Exact Logic Minimization and Multiplicative Complexity of Concrete Algebraic and Cryptographic Circuits

Nicolas T. Courtois
University College London,
Gower Street, London, UK
n.courtois@cs.ucl.ac.uk

Theodosis Mourouzis
University College London,
Gower Street, London, UK
theodosis.mourouzis.09@ucl.ac.uk

Daniel Hulme
University College London,
Gower Street, London, UK
d.hulme@cs.ucl.ac.uk

***Abstract*—Two very important NP-hard problems in the area of computational complexity are the problems of Matrix Multiplication (MM) and Circuit Optimization. Solving particular cases of such problems yield to improvements in many other problems as they are core sub-routines implemented in many other algorithms. However, obtaining optimal solutions is an intractable problem since the space to explore for each problem is exponentially large. All suggested methodologies rely on well-chosen heuristics, selected according to the topology of the specific problem. Such heuristics may yield to efficient and acceptable solutions but they do not guarantee that no better can be done. In this paper, we suggest a general framework for obtaining solutions to such problems. We have developed a 2-step methodology, where in the first place we describe algebraically the problem and then we convert it to a SAT-CNF problem, which we solve using SAT-solvers. By running the same procedure for different values of k we can obtain optimal solutions and prove that no better can be done. We decrease the k until "UNSAT" is obtained. Using the suitable encoding step for each problem we have been able to obtain exact and optimal solutions for different problems which are sufficiently small, allowing us to solve them on an average PC. We have been able to prove the exact number of multiplications needed for multiplying two non-square matrices of sufficiently small dimensions, as well as obtaining optimal representations with respect to meaningful metrics for several S-boxes used in prominent ultra-lightweight ciphers such as GOST, PRESENT and CTC2.**

***Index Terms*—Linear Algebra, Fast Matrix Multiplication, Complex Numbers, quaternions, Strassen's algorithm, Multiplicative Complexity, Asynchronous Circuits, Logic Minimization, Automated Theorem Provers, Block Ciphers, CTC2, PRESENT, GOST, SAT solvers**

I. Introduction

Optimization of arbitrary algebraic computations over rings in the general non-commutative setting is considered as one of the most interesting topics in theoretical computer science and mathematics. In general such optimization problems are expected to be computationally very hard [1], [2].

In this paper, we study two fundamental problems. We study the problem of minimizing the Multiplicative Complexity (MC) of algebraic computations, such as the Matrix Multiplication (MM) [1]. MC is the minimum number of AND gates that are needed, if we allow an unlimited number of NOT and XOR gates. Informally, we are interested in reducing the number of multiplications involved in an arbitrary algebraic computation to the lowest possible bound, allowing unlimited number of additions. Initially, we study the optimization problem over small fields such as $\mathrm{GF}(2)$. However, in some cases solutions found do not yet yield a general solution for a larger ring, and there can be additional lifting steps [1].

The second problem we study is the combinatorial logic optimization of general Boolean circuits, with respect to a given set of elementary operations. Logic optimization is also a well-known hard problem which interests the chip maker industry and researchers in complexity. Good optimizations are particularly important in industrial hardware implementations of standard cryptographic algorithms [3], [4]. This is because cryptography is computationally very costly and the improved designs can be used in hundreds of millions of integrated circuits and produce important savings. These ciphers have very small S-boxes, yet, nobody knows how to implement them in an optimal way, and new cryptographic implementations with less gates are obtained almost each year [5], [6].

In practice, there are no known analytic techniques nor direct prescriptive algorithms, which can construct such optimal circuits. Developing an optimal circuit representation for a small-size Boolean function of the form $GF(2)^8 \rightarrow GF(2)$ with respect to AND gates remains still an open problem. Is it possible to determine once for all, what is the minimum possible number of gates? Exact bounds are very hard to be obtained in these areas, as the problem is mainly solved by heuristic techniques and is known to be computationally very hard.

In this paper, we view these problems as constraint satisfaction problems which we attempt to solve them by methods of formal coding [7] and later solve with software such as SAT solvers [8]. The striking feature of this type of methods is that, if we use a "complete" SAT solver (and we have enough CPUs), as opposed to a "stochastic" one, and if it is fast enough to complete, and it outputs UNSAT, we obtain a proven lower bound, a very rare thing in complexity.

Our method consists of three basic steps. In the first step, we formally encode the problem by writing a system of equations which describes our problem as a system of polynomial

equations over the finite field of two elements $\mathrm{GF}(2)$. In the case of the MM problem and some other of our algebraic optimizations [1], we use the Brent Equations [9] in the encoding step. Circuit minimization problems are encoded formally as a form of straight-line representation problem, which we encode them as a quantified set of multivariate relations that need to be satisfied. Then, we proceed by converting our defined modulo 2 problem to a SAT problem using the Courtois-Bard-Jefferson method [7] and then (only if required) we may add additional steps such as lifting the solution to larger fields or rings [1] or re-optimize for circuit depth, or many other [5], [6].

This type of methodology was recently applied with success to optimize linear circuits [10] and bi-linear circuits [11]. We have developed a method to do this also for non-linear circuits. We have been greatly influenced by the work of Boyar and Peralta on the AES cipher S-box and its MC, however, we can also produce many optimizations from scratch with arbitrary gates and without the Boyar-Peralta heuristic. Though this type of exact optimizations is computationally very intensive and therefore currently only possible for fairly small circuits, the preliminary results obtained are very encouraging and allow for direct applications in cryptographic hardware synthesis, systematic synthesis of implementations resistant to side-channel attacks, and also in cryptanalysis [12], [3], [13]. In this paper, we also report our results on PRESENT and GOST, two block ciphers known for their exceptionally low hardware cost.

A. Structure of the Paper

The organization of this paper is as follows:

Section II: We refer to several reasons highlighting the importance of solving such problems. We outline several improvements which yield in many other applications, as a consequence of improving the state-of-art algorithms for solving MM and combinatorial circuit optimization problems. Obtaining even solutions to small problems may yield significant improvements to general problems as these solutions to smaller instances can be recursively used to handle the general problem.

Section III: We describe all technical details of our 2-step methodology. Initially, we describe the encoding step which we employ for each problem. For example in case of MM as a tool of encoding we use the Brent Equations, while for optimizing circuits we invented a general framework of encoding. Then we briefly analyze how to obtain the corresponding CNF-SAT problem of a given problem, which is algebraically encoded. Additionally, we describe provably aspects of our methodology and we highlight how powerful are the SAT solvers for solving exactly such NP-hard problems.

Section IV: We apply our methodology for obtaining new formulaes for multiplying two non-necessarily square matrices. We have been able to solve exactly with respect to the number of multiplications needed to multiply such matrices, for sufficiently small matrices. Several small instances are solved and presented.

