, $P_1(L-T, u) \le \frac{g^u(x)\,(L-T)!}{x^{L-T}\,u^{L-T}}$, where x is a nonnegative root of the equation $\beta\frac{xg'(x)}{g(x)} = \alpha - \gamma$. Therefore, by estimating $(L-T)!$ with Lemma 10, we get

$$P_1(L-T, u) \le \left[\frac{g^\beta(x)}{x^{\alpha-\gamma}} \left(\frac{\alpha-\gamma}{\beta e}\right)^{\alpha-\gamma} + \epsilon\right]^n, \tag{14}$$

for any positive ϵ and big enough n. By Lemma 9, $P_2(T, n-u) \le \frac{h^{n-u}(y)\,T!}{y^T\,(n-u)^T}$. Therefore,

$$P_2(T, n-u) \le \left[\frac{h^{1-\beta}(y)}{y^\gamma} \left(\frac{\gamma}{(1-\beta)e}\right)^{\gamma} + \epsilon\right]^n, \tag{15}$$

for any positive ϵ and all big n, where y is a nonnegative root to $(1-\beta)\frac{yh'(y)}{h(y)} = \gamma$. By Lemma 13,

$$C_T \le \left(\frac{f^{\frac{\alpha}{l}}(z)}{z^\gamma}\right)^n, \tag{16}$$

where z is a nonnegative root to $\frac{\alpha}{l}\frac{zf'(z)}{f(z)} = \gamma$. We remark that for any positive ϵ bounds (11), (14), (15) are true simultaneously for any α, β, γ and big enough n. Therefore, taking all these bounds into account, from (13) we get

$$\mathbf{E}|W_r(k)| \le \frac{(n+1)(lm+1)}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^m} \max\left[\max\left(\frac{q^\beta\, g^\beta(x)\, h^{1-\beta}(y)\, f^{\frac{\alpha}{l}}(z)\, (\alpha-\gamma)^{\alpha-\gamma}\, \gamma^\gamma}{\beta^\beta\, (1-\beta)^{1-\beta}\, x^{\alpha-\gamma}\, y^\gamma\, z^\gamma\, e^\alpha}\, F(\beta)^{d-\frac{\alpha}{l}}\right) + \epsilon\right]^n,$$

for any positive ϵ and big enough n, where Lemma 10 was used to bound the binomial coefficient $\binom{n}{u}$. Therefore,

$$\mathbf{E}|W_r(k)| \le (\max G(\alpha,\beta,\gamma) + \epsilon)^n \tag{17}$$

for any positive ϵ and big enough n, where

$$G(\alpha,\beta,\gamma) = \frac{q^\beta\, g^\beta(x)\, h^{1-\beta}(y)\, f^{\frac{\alpha}{l}}(z)\, (\alpha-\gamma)^{\alpha-\gamma}\, \gamma^\gamma}{\beta^\beta\, (1-\beta)^{1-\beta}\, x^{\alpha-\gamma}\, y^\gamma\, z^\gamma\, e^\alpha}\, F(\beta)^{d-\frac{\alpha}{l}}.$$

The maximum in (17) is over $0 \le \beta \le 1$ and $0 \le \gamma \le \alpha$. We remark that the parameters α, β, γ should satisfy $r\beta \le \alpha - \gamma$ and $(r-1)(1-\beta) \le \gamma$, otherwise $P_1(L-T,u) = 0$ or $P_2(T, n-u) = 0$. The first stage complexity is upper bounded by (17) with the maximum over above α, β, γ, where nonnegative x, y, z satisfy

$$\beta\frac{xg'(x)}{g(x)} = \alpha - \gamma, \tag{18}$$

$$(1-\beta)\frac{yh'(y)}{h(y)} = \gamma, \tag{19}$$

$$\frac{\alpha}{l}\frac{zf'(z)}{f(z)} = \gamma. \tag{20}$$

However, the function $G(\alpha,\beta,\gamma)$ may have some singularities. For instance, at $\beta = 0$, we should have $\alpha - \gamma = 0$ and every $0 \le x \le \infty$ is the solution to (18). The similar is true at $\beta = 1$ with (19) and at $\alpha = 0$ with (20). So one may then take a small $\epsilon_1 > 0$ and consider the extrema of $G(\alpha,\beta,\gamma)$ in the area $\epsilon_1 \le \beta \le 1-\epsilon_1$, $\epsilon_1 \le \alpha \le dl$, where the function is well-defined and continuous. Out of this area the contribution of the right hand side terms in (13) is negligible. The maximum is unique and it is computed with an advanced optimization package like MAPLE.

