DEMOCRACY—A Heuristic for Polynomial Systems of Equations over Finite Fields (Extended Abstract)

Gregory V. Bard

April 30, 2010

Abstract

This paper presents a heuristic for solving polynomial systems of equations on finite fields larger than GF(2), via stochastic local search (SLS). It was inspired by the SLS-based SAT-solvers G-SAT, Walk-SAT and Tabu-SAT. Called DEMOCRACY, the equations vote on which values for a given variable will satisfy as many as possible of them. Variables, one at a time, are thusly changed from an initial random setting, until all equations are satisfied.

We show that this algorithm can solve the Brent Equations, a cubic system of 64 equations and 84 unknowns, taken mod 7, in 2–8 minutes. On this system, MAGMA's F4 algorithm crashed for lack of memory, and Singular via SAGE timed out after 10 hours. Similar results have been observed for random quadratic systems of equations also.

Key Words: polynomial system, finite fields, stochastic local search algebraic cryptanalysis, the Brent equations.

1 Introduction

The study of the solution of polynomial systems of equations over GF(2) has attracted a great deal of attention in the past decade, particularly because of multivariate cryptography and block-cipher cryptanalysis. In fact, there are too many papers to cite. But this raises the question of other finite fields. Why should GF(3), GF(5) or GF(7) be so different? Many papers have been written about solving polynomial systems over finite fields in general, and the method-of-choice for such problems is Jean-Charles Faugere's algorithm known as F4 [**?**]. Furthermore, it is known that the implementation in MAGMA [MAG] is the best available at this time. Here is presented an alternative approach.

Several features of problems to be solved (via polynomial systems of equations) in multivariate cryptography, error-correcting codes, algebraic cryptanalysis, graph coloring, or combinatorial designs, are notable and unusual. The first and most pivotal is that one is usually interested only in those solutions inside the base field, not inside the algebraic closure. However, this can be easily remedied by adding the equation $x^q - x = 0$ for all variables x, where q is the size of the base field. Second, in most of the above cases, we know a solution exists and we merely wish to find it. The case of "no solution" does not occur—for example in algebraic cryptanalysis a message was indeed sent, and so it was sent under some key. Third, if there are multiple solutions, we need only one solution. It is often stated that a problem with a large but finite number of distinct solutions is the most difficult for any Gröbner Bases approach. This is discussed in the paper [Bar08]. Fourth, the problems are often huge, so the usage of memory can be the limiting factor in the feasibility of the computation.

Over GF(2), while F4 is often faster, SAT-solvers often address these issues. Here we present an interesting problem from the theory of matrix multiplication (which we call "The Brent Equations" in honor of Richard Brent, see Table 1.) The method in this paper—called DEMOCRACY—solves a system of equations

$$\sum_{k=1}^{k=s} \alpha_{wxk}\beta_{yzk}\gamma_{ijk} = \delta_{iw}\delta_{xy}\delta_{jz}$$

$$\forall \left\{ \begin{array}{ll} 1 \leq i \leq c & 1 \leq w \leq a \\ 1 \leq x \leq b & 1 \leq y \leq b \\ 1 \leq z \leq c & 1 \leq j \leq c \end{array} \right.$$

Table 1: The Brent Equations for Rectangular Shortcuts

related to this theory, in 8 minutes. The system has 64 equations, 84 unknowns, and is taken over GF(7). Each equation is a cubic form set to either zero or one. Meanwhile, on the same system, MAGMA crashes due to allocating too much memory, and SAGE/Singular [S^+] [sin] timed-out after 10 hours. Each of the four above conditions is represented here. First, that we only desire base-field solutions. But, the "field equations" of the form $x^7 - x = 0$ are problematic, because they are degree seven, namely 4 degrees above the equations of the system. Second, we know a solution exists (more detail will be given in Section 5.1). Third, one solution is as useful as knowing several solutions. Fourth, our method uses only 4 megabytes. Just like SAT-solvers, DEMOCRACY is not in need of much memory. These comparisons are not surprising, because this paper's approach was inspired by the SAT-solver community.