Section V: We optimize arbitrary non-linear digital circuits for silicon implementation and cryptanalysis. We apply our methodology for obtaining optimal circuit representations for the S-boxes used in many prominent ultra-lightweight block ciphers such as PRESENT and GOST. For experimentation we have been able to optimize the 3-bit to 3-bit S-box of CTC cipher with respect to different meaningful metrics.

II. Motivation for Low MC and Low Gate Count Optimizations

We briefly outline what will be the benefits in both academic and industrial world if some better optimizations are found for the problems of MM and gate-efficient implementation.

A. Matrix Multiplication Problem

Obtaining the minimum number of 2-input multiplications needed for computing the product of two matrices A, B, is considered among the most difficult optimization problems in the area of computer science and mathematics. Given two matrices A, B the MM is defined as follows (*Def.* 1).

Definition 1: (Matrix Multiplication [MM])

Let A and B two $n \times n$ matrices, $n \in \mathbb{N}$, with entries in a ring $\mathcal{R}$ (not necessarily commutative), such that

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \text{ and}$$

$$B_{m,n} = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{pmatrix}.$$

Then, the entries of the product matrix $C = AB$ are given by

$$C_{p,q} = [AB]_{p,q} = \sum_{i=1}^{n} a_{p,i} b_{i,q}. \quad (1)$$

The multiplication is defined as in the ring $\mathcal{R}$.

We are interested in obtaining new formulaes for computing the product matrix C involving as few number of 2-input multiplications as possible. Fast linear algebra for large matrices leads to significant improvements in many other areas as follows [14]:

- Commercial software such as MATLAB, MATHEMATICA and GAUSS
- Economic Modeling
- Weather prediction
- Signal processing
- Gauss Elimination algorithm for solving a system of linear polynomial equations
- Algorithms for solving non-linear polynomial equations
- Recognizing if a word of length n belongs to a context-free language
- Transitive closure of a graph or a relation on a finite set
- Statistical analysis of large data sets

- Integer factorization
- Cryptanalysis

B. Circuit Complexity

There are many reasons why circuits of low MC are very important especially for industrial applications and for cryptography. More analytic explanations can be found in [3].

- Develop certain so called Bitslice parallel-SIMD software implementations of block ciphers such as in [15]
- Lower the hardware implementation cost of encryption algorithms in silicon chips
- Prevent Side Channel Attacks (SCA) on smart cards such as Differential Power Analysis (DPA) [16]

C. Cryptanalysis Applications:

In addition, more or less all optimizations in this paper have direct applications in software cryptanalysis, cf. [12], [3], [4], [13]. One reason for that is the fact that dense linear algebra is very frequently the last step in many algebraic attacks. Another family of applications are algebraic attacks on symmetric cryptography. A more compact representation of a cipher is known to improve the running time of many such attacks [12], [13].

III. A SAT-solver Based Methodology

A. General Methodological Framework

In this paper, we formally encode optimization problems as systems of multivariate equations over $GF(2)$, then we convert them to SAT problems, and then we solve them.

This sort of encoding is subject to continuous improvement, and it can be viewed itself as a hard but solvable optimization problem. Therefore, our current timings (current methods are quite slow) and results are very likely to be improved in the future in very substantial ways.

We describe all the main steps in our approach. Our main scientific contributions are the methodology and concrete results that are mathematical theorems about lower complexity bounds and concrete optimizations, which are reusable building blocks for algorithms, scientific computing and the industry. For those results which we show to be optimal they can no longer be improved, however, the timing of obtaining these results (computations that need to be done only once) can still certainly be improved.

In what follows, we present two major encoding methodologies. The Coding Methodology 1 is designed to address the efficient MM problem and other algebraic optimizations, We use the Brent Equations [9] in the encoding step.

The Coding Methodology 2 is designed to address more general problems of MC and other optimizations of arbitrary circuits which are no longer initially described by multivariate polynomial expressions, but by a truth table. Circuit minimization problems are encoded formally as a form or straight-line representation problem, then we describe it as a quantified set of multivariate relations [3], and then it has to hold simultaneously for different input values in the truth table.

B. Coding Methodology 1

Our algorithm for solving for MM problems is as follows:

1) Form the Brent Equations (or write a quantified set of multivariate relations that describes the problem)
2) Consider only solutions in 0,1=integers modulo 2
3) Convert to SAT with Courtois-Bard-Jefferson method [7]
4) Lift the solution from $\mathrm{GF}(2)$ to the general bigger fields by another constraint satisfaction algorithm

C. Brent Equations

Brent Equations are used for encoding problems in a more formal algebraic language. After the encoding we convert this problem to a SAT problem and then we try to obtain a solution using several SAT solvers.

Brent's Method: Suppose we want to multiply a $M \times N$ matrix A by a $N \times P$ matrix B using T 2-input multiplications. We solve the above problem by solving the following system of $(MNP)^2$ equations in $T(MN + NP + MP)$ unknowns, cf. [9]:

$$\{\forall i \forall j \forall k \forall L \forall m \forall n, \sum_{p=1}^{T} \alpha_{ijp}\beta_{kLp}\gamma_{mnp} = \delta_{ni}\delta_{jk}\delta_{Lm}\}$$

A solution to this set of equations implies that the coefficient entries c_{ij} of the product matrix $C = AB$ can be written as

$$c_{nm} = \Sigma_{p=1}^{T}\gamma_{mnp}q_p,$$

where the products $q_1, q_2, ..., q_T$ are given by the expression $q_p = (\Sigma\alpha_{ijp}a_{ij})(\Sigma\beta_{KLp}b_{KL})$.

This form of encoding can be generalized for describing other problems such as complex number multiplication and quaternion multiplication.

D. Coding Methodology 2

This methodology is very different, and in fact it is possible to see that for many circuits both methodologies could be applied. The motivation for this second method is that not every circuit is very algebraic and it can be described efficiently by sparse multivariate polynomial expressions. Especially, in industrial cryptographic primitives we expect that the resulting circuits will have a very low gate count since the efficient hardware implementation is one of the main priorities of designers. Thus, it is very realistic that a system of equation describing such a cryptographic primitive will be very sparse.

Therefore, the Brent-like approach could lead to a very large problem to solve. However, we can also describe the initial problem as a substitution box with a truth table. We proceed as follows.

First, we write a certain system of equations $\mathcal{C}1$ in which variables will be divided in several disjoint categories:

1) We will have the "x" variables which will be inputs of the truth table.
2) The "y" variables which will be outputs of the truth table.
3) The "t" variables which will be inputs of gates.
4) The "q" variables which are outputs of gates.
5) The "b" variables will define the function of each gate. (For example, one gate could be AND, OR, XOR and

the model is $b(uv) + b'(u + v)$, and when $b = 1, b' = 0$ this will be AND gate.)

6) The "a" variables which will be the unknown connections between different gates.

A crucial problem is how these connections are described in $\mathcal{C}1$. The purpose of the "a" variables is to make each new "t" variable depended on some combinations of past variables of type "x","t","q". These "a" variables will encode ALL the unknown connections between different gates, their inputs, their outputs, our inputs "x" and our outputs "y".

