A crossbred algorithm for solving Boolean polynomial systems

Antoine Joux[1] and Vanessa Vitse[2]

[1] Chaire de Cryptologie de la Fondation de l'UPMC
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, Paris, France
[2] Institut Fourier, Université Grenoble-Alpes
antoine.joux@m4x.org vanessa.vitse@univ-grenoble-alpes.fr

Abstract. We consider the problem of solving multivariate systems of Boolean polynomial equations: starting from a system of m polynomials of degree at most d in n variables, we want to find its solutions over $\mathbb{F}_2$. Except for $d = 1$, the problem is known to be NP-hard, and its hardness has been used to create public cryptosystems; this motivates the search for faster algorithms to solve this problem. After reviewing the state of the art, we describe a new algorithm and show that it outperforms previously known methods in a wide range of relevant parameters. In particular, the first named author has been able to solve all the Fukuoka Type I MQ challenges, culminating with the resolution of a system of 148 quadratic equations in 74 variables in less than a day (and with a lot of luck).

Key words: Multivariate polynomial systems, Gröbner basis, XL, multivariate cryptography, algebraic cryptanalysis

1 Introduction

The resolution of systems of polynomial equations is a fundamental mathematical tool with numerous applications. It is well known that solving systems of multivariate equations is NP-hard in general, but it does not preclude from seeking the most efficient algorithms; besides, systems coming from applications are often easier to solve than predicted by the worst-case complexity. In this paper, we mostly focus on random instances which is presumably the hardest case.

Actually, there is a subtlety in the signification of "solving". Usually, it means finding all solutions of a given system, i.e. all tuples $(x_1, \ldots, x_n) \in K^n$ satisfying

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \qquad\vdots \\ f_m(x_1, \ldots, x_n) = 0 \end{cases}$$

where $f_1, \ldots, f_m$ are elements of $K[X_1, \ldots, X_n]$. This is mostly fine if the system has a finite number of solutions, or more precisely is zero-dimensional. *Mostly*

because this approach ignores the solutions that may exist in a field extension or at infinity, and also because the solution set may be too large to be practically listed. In this latter case, or if the solution set has positive dimension, the alternative is to find a practical description of the corresponding algebraic variety, and Gröbner bases usually fill that role. Note that, in many applications, including cryptographic ones, it can be sufficient to find a single solution of a system. We also consider this weaker form of solving.

In this article, we focus on systems of quadratic (i.e. total degree 2) equations. It is the simplest case beyond the polynomially-solvable linear case, but the method we propose can also be applied to systems with an higher degree. Of course, the complexity quickly grows with the degree. Remarkably, there exists a general method to transform a system of arbitrary high degree equations into an equivalent quadratic system. This is done by introducing new variables to encode high degree monomials and new equations relating them. Due to the large number of new variables, combining this approach with the resolution of a quadratic system is usually very unefficient.

More importantly, our work focuses on the Boolean case, i.e. we are looking for solutions in $\mathbb{F}_2^n$ of systems of quadratic polynomials with coefficients in the field with two elements $\mathbb{F}_2$. This is relevant for applications in computer science, in coding theory and cryptography (see for instance [5,14,18]); furthermore, any polynomial system defined over a binary field $\mathbb{F}_{2^d}$ can be translated using Weil descent as a system over $\mathbb{F}_2$. The Boolean case has two important implications:

- Since we are looking for solutions defined over $\mathbb{F}_2$ and not an extension, we can add the field equations $x_i^2 + x_i = 0$ to the system. Equivalently, we can work in the Boolean polynomial ring $B[X_1, \ldots, X_n] = \mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 + X_1, \ldots, X_n^2 + X_n)$, where the equations become simpler since no variable may occur with (individual) degree equal to 2 or more.
- In small finite fields, exhaustive search becomes a viable option. This is obviously true for $\mathbb{F}_2$, but also in a lesser extent for other small finite fields such as $\mathbb{F}_3$ or $\mathbb{F}_5$. Our new algorithm, as most current algorithms for solving Boolean systems, partly relies on exhaustive search.

Despite this, Boolean quadratic systems still capture the NP-hardness of polynomial solving. In fact, because the 3-SAT problem can be reduced to the resolution of such systems [10], the existence of an algorithm with worst-case subexponential complexity would refute the Exponential Time Hypothesis [13], a conjecture in complexity theory, generalizing P$\neq$ NP and widely believed to be true.

For the analysis of our algorithm, we will consider systems of random equations (where the monomial coefficients are chosen independently and uniformly in $\{0, 1\}$). The behaviour of such systems differs according to the relative values of m and n [11]. If $m < n$ (there are more unknowns than equations), the system is underdetermined and admits on average $O(2^{n-m})$ solutions. If $m = n$, the system is determined, and has at least one solution with a probability converging to $1 - 1/e$ has n grows to infinity. If $m > n$ (there are more equations than un-

knowns) the system is overdetermined and has no solution with overwhelming probability.

But in practical applications such as cryptography, the polynomial systems, even when overdetermined, always have at least one solution. For this reason, we also consider random consistent systems, i.e. chosen uniformly from the set of systems of m quadratic Boolean polynomials in n variables with at least one solution in $\mathbb{F}_2^n$. Then when m is larger than n, this forced solution is unique with overwhelming probability.