11 Complexity Estimate. Stage 2

We recall that if $r \le 2$, then nothing to do. Let $r \ge 3$. Let $W_r = W_r(m)$ and $Z_r = Z_r(m)$. Let $X_1, \dots, X_m$ be fixed and $f_1, \dots, f_m$ randomly generated. It was proved in Section 7 that the expected complexity of the second stage is $\mathbf{E}(q^{n-m} + |W_r|)c^{n-|Z_r|}$, where c is defined in Theorem 1. As in (3), we prove

$$\mathbf{E}\left(|W_r|c^{n-|Z_r|}\right) = \mathbf{E}_{X_1,\dots,X_m}\left(q^{|Z_r|}c^{n-|Z_r|}\prod_{i=1}^{m}\left(1-(1-\frac{1}{q})^{q^{|X_i\setminus Z_r|}}\right)\right).$$

Let $L = lm$. Similarly to (13),

$$\mathbf{E}\left(|W_r|\ c^{n-|Z_r|}\right) \le \frac{1}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^m} \sum_{u=0}^{n} \binom{n}{u} q^u c^{n-u}$$
$$\times \sum_{T=0}^{L} C_T \left(\frac{u}{n}\right)^{L-T} \left(\frac{n-u}{n}\right)^{T} P_1(L-T,u)\ P_2(T,n-u),$$

where $C_T = \sum_{t_1+\ldots+t_m=T} \prod_{i=1}^{m} \binom{l}{t_i} \left(1-(1-\frac{1}{q})^{q^{t_i}}\right)$. Therefore,

$$\mathbf{E}\left(|W_r|\ c^{n-|Z_r|}\right) \le \left[\max c^{1-\beta} G(dl,\beta,\gamma) + \epsilon\right]^n, \tag{21}$$

for any positive ϵ and big enough n. The maximum is over $0 \le \beta \le 1$ and $0 \le \gamma \le dl$, where nonnegative x, y, z satisfy (18),(19),(20) and $\alpha = dl$. Similarly to (13),

$$\mathbf{E}\left(q^{n-m} c^{n-|Z_r|}\right) \le \frac{1}{\prod_{i=1}^{l-1}(1-\frac{i}{n})^m} \sum_{u=0}^{n} \binom{n}{u} q^{n-m} c^{n-u}$$
$$\times \sum_{T=0}^{L} C'_T \left(\frac{u}{n}\right)^{L-T} \left(\frac{n-u}{n}\right)^{T} P_1(L-T,u)\ P_2(T,n-u),$$

where $C'_T = \sum_{t_1+\ldots+t_m=T} \prod_{i=1}^{m} \binom{l}{t_i} = \binom{lm}{T}$. Therefore,

$$\mathbf{E}\left(q^{n-m} c^{n-|Z_r|}\right) \le \left[\max\left(\frac{q^{1-d} c^{1-\beta}\ g^{\beta}(x)\ h^{1-\beta}(y)\ (dl)^{dl}}{\beta^{\beta}\ (1-\beta)^{1-\beta}\ x^{dl-\gamma}\ y^{\gamma}\ e^{dl}}\right) + \epsilon\right]^n \tag{22}$$

for any positive ϵ and big enough n. The maximum is over $0 \le \beta \le 1$ and $0 \le \gamma \le dl$, where nonnegative x, y satisfy (18),(19). Computations with MAPLE, similar to those in the previous Section, shows that $\mathbf{E}|W_r|\ c^{n-|Z_r|}$ dominates the complexity of the second stage and the overall algorithm complexity is dominated by the first stage at least for the parameters in Tables 1 and 2.

12 Trivially Unsolvable Equations

Trivially unsolvable equations are often generated according to the Section 1.3 model. However that phenomenon only negligibly contributes to the average complexity bounds if they are exponential.