A SAT-solver takes a series of logic statements, called clauses, written in many variables. Clauses can be composed of "implies", "and", "or", "not", as well as "if and only if", but not "for all" nor "there exists. The SAT-solver tries to solve them by assigning a truth value (either true or false) to each variable, so that every clauses comes to evaluate as true. While this is an NP-Complete problem, for thousands of variables, and hundreds of thousands of clauses, many SAT-solvers can respond within minutes.

The world of SAT-solvers is divided into two categories. The first are the conflict-driven solvers, such as minisat [ES05], which have demonstrated their usefulness in solving polynomial equations mod 2 [BCJ06]. This category has dominated the SAT world for about a decade. The second are those that work off of stochastic local search, and it is these which the author sought to explore. These too have been tried in cryptanalysis, but with rather limited success [Mas99] [MM99].

Essentially, the first stochastic local search SAT-solver was G-SAT [SLM92], which evolved into Walk-SAT almost immediately [SKC93], and that had many descendants which were mostly minor variations. An ancestor of these is [Pap91]. Rather than convert polynomial systems mod odd primes into SAT problems, it soon became apparent that it would be easier to take the methodologies and principles of those algorithms and heuristics from the SLS-SAT world and apply them to the special case of polynomial systems of equations over finite fields larger than $\mathbb{GF}(2)$, by making a new, custom algorithm.

1.1 Target Problems

There are four main target problems in this research. The first, is to be able to conduct algebraic cryptanalysis (as well as hash-function attacks) on cryptosystems already based on finite fields of odd characteristic. The second is graph coloring. The third are the Brent Equations [Bre70], given in Table 1, explained later and in [Bar09]. The fourth are certain problems in combinatorial designs. Here the graph theory and design theory examples are omitted, as they are not near completion.

2 DEMOCRACY

In this section is described the basic heuristic. Later sections will add features to it and optimizations. Perhaps one would like to call DEMOCRACY an algorithm, but there is no reason to expect it to terminate in general, so one cannot. In fact, for systems with empty solution set, it will run forever. For this reason, we are compelled to call it a "heuristic".

2.1 Inspiration

Suppose one has a system of equations over a finite field. Fix a variable of interest. Perhaps

this variable, x, appears only with exponent 1 or 2 in the system. Then, for any equation that has an x, it can be rewritten as a quadratic or linear polynomial in x alone, in the following sense:

$$w^4y^3x+3x^2y^2+4wz^3x-x^2z+3wyz+w^5+w^2y^2$$

can be made to become

$$(3y^2-z)x^2+(w^4y^3+4wz^3)x+(3wyz+w^5+w^2y^2)$$

so if we treat x as unknown and the other variables as known, we have an ordinary quadratic equation.

Now if x appears with degree 4 or less, with the quadratic equation and the formulas of Gerolamo Cardano (1501–1576), we can solve for x explicitly—if all the other variables are known. Let us consider the quadratic equation:

$$ax^2 + bx + c = 0 \rightarrow \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

becomes in the finite field world

$$ax^2 + bx + c = 0 \rightarrow \left(-b + QR(b^2 - 4ac)\right)(2a)^{-1}$$

where $QR(n)$ is any of the set of elements in the finite field that square to n. If $n = 0$ this will be 0, otherwise there will be either 2 solutions or no solutions. It is easy to implement QR as an array lookup. Cardano's cubic equation is not quite as easy as the quadratic equation but we have implemented it, based on the entry in Dummit & Foote [DF03, Ch. 14.6], and the assistance of Kyle Kloster, a Fordham undergraduate in the mathematics department.