2 State of the art

2.1 Under- and overdetermined systems

Extremely overdetermined ($m > n(n+1)/2$) or underdetermined ($n > m(m+1)$) random Boolean quadratic systems can be solved in polynomial time. The first case simply requires Gaussian elimination on the equations and can be seen as a particular instance of the general approach presented in Sect. 2.4. The second case was solved by Kipnis, Patarin and Goubin in [14]. At PKC 2012, Thomae and Wolf [19] have generalized the algorithm of Kipnis-Patarin-Goubin to other underdetermined systems, and their complexity interpolates between polynomial for $n > m(m+1)$ and exponential for n close to m.

Beyond these two extremes, the $m = n$ case is essentially the hardest. For $m > n$, the additional information given by the extra equations can simplify the problem. And when $n > m$, it is always possible to specialize $n - m$ variables (i.e. set them to arbitrary values) and get back to the case of as many equations as unknowns; at least, if we only seek a single solution which it usually the case for such underdetermined systems.

2.2 Exhaustive search

Obviously, since there are 2^n possible values to instantiate n variables in $\mathbb{F}_2$, it is possible to evaluate the m polynomials for all values in order to find all solutions. At first glance, this costs $m \cdot 2^n$ evaluations of a degree d polynomial. However, this first estimation is too pessimistic. Optimizing the 2^n evaluations is quite subtle, but Bouillaguet et al. proposed in [3] a faster method that relies on the remark that if we know the evaluation of a polynomial at one point and only change the value of one variable, the evaluation at the new point can be computed faster. Their idea is based on the use of partial derivatives of the polynomials. Combined with the use of Gray codes and other techniques, it allows to find all solutions of a system of m Boolean quadratic equations in n variables in $O(\ln(n)2^n)$ elementary operations. Remarkably, this complexity does not depend of m; but obviously if only one solution is needed the search will finish faster for smaller m since there are more solutions then. This fast exhaustive search algorithm is implemented in the `libFES` library (`http://www.lifl.fr/~bouillag/fes/`) and holds several resolution records.

2.3 A provable method faster than exhaustive search

Recently, Lokshtanov et al. [16] proposed a probabilistic method that outperforms exhaustive search asymptotically. Their idea stems from the following observation: $(x_1, \ldots, x_n) \in \mathbb{F}_2^n$ is a solution of the polynomial system generated by $f_1, \ldots, f_m$ if and only if $y = (x_{k+1}, \ldots, x_n)$ is a solution of the equation

$$\prod_{a \in \mathbb{F}_2^k} (1 - \prod_{i=1}^{m} (1 - f_i(a, y))) = 0.$$

Instead of working with this unwieldly polynomial, they consider its probabilistic counterpart

$$R(y) = \sum_{a \in \mathbb{F}_2^k} t_a \prod_{i=1}^{l} (1 - \sum_{j=1}^{m} s_{aij} f_j(a, y))$$

where s_{aij} and t_a are chosen independently and uniformly in $\mathbb{F}_2$ and $l \leq m$ is a parameter. If y is the last part of a solution, then $R(y)$ is uniformly distributed in $\mathbb{F}_2$, but otherwise $R(y) = 0$ with a probability greater than $(1 - 2^{-l})^{2^k}$. By performing several complete evaluations of R on all its 2^{n-k} input values of y, for varying coefficients s_{aij}, t_a, it is possible to recover with high probability the last part of all the solutions of the system. Overall, the complexity is in $\tilde{O}(2^{0.8765n})$, faster than the brute force approach.

As far as we know, this method as not been implemented and it seems unlikely that it outperforms exhaustive search in the range of systems which can be solved with current computer. However, it is remarkable that it asymptotically beats brute force without relying on any heuristic hypothesis concerning the given system. An unfortunate consequence is that the method cannot take advantage of a large value of m compared to n, since it would necessarily requires some hypothesis of "independence" between the equations. Indeed, if we don't care about independence, it is easy to add extra equations by taking linear combinations of the initial ones. As a final remark, one should note that the algorithm of Lokshtanov et al. makes the assumption that the number of solutions of the system is smaller than $2^{0.8765n}$, since otherwise, it would not be possible to list all of them in the indicated complexity.

2.4 Algebraic methods

Algebraic methods consider systems of polynomial equations by looking at the ideals they generate and try to solve them by finding a good representation of the corresponding ideal. More precisely, let $\mathcal{F} = \{f_1, \ldots, f_m\}$ be a family of elements in a multivariate polynomial ring $K[X_1, \ldots, X_n]$ and form the ideal $I = \langle f_1, \ldots, f_m \rangle$ generated by the family $\mathcal{F}$. By definition, I is the following set of polynomials:

$$I = \left\{ \sum_{i=1}^{m} p_i f_i \mid (p_1, \ldots, p_m) \in K[X_1, \ldots, X_n]^m \right\}.$$

Thus, for any element f of the ideal I, there exist polynomials $p_1, \ldots, p_m$ such that $f = \sum_{i=1}^{m} p_i f_i$; in other words, there exists an integer $D = \max\{\deg p_i : 1 \leq i \leq m\}$ such that f belongs to the vector space

$$V_{\mathcal{F},D} = Span_K \left\{ uf_i \middle| i \in [1;m]; u \text{ a monomial with} \deg u \leq D - \deg f_i \right\}.$$

Macaulay matrices. The above observation implies that relevant information on the ideal I can be obtained by studying these vector spaces and motivates the following definition.

