The Application of Polynomials over the Field of Two Elements to a problem in Intellectual Property

Gregory Bard

Department of Mathematics, Fordham University, The Bronx, NY, 10458

Abstract

It is a routine task to convert a digital circuit to a system of polynomial equations over GF(2). A very well-studied set of tools called "SAT-solvers", which solve Conjunctive Normal Form logical-satisfiability problems, can be used to determine if two circuits are equivalent, as is commonly done in computer engineering. An interesting problem in intellectual property is to determine if two circuits are identical but after some unknown permutation of the inputs and outputs. This could be of interest if a manufacturer suspects his encryption circuit has been stolen. While this is easily shown to be a sub-problem of the famously hard "isomorphism of polynomials" problem, we show that in fact it can be easily solved via conversion to a polynomial system of equations over GF(2), and is only slightly harder than if the permutations were in fact known.

Key words: Isomorphism of Polynomials, Intellectual Property Problem, Morphism of Polynomials, Circuit Satisfiability, Circuit Equivalence, SAT, Permutations of Polynomial Systems.

1. Introduction

The application of verifying the equivalence of two digital circuits (i.e. that they output the same for all inputs) has been very well studied, and is usually accomplished using boolean algebra in some way, though there are many techniques. In this paper we will address a particular question in Intellectual Property, which is a generalization of determining the equivalence of two circuits, but yet is a sub-problem of the very difficult and less well-understood problem of "isomorphism of polynomials." We show that this

⋆ This research was partly supported by funding from the Department of Mathematics of Fordham University

Email address: bard@fordham.edu (Gregory Bard).

URL: http://www.math.umd.edu/ bardg/ (Gregory Bard).

particular question is much easier than previously suspected, as conjectured by Patarin, Goubin, and Courtois in (12).

1.1. Background: Circuit Equivalence

This problem is crucial in the "formal verification" of digital circuits. The relationship to polynomials over $\mathbb{GF}(2)$, the field with two elements, arises from the fact that a combinatorial digital circuit is a collection of logic gates, and each logic gate is a simple polynomial over $\mathbb{GF}(2)$. Thus, because the composition of two polynomials is a polynomial, the entire circuit is a polynomial. For example, the following

Digital Gate	Logical Operation	Polynomial
x AND y	Conjunction	xy
x OR y	Disjunction	$x+y+xy$
NOT x	Negation	$1+x$
x XOR y	Distinction	$x+y$
x NAND y	Incompatibility	$1+xy$
x XNOR y	Equivalence	$1+x+y$
IF x THEN y ELSE z	Switching	$z+xy+xz$
MAJ(x,y,z)	Majority Vote	$xz+xy+yz$

are some simple logic gates with their polynomials. Thus converting a digital circuit to its polynomial system is often straightforward, and results in one equation for each output bit, and one variable for each input bit. Let m denote the number of equations (number of outputs) and n the number of variables (number of inputs), as is standard.

The "circuit equivalence" problem consists of verifying if all outputs are equal for all inputs. For two circuits f and g, this can be written

$$\exists i \in \{1, 2, \ldots, m\} \quad \text{st} \quad f_i(x_1, \ldots, x_n) + g_i(x_1, \ldots, x_n) = 1$$

because if that system of polynomial equations has a solution, then the two circuits are not equivalent. Likewise, if there is no solution, then the circuits are equivalent.

A simpler form is

$$(f_1 + g_1 + 1)(f_2 + g_2 + 1) \cdots (f_m + g_m + 1) = 0$$

which has a solution if and only if the circuits are not equivalent. We will assume that an oracle exists to solve this problem. For example, one can write polynomials and use a computer algebra system such as SAGE (3) or MAGMA (1), or one can convert to circuit satisfiability and use a SAT-Solver (6), such as MiniSAT (2). A SAT-Solver is a software tool, which given a logical sentence written in the predicate calculus, tries to find an assignment of true and false to each variable to make the entire sentence come out true. While this is NP-hard, SAT-solvers tend to be very fast on problems arising from applications. Converting a $\mathbb{GF}(2)$-polynomial system of equations into a satisfiability problem for a SAT-solver is not too difficult, as first shown in (6).

