Extending SAT-Solvers to Low-Degree Extension Fields of $\mathbb{GF}(2)$

Gregory V. Bard, Fordham University

1. Summary

There are several ways to solve polynomial systems of equations over $\mathbb{GF}(4)$, $\mathbb{GF}(8)$, ..., $\mathbb{GF}(64)$. In algebraic cryptanalysis, one can be interested in this problem or its analog over higher-degree extensions. We introduce the method of solving these systems via SAT-solvers, as a natural extension of the work in solving polynomial systems over $\mathbb{GF}(2)$ via SAT-solvers, as shown by Bard, Courtois, and Jefferson in 2006 [3].

A SAT-solver is a black box program that given a logical sentence, will search for a set of values for the variables such that the entire sentence is true. Since this is an NP-Complete problem (or rather the existence of such a set of values is NP-Complete), and solving polynomial systems of equations is also (or rather determining if a set of polynomials has a common solution), then one should solve the other. It turns out publicly available SAT-solvers are quite good.

In particular, we will show how to use a classic matrix representation of these fields to efficiently produce algebraic normal forms (logical sentences) for the multiplication operation of the extension field as viewed from $\mathbb{GF}(2)$. These formulae can then be used to rapidly convert the system from over $\mathbb{GF}(2^n)$ to over $\mathbb{GF}(2)$, with n times as many variables. We have formulas for $\mathbb{GF}(4)$ to $\mathbb{GF}(64)$. It is important to note that the addition is trivial, because $\mathbb{GF}(2^n)$ forms a vector-space of dimension n over the field $\mathbb{GF}(2)$.

Once these formulae are found, the system can be solved with SAT-solvers (explained below) rather than with traditional Gröbner Bases methods, such as Singular [22], which uses the Buchberger algorithm, and Magma [19], which uses Faugère's F4 algorithm. Even for small systems, Magma (using F4) would crash due to a lack of memory, for example allocating 29.9 gigabytes for a system of 8 cubic equations in 8 unknowns over $\mathbb{GF}(32)$.

We also made a formula for the "para-inverse", used in the AES, which is the field element inverse, except for zero which returns zero. This too had an easy formula.

2. Methodology

For $\mathbb{GF}(2^n)$, one can view any element as

$$a = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_n\alpha^n$$

where α is the root of a degree n irreducible polynomial over $\mathbb{GF}(2)$, and the a_i are either 0 or 1. The choice of irreducible polynomial is to be one of low weight (most coefficients equal zero).

For the system of equations, a multivariate polynomial is a sum of terms. Each term is a coefficient times several variables, some of which may be raised to powers. In a cubic polynomial, there are terms of types

$$cxyz, cx^2y, cx^3, cxy, cx^2, cx, c$$

where c is a constant. For initial work, it suffices to consider the cx^2y and cx^2 classes as special cases of the $cxyz$ and cxy classes. In fact, one could generalize further. One could see the cxy case as the $cxyz$ case with $z = 1$ and also the cx case as $cxyz$ with both $y = 1$ and $z = 1$. This turned out to be inefficient, for reasons that do not fit in this extended abstract.

Thus we are left with the $cxyz$, cxy, cx and c cases. Our methodology was simple. Take $\mathbb{GF}(16)$ as an extended example:

$$A^0 = \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}, A^1 = \begin{bmatrix}0&0&0&1\\1&0&0&1\\0&1&0&0\\0&0&1&0\end{bmatrix}, A^2 = \begin{bmatrix}0&0&1&0\\0&0&1&1\\1&0&0&1\\0&1&0&0\end{bmatrix}, A^3 = \begin{bmatrix}0&1&0&0\\0&1&1&0\\0&0&1&1\\1&0&0&1\end{bmatrix}, A^4 = \begin{bmatrix}1&0&0&1\\1&1&0&1\\0&1&1&0\\0&0&1&1\end{bmatrix}, \cdots$$

This satisfies the matrix equation $A^4 + A + I = 0$ which is the analog of $\alpha^4 + \alpha + 1 = 0$, a minimal polynomial for $\mathbb{GF}(16)$. Writing an element as a linear combination of these matrices allows us to get a product as a final matrix. But how do we go back to the field element?

TABLE 1. Experimental Results: Magma, Singular, and Mini-SAT.

Num Vars	Num Eqns	β or Sparsity	Magma	Singular	SAT
2	2	1.0	0.02 sec	0.01 sec	0.01 sec
3	3	1.0	0.04 sec	0.04 sec	0.07 sec
4	4	1.0	0.43 sec	423.48 sec	213.56 sec
5	5	1.0	4.32 sec	$>$ 75 mins	19278.9 sec
6	6	1.0	42.78 sec	no trial	$>$ 75 mins
7	7	1.0	1139.8 sec	no trial	no trial
8	8	1.0	crashed[a]	no trial	no trial
9	9	1.0	no trial	no trial	no trial
4	4	0.2	0.03 sec	0.08 sec	0.02 sec
5	5	0.2	0.55 sec	14.89 sec	61.89 sec
6	6	0.2	10.04 sec	6.74 sec	0.03 sec[b]
7	7	0.2	52.89 sec	$>$ 70 mins	4111.71 sec
8	8	0.2	crashed[c]	no trial	$>$ 75 mins
9	9	0.2	no trial	no trial	no trial
4	2	1.0	20.39 sec	$>$ 70 mins	0.14 sec
5	3	1.0	192.69 sec	no trial	20.51 sec
6	3	1.0	$>$ 70 mins	no trial	17.44 sec
7	4	1.0	no trial	no trial	5388.9 sec
8	4	1.0	no trial	no trial	no trial

[a] At one point the process had allocated 29.9 Gigabytes of RAM.
[b] This phenomenon remains unexplained, but is reproducible on repeated trials.
[c] At one point the process had allocated 24.9 Gigabytes of RAM.