Definition 1 *For any integer k, let T_k be the set of monomials of $K[X_1, \ldots, X_n]$ of degree smaller than or equal to k. The degree D Macaulay matrix of $\mathcal{F}$, denoted by $Mac_D(\mathcal{F})$, is the matrix with coefficients in K whose columns are indexed by T_D, whose lines are indexed by the set $\left\{(u, f_i) \middle| i \in [1;m]; u \in T_{D-\deg(f_i)}\right\}$, and whose coefficients are those of the products uf_i in the basis T_D.*

Macaulay matrices can be thought as multivariate analogs of the classical Sylvester matrix. Lazard first showed in [15] that they can be used to compute Gröbner bases: for any monomial order $\succ$, there exists a degree D such that if the columns of $\mathrm{Mac}_D(\mathcal{F})$ are sorted according to $\succ$, the rows of its reduced echelon form contains the coefficients of a Gröbner basis of I. This idea of expressing many multiples of a family of polynomials in matrix form and reducing the resulting matrices is at the heart of most current algorithms for computing Gröbner bases, such as F4, F5, XL and their many variants [4,7,8].

When K is equal to $\mathbb{F}_2$, we usually want to add the field equations $X_i^2 = X_i$ for all $i \in [1;m]$. As stated before, it is more efficient to work directly in the quotient algebra $B[X_1, \ldots, X_n] = \mathbb{F}_2[X_1, \ldots, X_n]/(X_1^2 + X_1, \ldots, X_n^2 + X_n)$ (B stands for Boolean). The definition can be adapted by requiring that every monomial (either in T_k or in the products uf_i) has degree strictly smaller than 2 in each variable. Of course, we can proceed in a similar way when working over $\mathbb{F}_q$ with q small.

In many situations, the system $f_1 = \cdots = f_m = 0$ is overdetermined and so has none or very few solutions. This implies that the ideal $I = \langle f_1, \ldots, f_m \rangle$ will contain 1 (if there is no solution) or linear polynomials, from which it is easy to deduce the solutions. Again, such low degree equations can be obtained by reducing the Macaulay matrix $\mathrm{Mac}_D(\mathcal{F})$, with its columns sorted by total degree, for some degree D. The smallest such integer D is called the *degree of regularity* of the system and denoted by D_{reg}. (Note that this only one out of many other definitions of D_{reg}.)

With this approach, solving an overdetermined system of Boolean quadratic polynomials amounts to computing the row echelon form of a large matrix, for a total cost in

$$\tilde{O}\left(\binom{n}{D_{reg}}^{\omega}\right),$$

where ω is the exponent of matrix multiplication (smallest known value is $\omega = 2.373$; in practice $\omega = 2.807$ with Strassen algorithm). But this Macaulay matrix

is extremely sparse: by design, it has at most $1+n(n+1)/2$ non zero coefficients per row, which is negligible compared to its number of columns when n goes to infinity (as soon as $D > 2$, of course). This suggests that instead of Gaussian elimination, sparse linear algebra techniques such as Lanczös algorithm [17] or block Wiedemann algorithm [20] could be used. Indeed, it is possible to probabilistically test the consistency of a Boolean quadratic system in $\tilde{O}\left(\binom{n}{D_{reg}}^2\right)$ and to find a (small number of) solution(s) if any exists. It remains an open problem to find all solutions with the same complexity, when there are many.

However, determining the degree of regularity is not straightforward, although a practical option is to reduce several Macaulay matrices in increasing degrees until enough linear polynomials have been found. Asymptotic estimates exist for an important class of systems, called "semi-regular"; heuristic arguments and experimental evidence suggest that random systems fall in this class with overwhelming probability. In this case, Bardet et al. showed in [2] that if $m \sim \alpha n$ ($\alpha \geq 1$ fixed), as n goes to infinity, then $D_{reg} \sim M(\alpha)\, n$ where $M(\alpha)$ is an explicit decreasing function of α. In particular for $\alpha = 1$, with $\omega = 2$ this yields an asymptotic complexity of $\tilde{O}(2^{0.8728n})$, faster than exhaustive search and even than Lokshtanov et al. But this complexity is conditional to the semi-regularity of the system: it is conjectured to hold with probability converging to 1 as n grows, but exceptional systems may be harder to solve with this technique. By contrast, the complexity of the methods of Sect. 2.2 and 2.3 does not rely on any assumption.

In practice, computing the row echelon form of the D_{reg} Macaulay matrix of $f_1, \ldots, f_m$ is too costly to be efficient. In particular, for the Boolean case, it has been estimated (see [3]) that these methods would not outperform exhaustive search for any value of n smaller than 200. Nevertheless, algebraic algorithms have proven themselves to be very efficient on specific systems with extra algebraic properties which imply a low degree of regularity. A striking example is given by systems arising in the Hidden Field Equations cryptosystem [18]. In this case, a consequence of the presence of a hidden backdoor is a degree of regularity smaller than expected [6,12], leading to devastating attacks [9].