Regardless if solving via SAT-solvers or computer algebra systems, the principle measures of difficulty of the problem are the number of equations m and the number of variables n, the maximum degree, the maximum weight [1] and the average weight. If an algorithm solves this problem in time upper-bounded by some polynomial in m and n, and without any other restrictions, then P=NP.

1.2. Background: The IP Problems

Suppose manufacturer A believes that manufacturer B has stolen an important circuit, and they wish to detect or confirm this belief. In reverse engineering circuit B, they might not know which output wires correspond to their own output wires, and likewise, which inputs of circuit B correspond to which inputs of circuit A. There are $m!$ possible permutations of the outputs and $n!$ possible permutations of the inputs, and thus a total of $n!m!$ possible permutations. Thus, one might naïvely believe that $n!m!$ oracle executions would be required. We will show that with some new variables and equations, this is not required.

If one views the m-equation, n-variable polynomial system as a map $F : \mathbb{GF}(2)^n \to \mathbb{GF}(2)^m$, then this question can be modeled as follows: Does there exist an $n \times n$ permutation matrix P and an $m \times m$ permutation matrix Q such that

$$G(\vec{x}) = QF(P\vec{x}) \qquad \forall \vec{x} \in \mathbb{GF}(2)^n$$

where F is the polynomial system for one circuit, and G is the polynomial system for the other.

If one drops the requirement that P and Q be permutation matrices, and require them instead to be merely non-singular, then one has the known and difficult "Isomorphism of Polynomials Problem" (11). If one further drops the requirement of non-singularity then it is the "Morphism of Polynomials Problem" which is proven NP-hard in (12). We are not the first to consider P and Q to be permutation matrices, as was done by Patarin, Goubin and Courtois in (12). In that paper, they prove that the permutation-matrix case is reducible to Graph Isomorphism, and conjecture that it is much easier than the non-singular-matrix case. Appendix A contains a description of the connection between the "Isomorphism of Polynomials Problem" and public-key cryptography.

While algorithms for solving the "Isomorphism of Polynomials Problem" have been published for certain cases, and many of them are feasible, these algorithms are very complex and some forms of the problem have no known feasible methods of solution (9), (12), (8). The Intellectual Property Problem seems similar, so one might imagine it too, is difficult. We will solve the problem in the next section.

2. The Solution

We will now solve the intellectual property problem, seeing if G is a copy of F merely with the inputs and outputs permuted. Consider, for simplicity, $n = m = 8$. Then the polynomial function

[1] The weight is the number of terms per polynomial.

$$\begin{aligned}\mu(i_2, i_1, i_0, s_0, s_1, s_2, \ldots, s_7) = {} & (1+i_2)(1+i_1)(1+i_0)s_0 + (1+i_2)(1+i_1)(i_0)s_1 + \\ & (1+i_2)(i_1)(1+i_0)s_2 + (1+i_2)(i_1)(i_0)s_3 + \\ & (i_2)(1+i_1)(1+i_0)s_4 + (i_2)(1+i_1)(i_0)s_5 + \\ & (i_2)(i_1)(1+i_0)s_6 + (i_2)(i_1)(i_0)s_7\end{aligned}$$

has the useful property that

$$\mu(i_2, i_1, i_0, s_0, s_1, s_2, \ldots, s_7) = s_{4i_2+2i_1+i_0}$$

and that, in turn, follows from the fact that this function will always have seven of its eight terms "zeroed-out" by i_2, i_1, i_0 regardless of the values that those three variables take. The remaining term is the correct s. The circuit given by this polynomial is called an 8-way multiplexer by computer engineers.

We "attach" one of these to each input of G, in the following sense. Let

$$x'_j = \mu(i_{j,2}, i_{j,1}, i_{j,0}, x_0, x_1, x_2, \ldots, x_7) \qquad i \in \{1, 2, 3, \ldots, n\}$$

where the x_js are the inputs to F, and μ is the polynomial defined above. The x'_js are the inputs to G.