The probability that a randomly chosen equation in l variables is solvable over F_q, i.e., admits at least one solution over F_q, is $1-(1-\frac{1}{q})^{q^l}$. So the probability the equation system (1) is trivially unsolvable(at least one of the equations has no solutions over F_q) is $1-\left[1-(1-\frac{1}{q})^{q^l}\right]^m$. This value tends to 1 as l and q are fixed and $m = dn$ tends to infinity. It is very easy to recognize, with some average complexity R, a trivially unsolvable equation system. However, for small d that only gives a negligible contribution to the average complexity estimate while it is exponential. Really, let Q denote average complexity of a deterministic algorithm on all instances of (1). Let Q_1 denote average complexity of the algorithm on the instances of (1) which are not trivially unsolvable, i.e.,

each equation has at least one solution over F_q. In both cases uniform distribution is assumed. By the conditional expectation formula,

$$Q = \left[1 - (1 - \frac{1}{q})^{q^l}\right]^{dn} Q_1 + \left(1 - \left[1 - (1 - \frac{1}{q})^{q^l}\right]^{dn}\right) R.$$

Therefore, $Q_1 < \left[1 - (1 - \frac{1}{q})^{q^l}\right]^{-dn} Q$. For $q = 2$ and $d = 1$ that will only change the bound at $l = 3$. For the Weak IAG Algorithm, Q_1 becomes bounded by 1.033^n while Q is bounded by 1.029^n for large n. For all other l the influence is negligible: estimates for Q and Q_1 are almost identical. For larger $d = 1 + \delta$ the contribution is larger, but Q becomes sub-exponential fast, that follows from the analysis in Sections 10 and 11. So Q_1 remains bounded by a very low exponential function at least for low δ. In fact, we believe that Q_1 becomes sub-exponential too, though it is not proved here.

Generally, a subsystem of some $t \geq 2$ equations may be inconsistent(without solutions). Then the whole system is inconsistent too. At least for low t the case may be identified by trying $\binom{m}{t}$ possible t-subsystems and therefore in polynomial time providing m does not grow very fast. It seems difficult to compute the probability of the event and give its asymptotical analysis. For $t = 2$ some heuristic argument shows that slightly affects algorithm's average running time only for $q = 2$ and very low l as it is exponential. That is despite the probability presumably tends to 1 as for $t = 1$. Anyway, the algorithms studied here equally handle this and more complicated cases.

13 Average Time Complexity Conjecture

A drastic improvement over last few years in average time complexity of solving (1) raises the question about the function type that may represent it. Exponential function in n representing the run-time of an Algorithm is, by definition, $(1 + \epsilon)^n$, where ϵ is a positive constant. For any sub-exponential function we have positive $\epsilon = \epsilon(n) \to 0$ as n tends to infinity. We remark

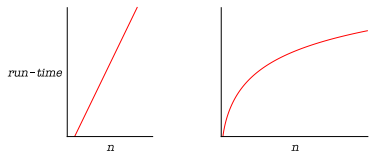

Fig. 3. Typical exponential and sub-exponential functions in log-scale

1. for $q = 2$ and very low l as $3, 4$ the estimates presented in Table 1 are as $(1 + \epsilon)^n$, where ϵ has tendency to diminish to 0,
2. generally, as l is bounded and n grows, the same type function(exponential or sub-exponential) likely represents the problem complexity for low and larger l,
3. experiments on random 5-sparse Boolean equations in $n = m \leq 220$ variables, Fig. 9 in [25], show the average running time of MiniSat obeys rather a sub-exponential law (on the right in Fig. 3 here) than an exponential (on the left).

Therefore, the following statement called *Average Time Complexity Conjecture* might be true(it was already formulated in [22]).

(1) *There exists an algorithm whose expected time complexity on uniformly random instances is sub-exponential in n as q and l are fixed, $m \geq n$ while n tends to infinity.*

Symmetric ciphers security is based on the assumption of exponential complexity. A cipher is commonly considered broken if there is an attack whose running time is less than the full search of the key-space, no matter how small the gain is. That differs much from the asymmetric case, where there are effective methods of sub-exponential complexity for integer factoring and discrete logarithms in finite fields. In elliptic curve crypto the underlying problem is exponential, though only half of the key-space in logarithmic measure is to be searched. The similar is true for lattice based crypto-systems.

We believe that sparse equation systems over finite fields are of fundamental importance in cryptanalysis as they provide a tool to write computational problems from either symmetric or asymmetric ciphers in one way. From the point of view of the above conjecture and precedent discussion, it would not be a big surprise if those problems are in nature sub-exponential. Remark that does not contradict with the problem of solving sparse polynomial equations over finite fields is still NP-hard.

Finding the conjectural algorithm(that may be already a Sat-solver like MiniSat, but we do not have any proof of that) might imply a series of improvements from the crypto communities as it was with the index calculus for discrete logs and factoring. Therefore, if proved the conjecture may have far-reaching consequences in the field of cryptanalysis, as changing the symmetric ciphers security assumption or may be breaking some of them, and in computing in general. Its publication may stimulate research in the field.