For example, in MC optimizations we can say that each variable is an affine or linear combination of previous variables. In other optimizations we can furthermore add constraints of type $a_i a_j = 0$ which say that in a certain set of a_i only at most one of these variables is at 1. We provide a toy example of our algebraic description $\mathcal{C}1$ below:

```
t1 = a1 * x1 + a2 * x2
a1 * a2 = 0
t2 = a3 * x1 + a4 * x2
a3 * a4 = 0
q1 = t1 * t2
y1 = a5 * q1 + a6 * x1 + a7 * x2
y1 = x1 * x2
```

Another problem is how to describe the relation between the inputs "x" and the outputs "y" efficiently. In the toy example above this is done in the last line. Several methods for coding small size I/O relations are described in [12]. It is the open problem to see what is the best method and the best method will depend a lot on the type of the circuit. One very good method is to use a previous circuit with more gates (!).

Now our circuit minimization problem is encoded formally as a straight-line computation problem. Then we expand this system of equations $\mathcal{C}1$ into another system of equations $\mathcal{C}2$ as follows:

1) Our problem $\mathcal{C}1$ is a quantified system of constraints. We need to determine the variables of types "a" and "b".
2) Our circuit $\mathcal{C}1$ (see toy example above) must be true for every "x". We can privilege values of small Hamming weight (for some circuits we do NOT need to put all possible values of "x".
3) We make several copies of the circuits and we rename the "x" and "y" and "q" and "t" variables, however, the "a" and "b" variables remain common in all circuits.
4) We write all these circuits as systems of multivariate relations [3] and concatenate them. We call $\mathcal{C}2$ the resulting system of equations.
5) We convert $\mathcal{C}2$ to SAT and solve for the "a" and "b" variables.
6) We take the values of the "a" and "b" variables, we ignore all the other assignments, and substitute in the original (single) circuit model $\mathcal{C}1$.
7) We check if our solution is correct, and potentially optimize for XORs, to decrease the number of intermediate variables, etc.

E. SAT Solver Step

Satisfiability (SAT) is the problem of determining if the variables of a given Boolean formula can be assigned in a way as to make the formula evaluate to TRUE [17]. SAT was the first known example of an NP-complete problem. A wide range of other decision and optimization problems can be transformed into instances of SAT and a class of algorithms called SAT solvers can efficiently solve a large enough subset of SAT instances such as MiniSAT solver [8]. Our aim is to transform problems like MM into SAT problems.

SAT solvers had both theoretical and practical improvements and have made a lot of progress in recent years. The basis of most SAT solvers is the Davis-Putnam backtrack search, which searches for a solution by recursively choosing a variable and trying to assign to it one value and then the other. At each stage of search a propagation step is performed which attempts to imply the assignments to as many unassigned variables as possible based on previous assignments. As a result of this it may uncover a clause which cannot be satisfied, so search backtracks.

In the major SAT competition every year, almost all the previous years winners are beaten by new competitors who design more efficient solvers. Thus SAT solvers are carefully designed to run on a large range of problems with no tuning required by users.

Problems arising either from academia or from industry can be solved by SAT solvers if are converted to Conjunctive Normal Form (CNF). In Boolean logic, a formula is in CNF if it is a conjunction of clauses, where a clause is a disjunction of literals.

At a first glance, this seems to be inefficient as conversion to CNF can skip extra structural information of the original problem. However, the performance of SAT solvers is often able to offset this structural information loss.

F. On the Complexity of SAT-solvers

Unfortunately, the time complexity of a SAT solver is not easy to determine. A very large system in CNF can be easily solved by a SAT solver on an average PC, but beyond some point the probability of solving such a system from 1 becomes 0.

In cryptanalysis, that implies we can derive the key of a reduced version. As the number of rounds grows, the time complexity of such an algebraic attack becomes infinite. Unfortunately, there is no clear indication when the problem becomes infeasible.

G. Conversion to SAT

In order to solve a problem using a SAT solver we need to convert this problem to its CNF (cf. *Def.* 2).

Definition 2: (Conjunctive Normal Form)

A Boolean function f is said to be in conjunctive normal form, if it is a conjunctive of clauses, where each clause is a disjunction of literals, i.e., f can be expressed in the form

$$\wedge_{I \subseteq M}(\vee_{i \in I} x_i),\ M = \{1, .., n\}$$

We have been using three major methods to convert a system of multivariate polynomial equations over $GF(2)$ to a SAT problem. This idea has been pioneered by Bard and Courtois see [12] and has become a very important tool in modern cryptanalysis and automated problem solving.

As these methods are quite slow, it is too early to say which one is better for the purpose of our optimizations.

1) We can use the Courtois-Bard-Jefferson tool [7] which is available for download.
2) Another method is local approximation, it has been frequently used in cryptanalysis, see [12], [13].
3) Yet another method is to use a SAT solver which accepts native XORs, such as CryptoMiniSat by Soos [18], and therefore new conversion methods can be proposed, see [13] for some applications of these very promising new encodings which seem to be really excellent in cryptanalysis applications however, they have not yet been tested in the setting of this paper.

A very basic approach to map a given problem to SAT-CNF is firstly to derive a 2-degree system of equations from the algebraic description of the problem using the fact that [7]:

$$\{m = wxyz\} \Leftrightarrow \{a = wx, b = yz, m = ab\}$$

In general, CNF expressions describe instances of SAT problems, thus we need to obtain the CNF of this multivariate system of quadratic equations. This conversion proceeds by three major steps as follows [7]:

STEP 0: The CNF form must not contain any constants. Since all clauses must be true in a solution, we introduce constants by adding clauses of the form $T \vee T \vee ... \vee T$, which implies that variable T is true in any satisfying solution. T encodes constant 1, while $\overline{T}$ encodes 0.

STEP 1: (Polynomial System to Linear System)

Every polynomial is a sum of linear and higher degree terms. Given a monomial $a = wxyz$ over $\mathbb{F}_2$, then this is tautological equivalent to

$$a \Leftrightarrow (w \wedge x \wedge y \wedge z)$$

$$(w \vee \bar{a})(x \vee \bar{a})(y \vee \bar{a})(z \vee \bar{a})(a \vee \bar{w} \vee \bar{x} \vee \bar{y} \vee \bar{z}).$$

Thus, for each monomial of degree d we have $d+1$ clauses, while the total length of clauses is $3d + 1$.