So long as the triple $(i_{j,2}, i_{j,1}, i_{j,0})$ is distinct from the triple $(i_{k,2}, i_{k,1}, i_{k,0})$ for all $j \neq k$, where $j \in \{1, 2, \ldots, 8\}$ and k also, then the mapping from the x_js to the x'_js is a permutation. This further can be caused to occur by adding the equation

$$\begin{aligned}(i_{j,2}+i_{k,2}) + (i_{j,1}+i_{k,1}) + (i_{j,0}+i_{k,0}) + (i_{j,2}+i_{k,2})(i_{j,1}+i_{k,1}) \\ +(i_{j,2}+i_{k,2})(i_{j,0}+i_{k,0}) + (i_{j,1}+i_{k,1})(i_{j,0}+i_{k,0}) \\ +(i_{j,2}+i_{k,2})(i_{j,1}+i_{k,1})(i_{j,0}+i_{k,0}) = 1\end{aligned}$$

for all $k \neq j$ where $j \in \{1, 2, \ldots, 8\}$ and k likewise. This is because that equation will be satisfied in all cases unless $i_{j,\ell} = i_{k,\ell}$ for all $\ell \in \{0, 1, 2\}$.

The same operation can be performed on the outputs, converting y_js into y'_js and using $o_{j,2}, o_{j,1}, o_{j,0}$ in place of $i_{j,2}, i_{j,1}, i_{j,0}$. The total cost of this is $3 \times 8 \times 2 = 48$ new variables, plus $2 \times 8 = 16$ new equations for the y' and x' definitions, and another $2\binom{8}{2} = 56$ equations for the guaranteeing of the non-identicality of the triples of is and of the triples of os. That is a total of 72 new equations, and 48 new variables.

One oracle call is sufficient to answer the question after these changes, which is significantly less work than $(8!)^2 \approx 1.63 \times 10^9$ oracle calls. Finally, note that this not only answers the decision question (are the circuits equivalent or not), but also explicitly identifies the specific permutations used on the inputs and the outputs, being the values of the $i_{1,0}, \ldots, i_{8,2}$ and the $o_{1,0}, \ldots, o_{8,2}$ variables, respectively.

2.1. The General Case

At the time of this writing, typical SAT-Solvers are able to handle 100s of variables, and 1000s of sparse equations (for cryptographic problems [2]), and so while this is non-trivial, the system will remain feasible. Also, the F_4s Algorithm of Jean-Charles Faugère

[2] For easier problems, one may square these quantities, and the SAT-solver community has identified ensembles of very hard, very small problems.

(7) implemented in MAGMA is a common tool for this task. An overview of solving $\mathbb{GF}(2)$-polynomial systems of equations appears in (4, Ch. 11, Ch. 12).

The $m \neq 8$ or $n \neq 8$ cases can be handled by allowing the second index of the is to be at most $\lceil \log_2 n \rceil$ and of the os to be at most $\lceil \log_2 m \rceil$. In this case there will be

$$n \lceil \log_2 n \rceil + m \lceil \log_2 m \rceil$$

new variables, and

$$m + n + \binom{n}{\lceil \log_2 n \rceil} + \binom{m}{\lceil \log_2 m \rceil}$$

new equations.

2.2. Very Large Cases

However, for the case $m = n = 32$, we would add 320 variables and 402,816 equations. This number of equations is not feasible by current methods. The map from x to x' will have five is because $32 = 2^5$; we call $i_{j,4}$, $i_{j,3}$, $\ldots$, $i_{j,0}$ a "quintuplet".

Omitting the equations that guarantee the non-identicality of the quintuplets seems foolish at first, but in reality, the following argument is likely to hold in any realistic scenario.

The equations that guarantee non-identicality of the quintuplets force P and Q as functions to be bijections. If two quintuplets are identical then two input variables of F map to the same input variables of G or two output variables of G map to the same output variable of F.

In the former case, by the pigeonhole principle, this means that an input variable of F was neglected, and not used by G at all. If a solution to the entire system is found, that means that one input wire in the design of F was superfluous, (in other words, not needed or used at all). This is very unlikely, but the prover knows about F and would be aware of that condition before beginning the problem. This case is therefore irrelevant to the practical problem.