The BooleanSolve hybrid algorithm. In order to improve on the above algebraic technique, Bardet, Faugère, Salvy and Spaenlehauer [1] have proposed the BooleanSolve algorithm which combines exhaustive search with linear algebra on a Macaulay matrix. It takes as input the family $\mathcal{F} = \{f_1, \ldots, f_m\} \subset B[X_1, \ldots, X_n]$ of Boolean polynomials, a parameter $k \leq n$ and proceeds as follows:

1. For each $a = (a_{k+1}, \ldots, a_n) \in \mathbb{F}_2^{n-k}$, compute the specialized polynomials $f_{1,a}, \ldots, f_{m,a}$ where $f_{i,a} = f_i(X_1, \ldots, X_k, a_{k+1}, \ldots, a_n) \in B[X_1, \ldots, X_k]$.
2. Using the Macaulay matrix of $f_{1,a}, \ldots, f_{m,a}$ in degree D_{reg}, check if the specialized system $f_{1,a} = \cdots = f_{m,a} = 0$ admits a solution. If no, continue with the next value of $a \in \mathbb{F}_2^{n-k}$; otherwise, find the the solution $(x_1, \ldots, x_k, a_{k+1}, \ldots, a_n)$ using e.g. exhaustive search on $x_1, \ldots, x_k$.

Specializing the equations allows to dramatically reduce the size of the Macaulay matrices, not only because the number of variables diminishes, but also because the degree of regularity decreases as the ratio between the number of equations and the number of variables goes up. Of course, it also entails a factor 2^{n-k} in the complexity, corresponding to the number of times the second step has to be executed.

In this second step, for most values a the specialized system will have no solution, meaning that 1 is in the ideal. As discussed above, it is possible to take advantage of the sparsity of the Macaulay matrix in testing this property. Indeed, the full row echelon form of the matrix is not needed; one just has to test whether a constant polynomial can be found in the Macaulay matrix. This can be done using a probabilistic method based on Lanczös algorithm.

Under the assumption that the specialized systems still behave like random ones — more precisely, that they remain semi-regular (the "strong semi-regularity" of [1]) — it is possible to derive a complexity estimate. In the case $m \sim \alpha n$ ($\alpha \geq 1$ fixed and n going to infinity), for $\alpha < 1.82$ the asymptotically best choice is $k = 0.55\,\alpha n$, for a complexity in $\tilde{O}(2^{(1-0.208\alpha)n})$. In particular, for $\alpha = 1$ this yields a (conditional) complexity of $\tilde{O}(2^{0.792n})$. For $\alpha \geq 1.82$ the asymptotically best choice is $k = n$, i.e. no variables are specialized: the system is too overdetermined for the algorithm, and it does not improve on the standard reduction of the full Macaulay matrix.

3 Our crossbred algorithm

3.1 General principle

In the BooleanSolve algorithm, the most costly step is the linear algebra of the Macaulay matrix, which is done 2^{n-k} times. In order to avoid this problem, we propose a new method that performs the specialization step on $n-k$ variables *after* working with the Macaulay matrix .

Basic idea. A first idea is to construct a degree D Macaulay matrix, sort its columns in lexicographical order, then compute the last rows of its row echelon form. This allows to generate degree D equations in which k variables have been eliminated, and this resulting system can then be solved using exhaustive search on $n-k$ variables. As a toy example, we can consider the following system:

$$\begin{cases} X_1X_3 + X_2X_4 + X_1 + X_3 + X_4 = 0 \\ X_2X_3 + X_1X_4 + X_3X_4 + X_1 + X_2 + X_4 = 0 \\ X_2X_4 + X_3X_4 + X_1 + X_3 + 1 = 0 \\ X_1X_2 + X_1X_3 + X_2X_3 + X_3 + X_4 + 1 = 0 \\ X_1X_2 + X_2X_3 + X_1X_4 + X_3 = 0 \\ X_1X_3 + X_1X_4 + X_3X_4 + X_1 + X_2 + X_3 + X_4 = 0 \end{cases}$$

The corresponding degree 2 Macaulay matrix, in lex order, is

$$\begin{array}{c} \begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_1X_4 & X_1 & X_2X_3 & X_2X_4 & X_2 & X_3X_4 & X_3 & X_4 & 1 \end{array} \\ \left(\begin{array}{ccccccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array} \right) \end{array}$$

and its reduced row echelon form is

$$\begin{array}{c} \begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_1X_4 & X_1 & X_2X_3 & X_2X_4 & X_2 & X_3X_4 & X_3 & X_4 & 1 \end{array} \\ \left(\begin{array}{ccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right) \end{array}$$

We obtain two equations not involving X_1, namely $X_2X_3 + X_3X_4 + X_3 + X_4 + 1 = 0$ and $X_2X_4 + X_2 = 0$, which can be solved for instance with exhaustive search; the solutions thus found must then be checked for compatibility with the remaining equations in X_1.

An obvious drawback of this method is that in order to eliminate a significant number of variables, the degree D should be taken large enough, and reducing large Macaulay matrices is quickly prohibitive.

A more refined variant. An important remark is that it is not necessary to completely eliminate k variables. We now illustrate this with the same example. First, we sort the columns, this time according to the graded reverse lexicographic order (grevlex), and obtain the following row echelon form:

$$\begin{array}{c} \begin{array}{ccccccccccc} X_1X_2 & X_1X_3 & X_2X_3 & X_1X_4 & X_2X_4 & X_3X_4 & X_1 & X_2 & X_3 & X_4 & 1 \end{array} \\ \left(\begin{array}{ccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{array} \right) \end{array}$$

The last three equations have degree 1 in X_1, X_2, X_3:

$$\begin{cases} (X_4 + 1)X_1 + X_2 + X_3 + 1 = 0 \\ (X_4 + 1)X_2 = 0 \\ X_1 + X_2 + (X_4 + 1)X_3 + 1 = 0 \end{cases}$$

Consequently, for any assignation of the last variable, we obtain a system that can be easily solved using linear algebra. Reducing the same Macaulay matrix, we have thus "eliminated" three variables from the exhaustive search procedure. This is somewhat reminiscent of Kipnis-Goubin-Patarin algorithm [14] for solving extremely underdetermined quadratic systems, whose idea is also to generate enough equations of the form

$$P_1(X_{k+1}, \ldots, X_n)X_1 + \cdots + P_k(X_{k+1}, \ldots, X_n)X_k + Q(X_{k+1}, \ldots, X_n) = 0,$$

yielding a linear system once the variables $X_{k+1}, \ldots, X_n$ are specialized.