One might imagine that if x appears with degree 5, then because of the work of Évariste Galois (1811–1832) on the quintic equation, we do not have any formula that will work all the time. But one could, if desperate, loop through all finite field values. This is obviously computationally expensive. A must wiser alternative are two algorithms, one by Berlekamp [Ber67], and the other by Cantor–Zassenhaus [CZ81], which can solve polynomials over finite fields, regardless of degree, if in one variable. They are not polynomial time in general, but are for any fixed degree (this is called pseudo-polynomial time). Furthermore, the multiplicative groups of finite fields are cyclic, and so all Galois groups over them are solvable groups. Due to a result of David Dummit [Dum91], this can be used to solve the equations.

2.2 An Election

First, let us have some temporary assignment of values to the variables, which satisfy many but not all of the equations. Therefore the system of equations is not yet solved, and so perhaps we decide by some mechanism to be defined later that x is a variable that we would like to change.

We could solve each equation by the method given in the previous subsection, and see what we get for each one. There is one "ballot box" for each finite field value, and one for "absention". These are just mere counters, of course.

- If x does not appear in the equation, then the equation votes for abstention. This is because the equation either looks like $0 = 0$, and so is satisfied regardless of the value of x (and is thus already satisfied), or it is of the form $1 = 0$, and so is not satisfied and cannot become so.
- Note that x can appear with coefficients which "cancel out" or which become zero in the temporary assignment, and so the above bullet happens a bit often.
- If x appears with highest exponent as 1, 2, or 3, then simple algebra, the quadratic equation or Cardano's equation will suffice, because the resulting polynomial is degree 1, 2, or 3. The equation places a vote in the ballot box for each finite field value, such that if x is set to this, it would result in the equation being satisfied.
- At the moment, higher degree for x is impossible, as the Brent Equations are cubic, and multivariate cryptography is usually based on quadratic systems of equations. However, there are techniques to address higher degree, mentioned earlier.

This is a mathematical description of the process. In reality, the votes for any equation can

be computed *far more rapidly* via shortcuts for particular problems, and this is key to the efficiency of the heuristic. For example, in the Brent Equations, focusing on one variable results in a linear equation. A simple formula will yield the unique vote.

Why Do This? If you take the number of votes in the ballot box of the finite field value with the most votes (ignoring the abstentions box and the box for the current value of x), and subtract form it the number votes in the ballot box of the current value of x in the temporary solution, then you get the net number of newly satisfied equations. By that is meant the number that went from unsatisfied to satisfied, minus the number that went from satisfied to unsatisfied. Of course, this might be positive, zero, or negative, and we call this the "mandate" of the election. The finite field value with the most votes (excluding abstentions and the current value of x) is called the "winner" of the election. However, no other finite field value, besides the current value of x, would produce a higher mandate than the winner—though there may be ties.

2.3 An Iteration

The core of the heuristic is the following structure, which we call a "basic iteration." First, a random unsatisfied equation is selected. Second, an election is held for each variable in that equation. The election with the highest mandate is considered the "best election", and the variable of that election is set to a new value—the winner of that election. The iteration process then repeats, however many small counters and arrays that track minor features of the algorithm (for efficiency of other steps) are updated at this time.

Notes:

- The 'best election' is a global, not local, property. Indeed, the randomly chosen unsatisfied equation might not become satisfied by the changes made. What is ensured is that the number given by newly satisfied equations minus newly unsatisfied equations is as high as can be for any choice of x, except possibly its current value.

- The first paper of this type for SAT-solvers involved G-SAT [SLM92], which picks a random variable among those found in unsatisfied clauses. However, Walk-SAT [SKC93] choses an unsatisfied clause first, and then tries a random variable from that clause. The latter method, it turns out, is heavily biased among those variables which appear *frequently* versus *rarely* in unsatisfied clauses, (a very good feature) without actually tabulating that frequency! After much preliminary experimentation, we found it is better to be "greedy" and consider all variables in one unsatisfied clause, and take the one with the best mandate, though future experiments might be helpful in clarifying this.