In the latter case, again by the pigeonhole principle, this means that an output of G was matched with two distinct outputs of F, and in any solution of the system that means that these two outputs of Fs are always identical, otherwise how can they both equal a particular output of G simultaneously? This means an output of F was superfluous, merely a copy of another output of F. This can happen because of "fan-out" considerations in Electrical Engineering, but again, the designers of F would know about this in advance, and one need not consider this case either.

Thus the equations for non-identicality can be dropped, and there will be only $m + n$ new equations. In this case, we are searching not only for P and Q that are permutation matrices, but the broader class of matrices with exactly one non-zero entry per row.

The only invertible members of this broader class of matrix are the permutation matrices themselves, since any matrix with two identical rows is singular. Therefore, this broader class is not of interest to the public-key system for which the Isomorphism of Polynomial problems was proposed.

Therefore, only 320 new variables and 64 new equations are required. While adding 320 variables is extremely non-trivial, it should be noted that normally in circuit verification there is roughly one variable per gate. A circuit with 32-bits of input and output would probably have very roughly $32^2 = 1024$ gates (or 2048 for both f and g) and thus it is only a 15.63% increase. Moreover, it was shown in (5) that SAT-Solvers perform extremely well when solving very underdetermined systems of equations.

2.3. Comparison with Patarin, Goubin, and Courtois

Alternatively, the proof of Theorem 4.1 in (12) had an approach to this problem, except with a directed graph instead of a circuit. They required $n = m$ and $(2n^3 - 3n^2 - n + 4)/2$ equations and $(n+1)(n^2 - 2n + 2)/2$ variables, which is much more. But they do not assume the existence of an oracle that can solve the non-permuted case, which is our model of a SAT-Solver or MAGMA. For example, with the above case 31,218 equations and 15,873 variables would be needed by (12).

Obviously, the question of the existence of the oracle is of little interest to (12). What they show is that they can turn a black-box that solves the "isomorphism of polynomials" problem into something that solves the "graph isomorphism problem", to show that the former is at least as hard as the latter. Here, we show a reduction from "intellectual property" to "circuit equivalence", to show the former is only slightly harder than the latter. A side note is that (12) add more equations and more unknowns than this paper adds, but what they add is quadratic (our equations are of higher degree). Degree is of no consequence for SAT-solvers, but it is of consequence for Gröbner Bases algorithms. The comparison here was pointed out by Charles Bouillaguet, whom we thank profusely.

3. Conclusions

While the "intellectual property" problem considered here is a sub-problem of the famously difficult "isomorphism of polynomials" problem, it is only slightly harder than the task of determining if two (non-permuted) circuits are identical. Furthermore, one obtains the actual permutations used with no additional effort. Thus the criminal activity can be easily detected.

A. Application to Communications Security

The Isomorphism of Polynomials problem was first proposed in connection with communications security. Suppose one has c polynomial systems of equations each with n variables and n unknowns. Viewing these as a map $F : \mathbb{GF}(2)^{cn} \to \mathbb{GF}(2)^{cn}$ is possible, but it is far easier to find pre-images under this map F than some arbitrary G with the same domain and range. That is because the c polynomial systems could be each solved separately.

Now fix two $cn \times cn$ dense invertible matrices over $\mathbb{GF}(2)$, call them S and T, and let $G(\vec{x}) = SF(T\vec{x})$. We must also assume that given G alone, it is infeasible to find S and T, and furthermore that it is hard to find other matrices that will likewise decompose G into a different, but still easy, system of equations. The system G will be the public key, and collectively T, S, F are the private key. To encrypt a message $\vec{x}$, one simply computes $G(\vec{x}) = \vec{y}$, which is easy, and transmits $\vec{y}$. The recipient calculates $\vec{z} = S^{-1}\vec{y}$, splits z into c vectors of length n, and then solves the c systems of n equations and n unknowns. The solutions would be c vectors of dimension n. The solutions are concatenated into 1 vector of dimension cn, called $\vec{w}$. Finally, one multiplies $T^{-1}\vec{w} = \vec{x}$ to obtain the original message. An adversary, being without T, S, or F would have to solve a system of cn equations in cn unknowns, which for careful parameter choices, is not feasible.

Unfortunately, these schemes were broken in many cases (9), (12), (8).