STEP 2: (Linear System to CNF expression)

After expressing each monomial involved the next step is to express the logical XORs. The sum $a \oplus b \oplus c \oplus d \oplus = 0$ is equivalent to:

$$(\bar{a} \vee b \vee c \vee d)(a \vee \bar{b} \vee c \vee d)(a \vee b \vee \bar{c} \vee d)(a \vee b \vee c \vee \bar{d})$$

$$(\bar{a} \vee \bar{b} \vee \bar{c} \vee d)(\bar{a} \vee \bar{b} \vee c \vee \bar{d})(\bar{a} \vee b \vee \bar{c} \vee \bar{d})(a \vee \bar{b} \vee \bar{c} \vee \bar{d})$$

However, handling long XORs is a hard problems for SAT solvers. For example, given a sum of length h we split it into different sub-sums and encode each sum separately. More details can be found in [7] since the scope of this paper to contribute towards the encoding step and both conversion and solving techniques can be considered as black-box procedures. Note that the conversion procedure in this section is polynomial in time and more details are found in [7].

H. Provably Optimal Aspects of Our Methods:

All the optimizations which are claimed EXACT in this paper are optimal: they have been proven impossible to further improve. This is achieved with an automated software proof with UNSAT and would be PROVABLY OPTIMAL if we had a proof of correctness of the SAT solver software and of course if there is no bug in the SAT solver software.

For example, a SAT solver could claim UNSAT for a certain problem or even output an incorrect proof of UNSAT. However, we can overcome this problem as we have a portfolio of around 500 different SAT solvers software and we can re-check our results with other SAT solvers. Even if we assume the presence of bugs in this software, one can consider that our proofs are *probabilistic proofs*.

Possibly the probability of error could be very small and under some additional assumptions we could have better confidence that our automated proof is indeed correct. We also claim that what we do could be extended to produce fully verifiable mathematical proofs written in a formal language, which prove these optimality results. Some SAT solvers already have the ability to output such proofs.

In order to obtain optimal solutions with respect to a count k for a problem X we proceed as follows in *Algorithm* 1:

Algorithm 1: Given a decision problem X and a count k for the metric of our interest proceed as follows:

1) Convert this to SAT-CNF
2) Obtain "SAT" and a solution
3) Set $k := k - 1$
4) Repeat Until "UNSAT"
5) Output: $k_{\min}$ such that is "SAT"

IV. On Solving the MM Problem

A very common approach for tackling the MM problem is to work by solving fixed-size problems and then apply the solution recursively to higher dimensions. The general framework for gluing together solutions for smaller instances and obtain solutions to the general problem is provided by the *divide-and-conquer* paradigm [19].

The complexity of solving the general problem depends on the complexity of solving the underlying smaller sub-problems. Thus, even a slight improvement in such a sub-problem may lead to a huge improvement in the general problem. This general concept can be seen as a pure combinatorial optimization problem with fixed size, which have been studied by many authors since Strassen [2], [20].

In this section, we provide a short description regarding the complexity of existing techniques for solving the MM up-to-date. Additionally, we apply our SAT-based methodology for solving smaller instances of the MM problem. We present new formulaes for multiplying sufficiently small matrices and in some cases we are able to prove that these formulaes are optimal with respect to the number of 2-input multiplications required.

A. On the Complexity of MM

The complexity of the naive algorithm for computing the product of two $n \times n$ matrices is $\mathcal{O}(n^3)$ and similarly the complexity for multiplying a $m \times p$ matrix by a $p \times n$ matrix is $\mathcal{O}(mpn)$. Clearly, as the computation of the product matrix of two $n \times n$ matrices contains n^2 entries, that implies at least n^2 operations are needed and that a proven lower bound for the complexity is $\mathcal{O}(n^2)$.

Thus, the exponent of MM problem over a general non-commutative ring $\mathcal{R}$ defined as

$$\omega(\mathcal{R}) := \inf \tau \in \mathbb{R} | \mathbb{M}_{\mathcal{R}} = \mathcal{O}(n^{\tau})$$

Improving the exponent τ of the complexity $\mathcal{O}(n^{\tau})$ of MM problem is one of the main interests of the academic community. The first attempt was in 1969 by Volker Strassen who has been able to decrease the complexity of square MM to $O(n^{2.807})$, by applying recursively the optimal solution he obtained for multiplying two 2×2 matrices with 7 multiplications [20] (cf. *Theorem* 1).

Theorem 1: (Strassen's Algorithm using 7 Multiplications) Given two 2×2 matrices A, B over a ring $\mathcal{R}$, with entries $a_{i,j}, b_{i,j} \in \mathcal{R}$ $1 \leq i, j \leq 2$, then the entries $c_{i,j}$ of the product matrix $C = AB$ can be computed by the following formulaes,

$P_1 = (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2})$
$P_2 = (a_{2,1} + a_{2,2})b_{1,1}$
$P_3 = a_{1,1}(b_{1,2} + b_{2,2})$
$P_4 = a_{2,2}(-b_{1,1} + b_{2,1})$
$P_5 = (a_{1,1} + a_{1,2})b_{2,2}$
$P_6 = (-a_{1,1} + a_{2,1})(b_{1,1} + b_{1,2})$
$P_7 = (a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2})$,
$c_{1,1} = P_1 + P_4 - P_5 + P_7$
$c_{1,2} = P_2 + P_4$
$c_{2,1} = P_3 + P_5$
$c_{2,2} = P_1 + P_3 - P_2 + P_6$

Afterwards, Coppersmith and Winograd developed an algorithm to perform MM of square matrices of complexity $\mathcal{O}(n^{2.376})$ [21]. They achieved such a complexity reduction by proving new formulas for computing the inner product of two n-dimensional vectors using fewer 2-input multiplications.

Later in 1975, Laderman published a solution for multiplying 3×3 matrices with 23 multiplications [22]. Since then, this topic generated very considerable interest and yet to this day it is not clear if Laderman's solution in case of 3×3 multiplication can be further improved. For example, we cannot prove if 23 is optimal and no formulas exist using 22 multiplications or even less.

In 2005, a team of scientists from Microsoft Research and two US universities established a new method for finding such algorithms based on group theory, and their best method so far gives an exponents of 2.41 [23], close to Coppersmith-Winograd result and subject to further improvement. However, exponent τ is quite low and it is conjectured that one should be able to do MM in so called *soft quadratic time*, with possibly some poly-logarithmic overheads, which could even be sub-exponential in the logarithm. This in fact would be nearly linear in the size of the input. Amazingly enough, many scientists conjecture that it could be nearly quadratic like $\mathcal{O}(n^2(log_2(n))^a)$, for some a.

Our Contribution: In this paper, we proceed by solving the corresponding Brent equations for a given MM problem[9], by converting it into a SAT-CNF problem. This approach has been tried many times before, cf. [9], [17]. Note that this methodology is more generic and it is also applied to multiplication of non-square matrices.

B. New Formulaes for MM Problem

Using our SAT-based methodology as described in previous chapters, we have been able to obtain the following results as presented in Table I.