- In the end, the best election has a mandate that is almost always -1 or better. Surely the iterations with best election mandate of 1 or more are ideal. After all, there are only finitely many equations, and if each iteration moves at least one closer to having all the equations satisfied, then eventually all equations are satisfied. Data on this point is tabulated by the program, and is available on request.

- Because the best mandate is often 0, numerous tie-breaking strategies have been explored. In fact, ties are very frequent. That will be described in a later section.

3 Assassinations

Walk-SAT [SKC93] also introduced noise into the general G-SAT structure [SLM92]. With some fixed probability, a randomly selected variable was simply flipped (recall, that is over $\mathbb{GF}(2)$. I also have a parameter for that probability, set at the command line. But after selecting a variable at random, an election is held. In $\mathbb{GF}(2)$ when you want to change a variable, there is no discussion as to what to change it

to—because there are only two values possible. But in larger finite fields, one must find a value to change it to. The electoral analog is that occasionally a politician dies in office, and so a special election must simply be called to fill the vacancy. A rate of approximately 25% turned out to be optimal.

4 Tie-Breakers

One possibility is that the algorithm could get stuck exploring a small region of the search space in an infinite loop. More precisely, perhaps the same set of three or four variables are being changed, while all other variables are not changing at all.

Traditionally, the approach used to prevent this is called Tabu-SAT [SHS03], which means that the last t variables to be changed, for some positive fixed constant t, cannot be the variable that gets changed (wins any election). The "runner-up" (second highest in votes) would be considered instead. This is particularly problematic if an equation has fewer than t clauses, and all variables in the equation have been taken recently and are thus tabu. But this was very far from effective in experiments.

Originally we defined the "age" of a variable in order to make the parameter t superfluous. The age of a variable is the number of iterations since it was last changed. If one assigns, among the n variables, the integers $\{-n+1, -n+2, -n+3, \ldots, 0\}$ as their default initial birthdays (randomly and without repeat), and one resets the birthday when the variable is changed, then one can entirely prevent ties. More plainly, we chose the election with the largest mandate, and in the event of a tie, we choose the older or younger variable to be changed, based on an age-bias parameter set at the command line. Now, further ties are impossible (since each variable always has a distinct age) and this prevents small infinite loops. Thus it is intuitive that the older-bias did experimentally better, avoiding stagnation of older variables and enlarging the search area.

Experiments proved birthdays vastly superior to tabu in running time for fixed problems. We attempt to explain this as follows. If one variable has been changed "recently" and has a mandate of $+500$, but another has not been changed in a while and has mandate -1, and all others are worse, then tabu would pass up the $+500$ and select instead the -1, while brithdays would not come into play because $+500 \neq -1$.

5 Experimental Results

Numerous experiments were performed to compare and contrast various tie-breaking strategies. The performance of the best choice was often an order of magnitude superior to the worst choice of strategy. Space forbids the details of such experiments appearing here.

5.1 The Brent Equations

The Brent Equations were solved by this method in the case $a = 2$, $b = 2$, $c = 2$ and $s = 7$. They can be taken over any field at all, and we chose GF(7). A solution of the Brent equations with $a = b = c = n$ yields an algorithm for $N \times N$ matrix multiplication in $\Theta(N^{\frac{\log s}{\log n}})$ time, over that same field and all its extension fields. Thus, our solution is an algorithm for matrix multiplication in fields of characteristic seven that runs in $\Theta(N^{\log_2 7}) = \Theta(N^{2.807})$ time—i.e. a repeat of Strassen's Algorithm. It is for this reason that we knew a solution existed to the equations, because Strassen's exists. It should be noted that DEMOCRACY, unlike F4, will loop forever on an unsolvable problem. Meanwhile, Singular via SAGE (running on computers provided by William[1] Stein) timed out after 10 hours. Furthermore, MAGMA, running under a limit of 8 gigabytes, ran out of memory and crashed. On the same machine as SAGE, the DEMOCRACY run was 2.5 minutes, and on the author's laptop, it was 8 minutes.