3.2 Description of the algorithm

Our algorithm implements this idea in a scalable way. It depends on three parameters D, d and k, with $D \geq 2$, $1 \leq d < D$ and $1 \leq k \leq n$. To simplify the description, for any polynomial $p \in B[X_1, \ldots, X_n]$, we let $\deg_k p$ stand for the total degree in $X_1, \ldots, X_k$.

When $d = 1$, it works as proposed above: from the degree D Macaulay matrix (sorted by decreasing value of $\deg_k$), we generate new equations that are linear in $X_1, \ldots, X_k$, i.e. we eliminate all monomials of degree larger than 1 in these variables. This can be achieved be computing elements in the kernel of the truncated matrix, from which the monomials containing at most one of the variables $X_1, \ldots, X_k$ have been removed. The choice of D is a critical parameter, it must be large enough for reduced equations to exist and as small as possible if we want the dimension of the Macaulay matrix to remain manageable. Note that, since the new equations become linear after performing the evaluation of variables X_{k+1} to X_n, it is sufficient to have a little more than k equations of this form.

To extend this to larger values of d, we want to contruct new equations of degree at most d in the first variables $X_1, \ldots, X_k$. For large systems, this allows to select smaller values of D and to work with smaller Macaulay matrices. However, the number of equations that we need in this context to solve the system after specialization through linear algebra becomes larger. Interestingly, when d is equal to or larger than the degree of the initial equations, these initials equations can be included in the pool of equations that we are keeping for specialization.

The main difficulty of this method is to analyze the optimal choices of parameters D, d and k for given values of the number of variables n, the number of equations m and the degree of these equations (2 if we restrict ourselves to quadratic systems).

We give below a pseudo-code description of the algorithm. The algorithm considers the two following submatrices of the full degree D Macaulay matrix:

- $\mathrm{Mac}^{(k)}_{D,d}(\mathcal{F})$ is the submatrix of $\mathrm{Mac}_D(\mathcal{F})$ whose rows correspond to products uf_i with $\deg_k u \geq d - 1$
- $M^{(k)}_{D,d}(\mathcal{F})$ is the submatrix of $\mathrm{Mac}^{(k)}_{D,d}(\mathcal{F})$ whose columns correspond to monomials M with $\deg_k \mathrm{M} > d$.

Basically, the algorithm works as follows:

1. Search elements $v_1, \ldots, v_r$ in the kernel of $M_{D,d}^{(k)}(\mathcal{F})$
2. Compute the polynomials p_i corresponding to $v_i.\mathrm{Mac}_{D,d}^{(k)}(\mathcal{F})$; they have total degree at most D, and at most d in $X_1, \ldots, X_k$.
3. For all $a = (a_{k+1}, \ldots, a_n) \in \mathbb{F}_2^{n-k}$:
   (a) Create the degree d Macaulay matrix $\mathrm{Mac}_d(\mathcal{F}^*)$ corresponding to the polynomials in $\mathcal{F}$ (partially) evaluated at a
   (b) Evaluate the polynomials p_i at a and append them to $\mathrm{Mac}_d(\mathcal{F}^*)$
   (c) Check if the resulting system (of degree d) if solvable in $X_1, \ldots, X_k$.

As a further refinement, it is possible to add an outer layer of hybridation. Indeed, we can start by iterating through the possible values of the h last variables $X_{n-h+1}, \ldots, X_n$, and apply the above algorithm 2^h times to the specialized systems of m quadratic equations in $n-h$ variables. The main interest of this outer hybridation is to allow an easy parallelization between 2^h computers and sometimes to offer a slightly better choice of parameters (see Section 3.3). However, in some sense, it goes against the philosophy of the algorithm and we do not expect this parameter to be asymptotically useful.

3.3 Finding valid parameters for the algorithm

The parameters D, d and k (and h when outer hybridation is used) control the course of the algorithm, but finding optimal (or even functional) values is far from obvious. As a first remark, since we want to find new relations of degree at most d in the first k variables, cancellations of the highest degree parts in $X_1, \ldots, X_k$ must occur. Thus under a strong semi-regularity assumption, we obtain that the parameter D must be greater than or equal to the degree of regularity of a semi-regular system of m equations in k variables.

In addition to that, we need (under a regularity assumption) to compute the number of equations that can be obtained for the final linear system and check that it is at least[3] equal to the number of monomials in the first k variables of degree at most d. We now explain how this is done in the case where $d = 1$ and $D = 3, 4$ that covers all of the reported experiments.

With $d = 1$, the matrix $\mathrm{Mac}_d(\mathcal{F}^*)$ is empty and the linear algebra is simply performed on the evaluated linear polynomials $(p_1^*, \ldots, p_r^*)$ in k variables. Thus it suffices to check that $r \geq k + 1$. As a consequence, we need enough linearly independent elements in the kernel of $M_{D,1}^{(k)}(\mathcal{F})$ which are not in the kernel of $\mathrm{Mac}_{D,1}^{(k)}(\mathcal{F}) = \mathrm{Mac}_D(\mathcal{F})$ (otherwise we get the trivial equation $0 = 0$). A lower bound on that number is simply given by the rank $R_{D,1}$ of $\mathrm{Mac}_D(\mathcal{F})$ minus the number of columns of $M_{D,1}^{(k)}(\mathcal{F})$.