We do this by "dead give aways", which are entries in the above matrices that are 1 for some power of A, but 0 for all other powers of A. For example, M_{12} will always be the coefficient of A^3, because of the mentioned property. Thus with one single matrix computation, we can find the general formula for each of the multiplications. These were quite large, taking up about 1/3 of a page for $\mathbb{GF}(32)$, each.

3. Experiments

We also have run some simple experiments on random polynomial systems of equations. The SAT-solver timings were comparable with Gröbner Bases approaches, but slower than them, when the Gröbner Bases software did not crash. However, Gröbner Bases approaches often run out of memory. On the other hand, SAT-solvers do not require much memory more than required to store the problem statement. This means they allow a patient user to attack problems that are otherwise infeasible, due to a lack of memory.

Curiously, however, we found a problem where SAT-solvers tremendously out-performed Gröbner Bases algorithms. Those situations where there were half as many equations as unknowns.

Also curiously, it appears that the standard assumption, that the F4 algorithm by Faugère *is always faster* than the Buchberger algorithm, is actually false. There were examples where the Buchberger algorithm did better. The experimental results are summarized in Table 1.

4. Applications

The algebraic cryptanalysis community has been interested in systems of polynomial equations over finite fields of characteristic two for quite some time. However, the finite field in question has usually been $\mathbb{GF}(2)$, with notable exceptions, including several ciphers. These include Rijndael [7] which later become the Advanced Encryption Standard (AES), which uses $\mathbb{GF}(256)$, and over

the same field, the cryptosystem called TTM [12]. Another example, for $\mathbb{GF}(2^{32})$, can be found in [13]. The Courtois Toy Cipher (CTC) [1] [6] can be modeled over $\mathbb{GF}(8)$ for certain settings.

5. Other Applications

There are applications to radio channel assignment, graph coloring, and compiler optimization, as well as distributing computing, but those are not relevant to the conference so we omit them for now.

References

[1] Martin Albrecht, *Algebraic Attacks on the Courtois Toy Cipher*, M.S. Thesis (Diplomarbeit), University of Bremen (Universität Bremen), Department of Computer Science, 2006.

[2] Gregory Bard. "Algorithms for the solution of linear and polynomial systems of equations over finite fields, with applications to cryptanalysis." PhD Thesis, Department of Applied Mathematics and Scientific Computation, University of Maryland at College Park, Summer of 2007.

[3] Gregory Bard, Nicolas Courtois, and Christopher Jefferson. "Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers." Cryptology ePrint Archive, Report 2007/024, 2006. Available at: `http://eprint.iacr.org/2007/024.pdf`

[4] Tomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. Second Edition. McGraw-Hill. 2002.

[5] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. "Efficient algorithms for solving overdened systems of multivariate polynomial equations." In Proceedings Of Eurocrypt 2000, LNCS 1807, pages 392–407. Springer, 2000.

[6] Nicolas Courtois. "How fast can be algebraic attacks on block ciphers?" Cryptology ePrint Archive, Report 2006/168, 2006. Available at: `http://eprint.iacr.org/2006/168.pdf`

[7] Joan Daemen and Vincent Rijmen. "Aesproposal:Rijndael", 1999. Available at `http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael-ammended.pdf`

[8] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner basis (f4)", 1999. Available at `http://modular.ucsd.edu/129-05/refs/faugeref4.pdf`

[9] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases without reduction to zero (f5)." In Proceedings of ISSAC, pages 7583. ACM Press, 2002.

[10] Niels Ferguson, Richard Schroeppel, and Doug Whiting. "A simple algebraic representation of Rijndael." Proceedings of Selected Areas in Cryptography: SAC01, Vol. 2259, *Lecture Notes in Computer Science*, Pages 103–111. Springer-Verlag, 2001.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[12] Louis Goubin and Nicolas Courtois. "Cryptanalysis of the TTM Cryptosystem." ASIACRYPT 2000: 44–57.

[13] Thomas Jakobsen. "Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree." Proceedings of CRYPTO 1998. *Lecture Notes in Computer Science* 1462. 1998.

[14] D. Joyner and R. Kreminski and J. Turisco. Applied Abstract Algebra. Free Internet Textbook. 2002. `http://web.ew.usna.edu/~wdj/book/book.html`

[15] Stephan Lucks. Private conversation with Stephan Lucks.

[16] Stein, William, *Sage Mathematics Software (Version 2.7)*, Computer Algebra software package. The Sage Group, 2007, `http://www.sagemat.org`.

[17] National Institute of Standards and Technology (NIST) publication U.S. FIPS PUB 197 on November 26, 2001.

[18] BOINC. Berkeley Open Infrastructure for Networked Computing. `http://boinc.berkeley.edu/`

[19] Magma. Computer Algebra software package. `http://magma.maths.usyd.edu.au/magma/`

[20] Maple. Computer Algebra software package. `http://www.maplesoft.com/`

[21] MiniSAT. Freely available SAT-solver. `http://minisat.se/Papers.html`

[22] Singular. Computer Algebra software package. `http://www.singular.uni-kl.de/`

[23] Nicolas Courtois. "Algebraic attacks over GF(2k), application to HFE challenge 2 and sash-v2." In Public Key Cryptography: PKC 2004, pages 201–217. Springer, 2004.