[1]Thanks for the National Science Foundation, Division of Mathematical Sciences, Grant No. DMS-0821725.

References

[Bar08] Gregory Bard, *Extending SAT-Solvers to low-degree extension fields of* $\mathbb{GF}(2)$, Presented at the Central European Conference on Cryptography, 2008, Available at `http://www.math.umd.edu/~bardg/extension_fields.pdf`.

[Bar09] ———, *New practical approximate matrix multiplication algorithms found via solving a system of cubic equations*, pre-print, 2009, Available at `http://www-users.math.umd.edu/~bardg/early_release.pdf`.

[BCJ06] Gregory Bard, Nicolas Courtois, and Christopher Jefferson, *Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over* $\mathbb{GF}(2)$ *via SAT-Solvers.*, Cryptology ePrint Archive, Report 2007/024, 2006, Available at `http://eprint.iacr.org/2007/024.pdf`.

[Ber67] Elwyn R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Technical Journal **46** (1967).

[Bre70] Richard Brent, *Algorithms for matrix multiplication*, Tech. Report Report TR-CS-70-157, Department of Computer Science, Stanford, March 1970.

[CZ81] David G. Cantor and Hans Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Mathematics of Computation **36** (1981), 587–592.

[DF03] David Dummit and Richard Foote, *Abstract algebra*, third ed., Wiley, 2003.

[Dum91] David Dummit, *Solving solvable quintics*, Mathematics of Computation **57** (1991), no. 195, 387–401.

[ES05] Niklas Eén and Niklas Sörensson, *Minisat — a SAT solver with conflict-clause minimization*, Proc. Theory and Applications of Satisfiability Testing (SAT'05) (Fahiem Bacchus and Toby Walsh, eds.), Lecture Notes in Computer Science, vol. 3569, Springer-Verlag, 2005, pp. 61–75.

[MAG] *Magma*, Software Package, Available at `http://magma.maths.usyd.edu.au/magma/`.

[Mas99] Fabio Massacci, *Using Walk-SAT and Rel-SAT for cryptographic key search*, Proc. 16th International Joint Conference on Artificial Intelligence (Thomas Dean, ed.), Morgan Kaufmann Publishing, 1999, pp. 290–295.

[MM99] Laura Marraro and Fabio Massacci, *Towards the formal verification of ciphers: Logical cryptanalysis of DES*, Proc. Third LICS Workshop on Formal Methods and Security Protocols, Federated Logic Conferences (FLOC-99), 1999.

[Pap91] C. H. Papadimitriou, *On selecting a satisfying truth assignment*, Proceedings of the Conference on the Foundations of Computer Science (FOCS'91), IEEE, 1991, pp. 163–169.

[S$^+$] William A. Stein et al., *Sage mathematics software (version 4.3)*, Software Package, Available at `http://www.sagemath.org/`.

[SHS03] Kevin Smyth, Holger H. Hoos, and Thomas Stützle, *Iterated robust tabu search for max-sat*, Advances in Artificial Intelligence (Y. Xiang and Chaib-Draa, eds.), Lecture Notes in Computer Science, vol. 2671, Springer-Verlag, 2003, pp. 499–519.

[sin] *Singular*, Software Package, Available at `http://www.singular.uni-kl.de/`.

[SKC93] Bart Selman, Henry Kautz, and Bram Cohen, *Local search strategies for satisfiability testing*, Cliques, Coloring, and

Satisfiability: Second DIMACS Implementation Challenge (DIMACS'93) (David S. Johnson and Michael A. Trick, eds.), vol. 26, AMS, 1993.

[SLM92] B. Selman, H. J. Levesque, and D. G. Mitchell, *A new method for solving hard satisfiability problems*, 10th National Conference on Artificial Intelligence (AAAI'92), 1992, pp. 440–446.