[3] Having a bit more equations is even better, since this leads to a smaller number of consistent systems of evaluation that lead to a finally incorrect solution.

Algorithm 1 The crossbred algorithm

procedure SYSTEM RESOLUTION($\mathcal{F} = (f_1, \ldots, f_m)$)
▷ *System of m equations in n variables. Parameters D, d and k.*
Construct $\mathrm{Mac}^{(k)}_{D,d}(\mathcal{F})$ and $M^{(k)}_{D,d}(\mathcal{F})$
Find r linearly independent elements $(v_1, \ldots, v_r)$ in the (left) kernel of $M^{(k)}_{D,d}(\mathcal{F})$.
▷ *Using (sparse) linear algebra.*
For all $i \in [1; r]$ compute the polynomial p_i corresponding to $v_i.\mathrm{Mac}^{(k)}_{D,d}(\mathcal{F})$.
▷ *Polynomials of total degree at most D and degree at most d in $(X_1, \ldots, X_k)$.*
Perform fast evaluation on $(f_1, \ldots, f_m, p_1, \ldots, p_r)$, n, k with
Callback procedure
▷ *Get $(f_1^*, ..., f_m^*, p_1^*, ..., p_r^*)$ evaluated at each $(x_{k+1}, ..., x_n) \in \{0,1\}^{n-k}$*
Construct the Macaulay matrix $\mathrm{Mac}_d(\mathcal{F}^*)$ of degree d from $(f_1^*, \ldots, f_m^*)$
Append $(p_1^*, \ldots, p_r^*)$ to $\mathrm{Mac}_d(\mathcal{F}^*)$
Use (dense) linear algebra to test the consistency of resulting system,
▷ *As in XL every monomial is viewed as an independent variable.*
if System is consistent **then**
Extract values of $(X_1, \ldots, X_k)$ and test the candidate solution.
Print any valid solution.
end if
end callback
end procedure

Algorithm 2 Fast Evaluation of a polynomial (over $\mathbb{F}_2$)

procedure FAST EVALUATION($(P_1, \ldots, P_R)$, ℓ, k, Callback action)
▷ *Polynomials of degree D in ℓ variables.*
if $\ell = k$ **then**
Perform Callback action on $(P_1, \ldots, P_R)$ and $(x_{k+1}, \ldots, x_n)$
else
Write each P_i as $P_i^{(0)} + X_\ell \cdot P_i^{(1)}$
Let $x_\ell \leftarrow 0$
Fast evaluate on $(P_1^{(0)}, \ldots, P_R^{(0)})$, $\ell - 1$, k and Callback action.
Let $x_\ell \leftarrow 1$
Fast evaluate on $(P_1^{(0)} + P_1^{(1)}, \ldots, P_R^{(0)} + P_R^{(1)})$, $\ell - 1$, k and Callback action.
end if
end procedure

The number $N_{D,d}^{(k)}$ of columns of $M_{D,d}^{(k)}(\mathcal{F})$ corresponds to the number of monomials labeling its columns and is given by the formula:

$$N_{D,d}^{(k)} = \sum_{d_k=d+1}^{D} \sum_{d'=0}^{D-d_k} \binom{k}{d_k} \binom{n-k}{d'}.$$

The number of independent rows of $\mathrm{Mac}_{D,1}^{(k)}(\mathcal{F}) = \mathrm{Mac}_D(\mathcal{F})$ is simple to evaluate when $D = 3$. In that case and in our range of values of (m, n), under the regularity assumption the matrix has full rank. Since every polynomial in $\mathcal{F}$ is multiplied by the monomials 1, x_1, x_2, $\ldots x_n$, there are $R_{3,1} = (n+1) \cdot m$ rows.

For $D = 4$, it is slightly more complicated, because we need to account for the trivial relations of the form $f_i f_j + f_j f_i = 0$ and $(f_i + 1) f_i = 0$. As a consequence, the Macaulay matrix $\mathrm{Mac}_D(\mathcal{F})$ has $R_{4,1} = (1+n+n\,(n-1)/2) \cdot m - m\,(m+1)/2$ independent rows.

Table 1 illustrates this on a few parameters extracted from Section 4.

n_0	m	h	$n = n_0 - h$	k	D	$N_{D,1}^{(k)}$	$R_{D,1}$	Exp. num of polys
35	35	0	35	9	3	1056	1260	204
35	70	0	35	14	3	2366	2520	154
41	41	0	41	11	4	31075	34481	3406
41	82	0	41	15	3	3290	3444	154
74	148	12	62	23	4	277288	278166	878

Table 1. Examples of parameters' computation

The $d > 1$ case, which is relevant asymptotically, is much more difficult. Its complete analysis is an ongoing work.