Previously, for quadratic semi-regular Boolean equation systems it was shown the Gröbner basis algorithm is of sub-exponential complexity provided $n = o(m)$; see [2]. The present conjecture claims that is true for average sparse equation systems regardless their regularity and algebraic degree, and for any $m \geq n$.

References

1. C. Bouillaguet, H.-C. K. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir, and B.-Y. Yang,*Fast exhaustive search for polynomial systems in F_2*, IACR ePrint Archive, report 2010/313.
2. M. Bardet, J.-C.Faugére, and B. Salvy, *Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F_2 with solutions in F_2,* Research report RR–5049, INRIA, 2003.
3. M. Bardet, J-C. Faugére, B. Salvy and B-Y. Yang, *Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems*, in MEGA 2005, 15 pages.
4. B. Buchberger, *Theoretical Basis for the Reduction of Polynomials to Canonical Forms,* SIGSAM Bull. 39(1976), 19-24.
5. E.T. Copson, *Asymptotic expansions,* Cambridge University Press, 1965.
6. R. Courant,*Differential and integral calculus, vol. 1,* Interscience publishers, 1988.
7. N. T. Courtois and G. V. Bard, *Algebraic Cryptanalysis of the Data Encryption Standard*, in Cryptogr. and Coding, LNCS 4887, pp. 152-169, Springer-Verlag, 2007.
8. E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. M. Kleinberg, C. H. Papadimitriou, P. Raghavan, U. Schning,*A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search.* Theor. Comput. Sci. 289(2002), pp.69–83.
9. N. Eén, N. Sörensson, MiniSat home page, http://minisat.se/

10. J.-C. Faugère, *A new efficient algorithm for computing Grbner bases (F4),* Journal of Pure and Applied Algebra, vol. 139 (1999), pp. 61-88.
11. J.-C. Faugère, *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),* in ISSAC 2002, pp. 75 – 83, ACM Press, 2002.
12. K. Iwama, *Worst-Case Upper Bounds for kSAT,* The Bulletin of the EATCS, vol. 82(2004), pp. 61–71.
13. V. Kolchin, A. Sevast'yanov, and V. Chistyakov, *Random allocations,* John Wiley & Sons, 1978.
14. D. Lazard, *Gröbner-bases, Gaussian elimination and resolution of systems of algebraic equations*, in EUROCAL 1983, pp. 146–156.
15. MAPLE home page, http://www.maplesoft.com
16. C. H. Papadimitriou. *On selecting a satisfying truth assignment,* In Proc. FOCS'91, pages 163–169, 1991.
17. H. Raddum, *Solving non-linear sparse equation systems over $GF(2)$ using graphs,* University of Bergen, preprint, 2004.
18. H. Raddum, I. Semaev, *Solving Multiple Right Hand Sides linear equations*, Des. Codes Cryptogr., vol.49 (2008), pp.147–160.
19. U. Schöning, *A probabilistic algorithm for k-Sat based on limited local search and restart,* Algoritmica, 32(2002), 615-623
20. I. Semaev, *On solving sparse algebraic equations over finite fields*, Des. Codes Cryptogr., vol. 49 (2008), pp.47–60.
21. I. Semaev, *Sparse algebraic equations over finite fields,* SIAM J. on Comp., vol. 39(2009), pp. 388–409.
22. I. Semaev and M. Mikus, *Methods to solve algebraic equations in cryptanalysis,* Tatra Mountains Math. Publ., **45**(2010), 107-136.
23. I. Semaev, *Improved Agreeing-Gluing Algorithm*, 2nd Int. Conf. on Symb. Comp. and Crypt., Royal Holloway, University of London, UK, June 23-25, 2010,(2010), 73-88.
24. I. Semaev, *Sparse Boolean equations and circuit lattices*, Des. Codes Cryptogr., vol. 59 (2011), pp. 349–364.
25. T.E.Schilling and H.Raddum *Solving Equation Systems by Agreeing and Learning*, in WAIFI 2010, LNCS 6087, pp. 151-165,Springer-Verlag, 2010.
26. D. H. Wiedemann, *Solving sparse linear equations over finite fields,* IEEE Trans. Information Theory, vol. 32(1986), pp. 54–62.
27. B.-Y. Yang, J-M. Chen, and N.Courtois, *On asymptotic security estimates in XL and Gröbner bases-related algebraic cryptanalysis,* LNCS 3269, pp. 401–413, Springer-Verlag, 2004.
28. A. Zakrevskij, I. Vasilkova, *Reducing large systems of Boolean equations,* 4th Int.Workshop on Boolean Problems, Freiberg University, September, 21–22, 2000.