TABLE I
THE OUTPUT OF APPLYING OUR METHODOLOGY FOR SOLVING THE MM PROBLEM USING A FIXED NUMBER OF MULTIPLICATIONS

Inputs	No.Mults.	SAT	Av.Time(s)
2,2,2	7	YES	0.55
2,2,2	6	NO	1062.7
2,2,3	11	YES	474.5
2,2,3	10	NO	4032.2
2,2,4	16	YES	0.63
2,2,4	15	YES	3152.8

As we see from the same table we can prove that multiplying two 2×2 matrices can not be done using less than 7 multiplications and thus Strassen's formulaes are optimal.

Using stochastic SAT solvers, we can solve exactly the decision problem: "Can we multiply two matrices A, B using exactly k 2-input multiplications?". We have tried to solve all these underlying decision problems for small problems and we have been able to prove that no better can be done. A new exact result is as formulated below in *Theorem* 2.

Theorem 2: Given two matrices matrices $A \in M_{2\times 2}(\mathcal{R})$ and $A \in M_{2\times 3}(\mathcal{R})$ where R an arbitrary non-commutative ring , then we can compute the product matrix $C = AB$ using at most 11 multiplications

Proof: An upper bound for solving this problem is by naive MM and it is 12 multiplications in total over a general non-commutative ring $\mathcal{R}$.

First, we consider the Brent Equations corresponding to 11 multiplications. Thus, we obtain 144 equations in 176 unknowns (12098 right clauses).

Then, we convert it to a SAT problem, which we solve using CryptoMiniSat in approximately 474.54s=0.132h. We have obtained the following set of equations for solving the MM problem using 11 multiplications.

$P01 := (-a_{11} - a_{12} + a_{21} + a_{22}) * (b_{23})$;
$P02 := (-a_{11} - a_{12} + a_{21}) * (b_{12} - b_{23})$;
$P03 := (a_{11} - a_{21}) * (-b_{13} + b_{23})$;
$P04 := (a_{11}) * (b_{11})$;
$P05 := (a_{22}) * (-b_{21} + b_{23})$;

$P06 := (-a_{11} - a_{12}) * (b_{12})$;
$P07 := (a_{21}) * (b_{12} - b_{13})$;
$P08 := (a_{22}) * (b_{21} - b_{22})$;
$P09 := (a_{12}) * (-b_{12} + b_{22})$;
$P10 := (a_{21}) * (-b_{11} + b_{12})$;
$P11 := (a_{12}) * (b_{21})$;
$c_{11} = (P04 + P11)$;
$c_{12} = (-P06 + P09)$;
$c_{13} = (P02 - P03 - P06 - P07)$;
$c_{21} = (P01 + P02 - P05 - P06 - P10)$;
$c_{22} = (P01 + P02 - P05 - P06 - P08)$;
$c_{23} = (P01 + P02 - P06 - P07)$;

Initially, only 117 out of 144 equations were also true over $\mathbb{Z}/4\mathbb{Z}$. After we applied our heuristic lifting technique we lifted all solutions over this ring and the solution was true over an arbitrary ring.

Then, we have obtained the corresponding Brent Equations for 10 multiplications, 144 equations in 160 unknowns and proceeded similarly. The output of our algorithm is UNSAT, implying that there is no solution for this problem. We have verified the UNSAT result using several other SAT-solvers for minimizing the errors due to bugs in software.

Hence, 11 multiplications is the minimal number of required multiplications for solving this problem. ■

In addition, we have applied our methodology for solving the Laderman's problem for multiplying 2 3×3 matrices using 23 multiplications. Amazingly, we have obtained a new non-isomorphic solution to the same problem and we present it below in *Theorem* 3.

Theorem 3: Given two square matrices matrices $A, B \in M(R, 3)$ where R an arbitrary non-commutative ring , then we can compute the product matrix $C = A.B$ using at most 23 multiplications

Proof: An upper bound for solving this problem is by naive MM and it is 27 multiplications in total over a general non-commutative ring $\mathcal{R}$.

Firstly, we write down the Brent Equations corresponding to 23 multiplications. Thus, we obtain 729 equations in 621 unknowns. Then we convert them to a SAT-CNF problem, which we solve using CryptoMiniSat. The following set of equations is obtained.

$P01 := (a_{23}) * (-b_{12} + b_{13} - b_{32} + b_{33})$;
$P02 := (-a_{11} + a_{13} + a_{31} + a_{32}) * (b_{21} + b_{22})$;
$P03 := (a_{13} + a_{23} - a_{33}) * (b_{31} + b_{32} - b_{33})$;
$P04 := (-a_{11} + a_{13}) * (-b_{21} - b_{22} + b_{31})$;
$P05 := (a_{11} - a_{13} + a_{33}) * (b_{31})$;
$P06 := (-a_{21} + a_{23} + a_{31}) * (b_{12} - b_{13})$;
$P07 := (-a_{31} - a_{32}) * (b_{22})$;
$P08 := (a_{31}) * (b_{11} - b_{21})$;
$P09 := (-a_{21} - a_{22} + a_{23}) * (b_{33})$;
$P10 := (a_{11} + a_{21} - a_{31}) * (b_{11} + b_{12} + b_{33})$;
$P11 := (-a_{12} - a_{22} + a_{32}) * (-b_{22} + b_{23})$;
$P12 := (a_{33}) * (b_{32})$;
$P13 := (a_{22}) * (b_{13} - b_{23})$;
$P14 := (a_{21} + a_{22}) * (b_{13} + b_{33})$;
$P15 := (a_{11}) * (-b_{11} + b_{21} - b_{31})$;
$P16 := (a_{31}) * (b_{12} - b_{22})$;
$P17 := (a_{12}) * (-b_{22} + b_{23} - b_{33})$;
$P18 := (-a_{11} + a_{12} + a_{13} + a_{22} + a_{31}) * (b_{21} + b_{22} + b_{33})$;
$P19 := (-a_{11} + a_{22} + a_{31}) * (b_{13} + b_{21} + b_{33})$;
$P20 := (-a_{12} + a_{21} + a_{22} - a_{23} - a_{33}) * (-b_{33})$;
$P21 := (-a_{22} - a_{31}) * (b_{13} - b_{22})$;
$P22 := (-a_{11} - a_{12} + a_{31} + a_{32}) * (b_{21})$;
$P23 := (a_{11} + a_{23}) * (b_{12} - b_{13} - b_{31})$;

$c_{11} = P02 + P04 + P07 - P15 - P22$;
$c_{12} = P01 - P02 + P03 + P05 - P07 + P09 + P12$
$+P18 - P19 - P20 - P21 + P22 + P23$;
$c_{13} = -P02 - P07 + P17 + P18 - P19 - P21 + P22$;
$c_{21} = P06 + P08 + P10 - P14 + P15 + P19 - P23$;
$c_{22} = -P01 - P06 + P09 + P14 + P16 + P21$;
$c_{23} = P09 - P13 + P14$;
$c_{31} = P02 + P04 + P05 + P07 + P08$;
$c_{32} = -P07 + P12 + P16$;
$c_{33} = -P07 - P09 + P11 - P13 + P17 + P20 - P21$; ■

This new set of equations for multiplying two 3×3 matrices is non-isomorphic to the system of equations obtained by Ladermann. A full explanation and proof of this is found in [1]. This embraces the conjecture that maybe it can be done with fewer multiplications. We will try to investigate even more this in the future by either seeking for further improvements in our encoding step or running our algorithms on more CPUs working in parallel.