4 Experiments and timings

4.1 Fukuoka Type I MQ Challenge

In order to experimentally test this new algorithm, we decided to tackle the Fukuoka MQ Challenges [21]. These challenges, available on `https://www.mqchallenge.org/`, were issued in 2015 with the explicit goal to help assess the hardness of solving systems of quadratic equations. The Type I challenges consist of $2n$ Boolean quadratic equations in n variables, and the designers have ensured that every system has some forced solution. At the time we started, the record on $n = 66$ was held by Chou, Niederhagen and Yang, using a fast Gray code enumeration technique on an array of FPGA. It took a little less than 8 days using 128 Spartan 6 FPGAs. It is interesting to note that this allows for a much faster resolution than on CPU based computation. According to our estimations, using libFES the same computation would have taken about 61 000 cpu · days

using Intel Core i7 processors at 2.8 GHz . Note that as mentioned before, the BooleanSolve algorithm on such systems performs no exhaustive search and boils down to working with the full Macaulay matrix, for an asymptotic complexity in $\tilde{O}(2^{0.585n})$.

We found the solution of all the remaining challenges $n = 67 \ldots 74$ by running our code on a heterogenous network of computers at the LIP6 laboratory. Due to this heterogeneity, the timings are not very precise and the running of identical jobs greatly varied depending on the individual machine it runned on. The processors types in the cluster ranged from Opteron 2384 at 2.8 GHz to Xeon 2690 at 2.6 GHz (the latter being about four times faster). Timings[4] are given in Table 2.

Number of Vars ($m = 2n$)	External hybridation h	Parameters $(D, n-h-k)$	Max CPU (estimate)	Real CPU (rounded)
67	9	$(4, 36)$	6 200 h	3 100 h
68	9	$(4, 37)$	11 200 h	4 200 h
69	9	$(4, 38)$	15 400 h	15 400 h
70	13	$(4, 34)$	33 000 h	16 400 h
71	13	$(4, 35)$	60 000 h	13 200 h
72	13	$(4, 36)$	110 000 h	71 800 h
73	13	$(4, 37)$	190 000 h	14 300 h
74	12	$(4, 39)$	360 000 h	8 100 h

Table 2. Fukuoka challenge

4.2 Crossover point compared to fast enumeration when $m = n$

In addition to the above records, which take advantage of having twice as many equations as variables, it is interesting to compare the relative performances of our algorithm and fast enumeration when $m = n$. We ran experiments on Intel Core i7 laptop at 2.8 GHz for values of n ranging from 35 to 46. For the fast enumeration, we used the state of the art library libFES. The results are summarized in Table 3 and make clear that the cross-over point is at $n = 37$ when $m = n$. The table also contains timings of our code when $m = 2n$, in order to illustrate the gain in terms of running time when extra equations are available.

5 Conclusion

In this article, we have presented a new "crossbred" algorithm for solving systems of Boolean equations, using both exhaustive search and the ideal-based approach

[4] As remarked in the abstract, the last two entries in this table correspond to extremely lucky running times. The desired solution just happened to be found by the first series of parallel jobs.

Number of Vars	libFES	Our code $(m = n)$	Parameters $(D, n-k)$	Our code $(m = 2n)$	Parameters $(D, n-k)$
35	2.3 s	3.8 s	$(3, 26)$	0.6 s	$(3, 21)$
36	4.9 s	7.2 s	$(3, 27)$	0.9 s	$(3, 22)$
37	9.6 s	9.5 s	$(3, 27)$	1.5 s	$(3, 23)$
38	20.1 s	16.5 s	$(3, 28)$	2.5 s	$(3, 24)$
39	40.2 s	33 s	$(3, 29)$	2.7 s	$(3, 24)$
40	84 s	65 s	$(3, 30)$	4.8 s	$(3, 25)$
41	162 s	131 s	$(4, 30)$	9 s	$(3, 26)$
42	317 s	242 s	$(4, 31)$	18 s	$(3, 27)$
43	642 s	437 s	$(4, 32)$	36 s	$(3, 28)$
44	1380 s	850 s	$(4, 33)$	71 s	$(3, 29)$
45	2483 s	989 s	$(4, 33)$	146 s	$(3, 30)$
46	5059 s	1905 s	$(4, 34)$	151 s	$(3, 30)$

Table 3. Comparison with libFES (Timings for full enumeration of search space)

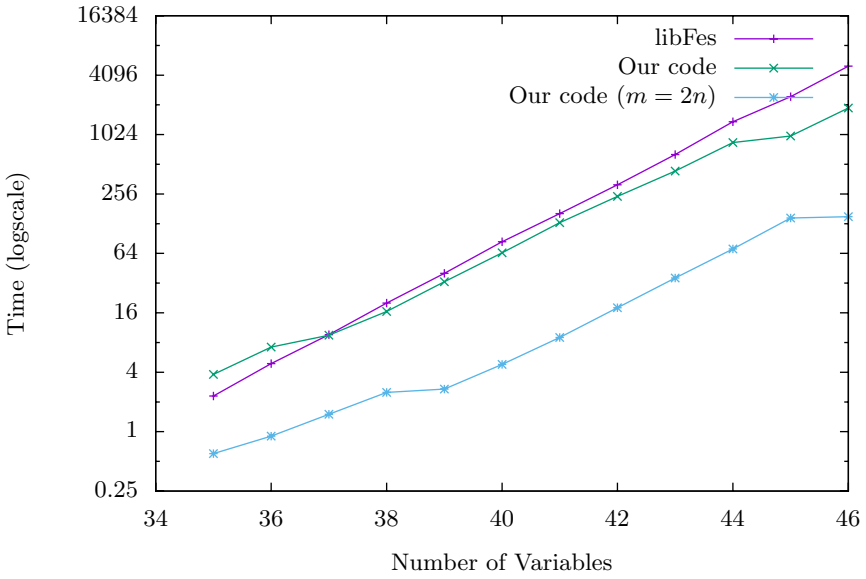

Fig. 1. Comparison with libFES

of Lazard. The main idea of the new algorithm is to reduce a partial Macaulay matrix before a specialization of part of the variables. Since it can be combined with outer hybridation, this leaves more room for optimization and allows to continue to interpolate between exhaustive search and Macaulay reduction even for highly overdetermined systems.