B. Criminal Motivations

A block cipher is a map $f : \mathbb{K} \times \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, where $\mathbb{K}$ is the set of possible keys, and $\{0,1\}^\ell$ is the set of all possible plaintext and ciphertext messages, namely the set of all ℓ-bit strings. The usual model for the security of a block-cipher is that encryption under a random key $k \in \mathbb{K}$ should "behave similarly" to a permutation selected at random from the set of those with the domain and range $\{0,1\}^\ell$. Therefore, permuting the inputs, the outputs, or both, cannot degrade the security properties of a block cipher, if the original meets this imprecise standard. Extensions of this standard, involving computational indistinguishability by probabilistic polynomial-time adversaries can make this standard precise (10). In short, the reason a competitor might steal and permute the circuit is that if σ is computationally indistinguishable from a random permutation, and τ_1 and τ_2 are fixed permutations, then $\tau_1 \circ \sigma \circ \tau_2$ is computationally indistinguishable from a random permutation. Thus, if the problem in this paper were hard, then permuting the inputs and outputs of the cipher would provide an easy escape from patent-licensing fees on ciphers or their silicon implementations.

Acknowledgements

The author would like to thank an employee of the United States Department of Defense, who suggested this problem, but who wishes to remain anonymous. Furthermore, Robert Lewis, Ludovic Perret, Domingo Gomez Perez, and most particularly Charles Bouillaguet provided useful feedback on drafts of this paper.

References

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

[2] MiniSAT. Software Package. Available at `http://www.cs.chalmers.se/cs/Research/FormalMethods/MiniSat/` or `http://minisat.se/Papers.html`

[3] Sage. Software Package. Available at `http://www.sagemath.org/`

[4] Bard, G.: *Algebraic Cryptanalysis.* Springer-Verlag. (2009).

[5] Bard, G.: "Extending SAT-Solvers to low-degree extension elds of GF(2)." Preprint, Presented at the Central European Conference on Cryptography (2008). Available at `http://www.math.umd.edu/~bardg/extension_fields.pdf`

[6] Bard, G., Courtois, N., Jefferson, C.: "Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-Solvers." Preprint. Cryptology ePrint Archive, Report 2007/024 (2006). Available at `http://eprint.iacr.org/2007/024.pdf`

[7] Faugère, J.-C. "A new efficient algorithm for computing Gröbner Bases (F_4)." *Journal of Pure and Applied Algebra.* **Vol. 139**. Pp. 61–88. (1999).

[8] Faugère, J.-C., and Perret, L.: "Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects." In: S. Vaudenay (Ed.): *Advances in Cryptology—Proc. of EUROCRYPT, Lecture Notes in Computer Science*, **Vol. 4004**, pp. 30–47. Springer-Verlag (2006). Available at `http://www-spaces.lip6.fr/@papers/FP06b.pdf`

[9] Geiselmann, W., Meier, W., and Steinwandt, R.: "An attack on the isomorphisms of polynomials problem with one secret". *International Journal of Information Security.* **Vol. 2**, No. 1, (2003). Available at `http://eprint.iacr.org/2002/143.pdf`

[10] Goldwasser, S., and Micali, S. "Probabilistic Encryption." *Journal of Computer Systems Sciences.* **Vol. 28**, No. 2, Pp. 270–299. (1984).

[11] Patarin, J.: "Hidden Field Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms." In: N. Koblitz (ed.) *Advances in Cryptology—Proc. of EUROCRYPT, Lecture Notes in Computer Science*, **Vol. 1070**, pp. 33–48. Springer-Verlag (1996). Available at `http://citeseer.ist.psu.edu/article/patarin96hidden.html`

[12] Patarin, J., Goubin, L., and Courtois, N.: "Improved Algorithms for Isomorphisms of Polynomials." In: K. Nyberg (Ed.): *Advances in Cryptology—Proc. of EUROCRYPT, Lecture Notes in Computer Science*, **Vol. 1403**, pp. 184–200. Springer-Verlag (1998). Available at `http://citeseer.ist.psu.edu/article/patarin98improved.html`

[13] Bartee, T.C.: *Digital Computer Fundamentals*, sixth edn. McGraw Hill (1985).