V. Exact Combinatorial Circuit Optimization

In this section, we apply our methodology for obtaining optimal circuit representations for sufficiently small digital circuits with respect to various meaningful metrics. We study circuit representations with respect to the following metrics:

1.*Multiplicative Complexity (MC)*:is the minimum number of AND gates (infinite number of XORs in allowed).

2.*Bitslice Gate Complexity (BGC)*: is the minimum number of 2-input gates of types XOR,OR,AND,NOT needed. This model is relevant in so called *bitslice parallel-SIMD* implementations of block ciphers, e.g. in [15].

3.*Gate Complexity (GC)*: is the minimum number of 2-input gates of types XOR,AND,OR,NAND,NOR,NXOR.

4.*NAND Complexity (NC)*: is defined by the minimum number of 2-input NAND gates.

In order to compute such circuits, we apply the heuristic methodology suggested by Boyar and Peralta [24] based on the notion of MC as follows:

Step 1: First compute the MC.

Step 2: Optimize the number of XORs separately, cf. [25], [10].

Step 3: (Optional Step) At the end do additional optimizations to decrease the circuit depth, and possibly additional software optimizations, cf. [5], [24], [6].

We apply Coding Methodology 2 and we encode the problem formally as a straight-line representation problem,

described by a quantified set of multivariate relations and we convert it to SAT with the Courtois-Bard-Jefferson tool [7] or other methods. Earlier work on computing the MC can be found in [3].

In the next section we apply our methodology for obtaining optimal representations with respect to all these circuit notions for the CTC S-box.

A. Optimal Representations of CTC S-box

More generally, Coding Methodology 2 allows to optimize for arbitrary gates, not only for MC. As a proof of concept we consider the following S-box with 3 inputs and 3 outputs, which have been generated at random for the CTC2 cipher [3] and is defined as,

$$\{7, 6, 0, 4, 2, 5, 1\}.$$

We have tried to optimize this S-box with the well known software Logic Friday (based on Espresso min-term optimization developed at Berkeley) and we obtained 13 gates, which obviously can be further improved. With our software and in a few seconds we obtained several interesting results, each coming with a proof that it is an optimal result. All our theorems are presented in the *Lemmas* below.

Lemma 4: The MC of CTC S-box is exactly 3 (we allow 3 AND gates and an unlimited number of XOR gates) (cf. Figure 1)

Proof: We have obtained the following straight-line program for this problem:

$$\begin{array}{ll}
t_{00} = x_{01} + x_{02} + 1 & y_0 = q_{00} + q_{01} + x_{02} \\
t_{01} = x_{00} + x_{02} + 1 & y_1 = q_{00} + q_{01} + q_{02} \\
q_{00} = t_{00} \times t_{01} & y_2 = q_{00} + x_{00} \\
t_{02} = x_{02} & \\
t_{03} = x_{00} + x_{01} + x_{02} & \\
q_{01} = t_{02} \times t_{03} & \\
t_{04} = x_{01} + x_{02} + 1 & \\
t_{05} = q_{00} + x_{01} + 1 & \\
q_{02} = t_{04} \times t_{05} &
\end{array}$$

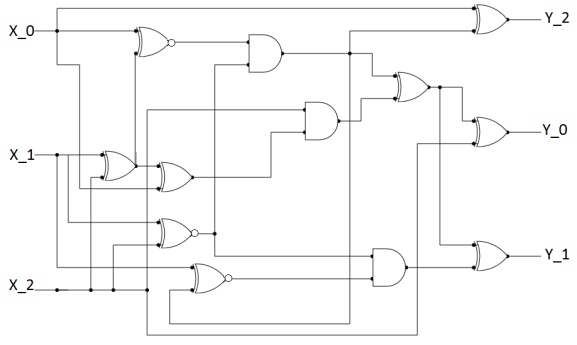

Fig. 1. Our provably optimal implementation of CTC2 S-box [3] with MC 3.

■

Lemma 5: The Bitslice Gate Complexity (BGC) of CTC S-box is exactly 8 (allowed are XOR,OR,AND,NOT) (we allow 3 AND gates and an unlimited number of XOR gates) (cf. Figure 2)

Proof: Straight-Line Program for CTC2 S-box w.r.t Bitslice Complexity

$$\begin{array}{lll}
t_{00} = x_{01} & q_{03} = t_{06} + t_{07} & t_{15} = x_{00} \\
t_{01} = x_{00} & t_{08} = q_{01} & q_{07} = t_{14} + t_{15} \\
q_{00} = t_{00} \times t_{01} + t_{00} + t_{01} & t_{09} = x_{02} & y_0 = q_{04} \\
t_{02} = q_{00} & q_{04} = t_{08} + t_{09} & y_1 = q_{05} \\
t_{03} = 1 & t_{10} = q_{00} & y_2 = q_{07} \\
q_{01} = t_{02} + t_{03} & t_{11} = q_{03} & \\
t_{04} = x_{00} & q_{05} = t_{10} + t_{11} & \\
t_{05} = x_{02} & t_{12} = q_{04} & \\
q_{02} = t_{04} \times t_{05} & t_{13} = q_{05} & \\
t_{06} = x_{01} & q_{06} = t_{12} \times t_{13} & \\
t_{07} = q_{02} & t_{14} = q_{06} &
\end{array}$$

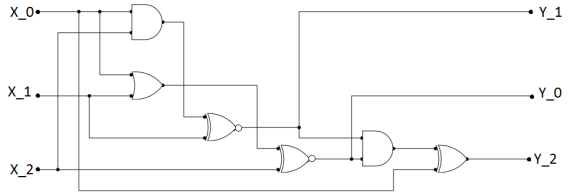

Fig. 2. Our provably optimal implementation of CTC2 S-box with Gate Complexity 6.