We have demonstrated that our mixed approach decisively beats the fast enumeration technique of [3] for large real-world (over)determined systems of Boolean quadratic polynomials. In particular, we have been able to solve all the Fukuoka Type I MQ Challenges [21] up to the last system of 148 quadratic equations in 74 variables, whereas the previous record using fast enumeration consisted in the resolution of a system of 132 equations in 66 variables. Note that, for such parameters ($m = 2n$) the hybrid BooleanSolve algorithm of [1] is optimal with an empty hybridation and thus becomes equivalent to the classical Lazard method [15].

Moreover as mentioned in [3], pre-existing algebraic methods are not expected to beat brute force for $n = m$ and n lower than 200. Yet, we have demonstrated that in practice the crossover point between exhaustive search and our method is $n = 37$.

We have only implemented and tested the case of quadratic systems over $\mathbb{F}_2$. However, the same principle applies to higher degree and other (small) finite fields of coefficient.

In addition, the complete complexity analysis of our new algorithm is quite complicated and involves many parameters. It will be the focus of a future work.

Acknowledgments

This work has been supported in part by the European Union's H2020 Programme under grant agreement number ERC-669891.

References

1. M. Bardet, J.-C. Faugère, B. Salvy, and P.-J. Spaenlehauer. On the complexity of solving quadratic boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.
2. M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. Presented at MEGA'05, Eighth International Symposium on Effective Methods in Algebraic Geometry, 2005.
3. C. Bouillaguet, H.-C. Chen, C.-M. Cheng, T. Chou, R. Niederhagen, A. Shamir, and B.-Y. Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In *Cryptographic hardware and embedded systems – CHES 2010. 12th international workshop, Santa Barbara, USA, August 17–20, 2010. Proceedings*, pages 203–218. Berlin: Springer, 2010.
4. N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology – EUROCRYPT 2000*, Lecture Notes in Comput. Sci., pages 392–407. Springer, 2000.

5. D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry. 2nd ed.* New York, NY: Springer, 2nd ed. edition, 2005.
6. V. Dubois and N. Gama. The degree of regularity of HFE systems. In *Advances in cryptology – ASIACRYPT 2010. 16th international conference on the theory and application of cryptology and information security, Singapore, December 5–9, 2010. Proceedings*, pages 557–576. Berlin: Springer, 2010.
7. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *J. Pure Appl. Algebra*, 139(1-3):61–88, June 1999.
8. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of ISSAC'02*, pages 75–83, New York, NY, USA, 2002. ACM.
9. J.-C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Grbner Bases. In B. Dan, editor, *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *Lecture Notes in Comput. Sci.*, pages 44–60. Springer Berlin / Heidelberg, 2003.
10. A. S. Fraenkel and Y. Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Appl. Math.*, 1:15–30, 1979.
11. G. Fusco and E. Bach. Phase transition of multivariate polynomial systems. *Math. Struct. Comput. Sci.*, 19(1):9–23, 2009.
12. L. Granboulan, A. Joux, and J. Stern. Inverting HFE is quasipolynomial. In *Advances in cryptology – CRYPTO 2006. 26th annual international cryptology conference, Santa Barbara, California, USA, August 20–24, 2006. Proceedings*, pages 345–356. Berlin: Springer, 2006.
13. R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.
14. A. Kipnis, J. Patarin, and L. Goubin. Unbalanced oil and vinegar signature schemes. In *Advances in cryptology—EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Comput. Sci.*, pages 206–222, Berlin, 1999. Springer.
15. D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Computer algebra (London, 1983)*, volume 162 of *Lecture Notes in Comput. Sci.*, pages 146–156. Springer, Berlin, 1983.
16. D. Lokshtanov, R. Paturi, S. Tamaki, R. Williams, and H. Yu. Beating brute force for systems of polynomial equations over finite fields. To appear in 27th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017).
17. P. L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In *Advances in cryptology - EUROCRYPT '95. International conference on the theory and application of cryptographic techniques, Saint-Malo, France, May 21-25, 1995. Proceedings*, pages 106–120. Berlin: Springer-Verlag, 1995.
18. J. Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In *Advances in cryptology – EUROCRYPT '96. International conference on the theory and application of cryptographic techniques, Saragossa, Spain, May 12-16, 1996. Proceedings*, pages 33–48. Berlin: Springer, 1996.
19. E. Thomae and C. Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. In *Public key cryptography – PKC 2012. 15th international conference on practice and theory in public key cryptography, Darmstadt, Germany, May 21–23, 2012. Proceedings*, pages 156–171. Berlin: Springer, 2012.
20. E. Thomé. Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. *Journal of Symbolic Computation*, 33(5):757–775, 2002.

21. T. Yasuda, X. Dahan, Y.-J. Huang, T. Takagi, and K. Sakurai. MQ Challenge: Hardness evaluation of solving multivariate quadratic problems. NIST Workshop on Cybersecurity in a Post-Quantum World, 2015. http://eprint.iacr.org/2015/275.