■

Lemma 6: The Gate Complexity (GC) of CTC S-box is exactly 6 (allowing XOR,OR,AND,NOT,NAND,NOR,NXOR) (cf. Figure 3)

Proof: Straight-Line Program for CTC2 S-box w.r.t Gate Complexity

$$\begin{array}{ll}
t_{00} = x_{01} & q_{03} = t_{06} + t_{07} \\
t_{01} = x_{00} & t_{08} = q_{03} \\
q_{00} = t_{00} \times t_{01} + t_{00} + t_{01} + 1 & t_{09} = q_{01} \\
t_{02} = x_{02} & q_{04} = t_{08} \times t_{09} \\
t_{03} = q_{00} & t_{10} = q_{00} + q_{04} \\
q_{01} = t_{02} + t_{03} & t_{11} = x_{00} \\
t_{04} = q_{01} & q_{05} = t_{10} + t_{11} \\
t_{05} = x_{00} & y_0 = q_{01} \\
q_{02} = t_{04} \times t_{05} + 1 & y_1 = q_{03} \\
t_{06} = q_{02} & y_2 = q_{05} \\
t_{07} = x_{01} &
\end{array}$$

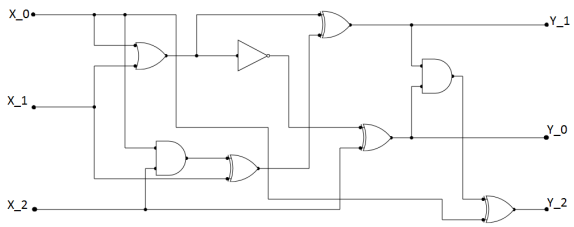

Fig. 3. Our provably optimal implementation of CTC2 S-box with Bitslice Gate Complexity 8.

■

Lemma 7: The NAND Complexity (NC) of CTC S-box is exactly 12 (only NAND gates and constants) (cf. Figure 4)

Proof: Straight-Line Program for CTC2 S-box w.r.t Bit-slice Complexity

$m_{00} = x_{00}$
$m_{01} = x_{01}$
$m_{02} = x_{02}$
$m_{03} = x_{00}$
$m_{04} = x_{01}$
$t_{00} = m_{03}$
$t_{01} = m_{02}$
$q_{00} = t_{00} \times t_{01} + 1$
$m_{05} = q_{00}$
$t_{02} = m_{05}$
$t_{03} = m_{01}$
$q_{01} = t_{02} \times t_{03} + 1$
$m_{06} = q_{01}$
$t_{04} = m_{06}$
$t_{05} = x_{02}$
$q_{02} = t_{04} \times t_{05} + 1$
$m_{07} = q_{02}$
$t_{06} = m_{06}$
$t_{07} = m_{07}$
$q_{03} = t_{06} \times t_{07} + 1$
$m_{08} = q_{03}$
$t_{08} = m_{06}$
$t_{09} = m_{04}$
$q_{04} = t_{08} \times t_{09} + 1$
$m_{09} = q_{04}$
$t_{10} = m_{05}$
$t_{11} = m_{06}$
$q_{05} = t_{10} \times t_{11} + 1$
$m_{10} = q_{05}$
$t_{12} = x_{02}$
$t_{13} = m_{10}$
$q_{06} = t_{12} \times t_{13} + 1$
$m_{11} = q_{06}$
$t_{14} = x_{00}$
$t_{15} = m_{09}$
$q_{07} = t_{14} \times t_{15} + 1$
$m_{12} = q_{07}$
$t_{16} = m_{08}$
$t_{17} = m_{12}$
$q_{08} = t_{16} \times t_{17} + 1$
$m_{13} = q_{08}$
$t_{18} = m_{10}$
$t_{19} = m_{09}$
$q_{09} = t_{18} \times t_{19} + 1$
$m_{14} = q_{09}$
$t_{20} = m_{12}$
$t_{21} = m_{13}$
$q_{10} = t_{20} \times t_{21} + 1$
$m_{15} = q_{10}$
$t_{22} = m_{15}$
$t_{23} = m_{11}$
$q_{11} = t_{22} \times t_{23} + 1$
$m_{16} = q_{11}$
$y_0 = m_{16}$
$y_1 = m_{14}$
$y_2 = m_{13}$

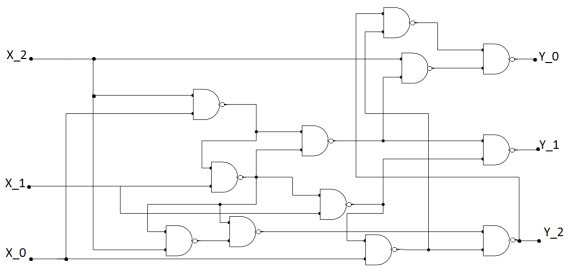

Fig. 4. Our provably optimal implementation of CTC2 S-box with NAND Complexity 12.

■

Proof: (Complimentary Optimality Proof)

Unlike the great majority of circuit optimizations, needed each time a given cipher is implemented in hardware, our results are exact. They are obtained by solving the problem at a given gate count k, the SAT solver outputs SAT and a solution, and if for k-1 gates the *SAT* solver is good enough and fast enough, it will output *UNSAT* and we obtain a proven lower bound, a rare thing in complexity. ■

B. Optimizing the PRESENT S-box

In this section we apply our methodology for optimal circuit representations with respect to bitslice implementation metric for the PRESENT S-box defined as,

$$\{12, 5, 6, 11, 9, 0, 10, 13, 3, 14, 15, 8, 4, 7, 1, 2\}$$ [26].

In Figure 5, we present a circuit implementation of it with MC 4 (i.e., using only 4 AND gates) and we prove that this circuit implementation is optimal with respect to the AND gates.

Lemma 8: The MC of the PRESENT S-box is exactly 4.

Proof: Initially, we encoded the problem using 3 AND gates and our thoroughly designed and tested system outputs UNSAT. This could be converted to a formal proof that the MC is at least 3. For 4 AND gates, our system outputs SAT and a solution. This can be seen as a software proof.

Further optimization of the linear part, which is also optimal as we also obtained UNSAT for lower numbers, allowed us to minimize the number of XORs to the strict minimum possible (prove by additional UNSAT results).

As a result, for we have obtained an implementation of the PRESENT S-box with 25 gates in total: 4 AND, 20 XOR, 1 NOT, which is optimal w.r.t our Boyar-Peralta 2-step methodology, which is as follows: In overall gate complexity since 25 gates are still not satisfactory.

Straight-Line Program for PRESENT S-box w.t.r Bit-slice Complexity

$u_{00} = x_3 + x_1$
$u_{01} = u_{00} + 1$
$u_{02} = x_1 + x_4$
$u_{03} = u_{01} + x_4$
$u_{04} = x_3 + x_2$
$u_{05} = u_{04} + x_4$
$u_{06} = u_{03} + x_2$
$u_{07} = u_{06} + u_{00}$
$u_{08} = x_1 + u_{03}$
$t_{00} = x_4$
$t_{01} = u_{05}$
$p_{00} = t_{00} \times t_{01}$
$t_{02} = p_{00} + u_3$
$t_{03} = p_{00} + x_2$
$p_{01} = t_{02} \times t_{03}$
$t_{04} = u_{04}$
$t_{05} = x_3$
$p_{02} = t_{04} \times t_{05}$
$t_{06} = u_{02} + u_{08}$
$t_{07} = p_{02} + u_{01}$
$p_{03} = t_{06} \times t_{07}$
$v_{00} = p_{03} + p_{00}$
$v_{01} = p_{03} + p_{02}$
$v_{02} = p_{00} + p_{02}$
$v_{03} = v_{02} + p_{01}$
$y_1 = p_{02} + u_{02}$
$y_2 = v_{01} + u_{06}$
$y_3 = v_{03} + u_{07}$
$y_4 = v_{00} + x_2$

A better result in terms of gate complexity can be achieved by the following method: we observe that AND gates and OR gates are affine equivalents, and it is likely that **if** we implement certain AND gates with OR gates, we might be able to further reduce the overall complexity of the linear parts. We may try all possible 2^4 cases where some AND gates are implemented with OR gates. Even better results can be obtained if we consider also NOR and NAND gates. By this method, starting with the right optimization with MC=4, as several such optimizations may exist, we can obtain the following new implementation of the PRESENT S-box which requires only 14 gates total.

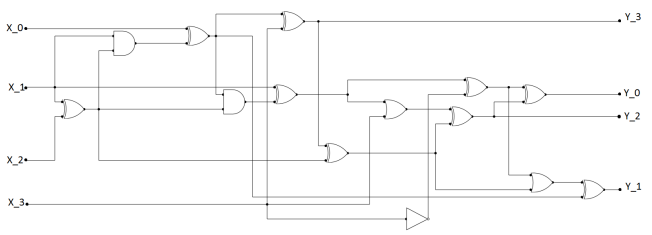

Fig. 5. Our provably optimal Bitslice-type implementation of PRESENT S-box with 14 gates.

■

Applications. This implementation is used in our recent bit-slice implementation of PRESENT, see [15]. In addition, we postulate that this implementation of the PRESENT S-box is in certain sense optimal for DPA-protected hardware implementations with linear masking, as it minimizes the number of non-linear gates (there are only 4 such gates).

Discussion. Our best optimisation of the PRESENT S-box seems to confirm the Boyar-Peralta heuristic to the effect that some of the best possible gate-efficient implementations are very closely related to the notion of MC. However, the most recent implementations of the AES S-box, in the second paper by Boyar and Peralta, show that further improvements, and also circuit depth improvements, can be achieved also by relaxing the number of ANDs used as in the latest optimization of the 4-bit inverse in $GF(2^4)$ for AES given on Figure 1 in [6].

C. Optimal Representations of GOST S-boxes

In this section, we apply our automated methodology encode-and-then-solve for obtaining optimal circuit representations of the 8 S-boxes $S1 - S8$ of GOST block cipher with respect to the number of AND gates.

GOST block cipher has a simple 32-round Feistel structure, which encrypts a 64-bit block using a 256-bit key defined in the standard GOST 28147-89 [27]. We consider the main standard and most widely known version of the GOST block cipher, known as "GostR3411 94 TestParamSet", also known as the one used by the Central Bank of the Russian Federation, and 7 additional versions which are found in the OpenSSL source code.

We have obtained for all these versions optimal representations with respect to the number of AND gates and these results are summarized in Table II.

Lemma 9: The MC of the eight S-boxes of GOST cipher S1-S8 and for 8 principal known version of GOST as specified in OpenSSL are EXACTLY equal to the values given on Table II.

Further work. With suitable encoding for other components, and in particular for the addition modulo 2^{32} in GOST cipher, we are potentially able to provably minimize the number of non-linear gates in a whole cipher to a (proven) lower bound. We can obtain very compact algebraic encodings of GOST which can used for algebraic cryptanalysis, see [4], [13].

TABLE II
MC FOR ALL KNOWN GOST S-BOXES

S-box Set Name							
$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
GostR3411_94_TestParamSet							
4	5	5	5	5	5	4	5
GostR3411_94_CryptoProParamSet							
4	5	5	4	5	5	4	5
Gost28147_TestParamSet							
4	4	4	4	4	5	5	5
Gost28147_CryptoProParamSetA							
5	4	5	4	4	4	5	5
Gost28147_CryptoProParamSetB							
5	5	5	5	5	5	5	5
Gost28147_CryptoProParamSetC							
5	5	5	5	5	5	5	5
Gost28147_CryptoProParamSetD							
5	5	5	5	5	5	5	5
GostR3411_94_SberbankHashParamset							
4	4	4	5	5	4	4	4

The better the optimizations obtained, the more compact the representation we get for a given cipher, and heuristically this leads to better algebraic attacks. This provides additional motivation for our work. Such compact representations can be combined with several complexity reduction and differential attacks to obtain attacks again full rounds of GOST.

D. Optimization of the Majority Function

The Majority function is a function of the form $\mathbb{F}_2^n \to \mathbb{F}_2$, which is False when $\frac{n}{2}$ of its inputs are false and vice versa for True. It is a highly non-trivial task to obtain circuit representations with optimal MC in the case when n is odd.

Using our 2-step automated procedure we have been able to find optimal circuit representation for the Majority function in cases when $n = 3, 5, 7$.

Lemma 10: The MC for the Majority Function when $n = 3$ is 1 (cf. Figure 6)

Proof: Using our methodology we have obtained the following circuit representation.

$$t_0 = x_0 \oplus x_2, t_1 = x_0 \oplus x_1$$
$$q_0 = t_0 \wedge t_1, q_1 = q_0 \oplus x_1$$

■

Lemma 11: The MC for the Majority Function when $n = 5$ is 3 (cf. Figure 7)

Proof: Using our methodology we have obtained the following circuit representation.

$$k_0 = x_0 \oplus x_1, k_1 = x_3 \oplus x_4, t_0 = k_0 \oplus k_1,$$
$$t_1 = x_1 \oplus x_2, q_0 = t_0 \wedge t_1, t_2 = x_0 \oplus x_3,$$
$$q_1 = k_1 \wedge t_2, k_2 = k_1 \oplus t_2, t_3 = q_1 \oplus k_2,$$
$$k_3 = x_2 \oplus x_4, t_4 = q_0 \oplus k_3, q_2 = t_3 \wedge t_4,$$
$$o_0 = q_2 \oplus x_4$$

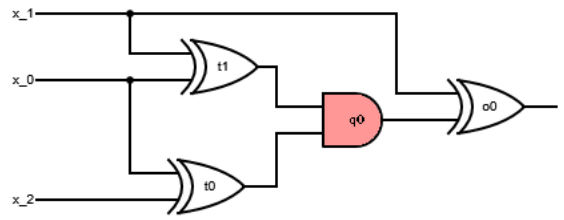

Fig. 6. Circuit Representation of Majority function on 3 inputs with optimal MC=1

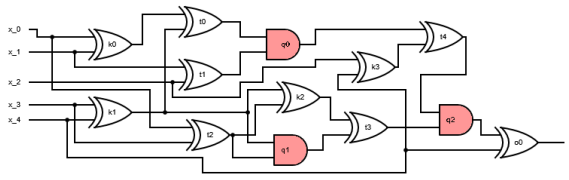

Fig. 7. Circuit Representation of Majority function on 5 inputs with optimal MC=3

■

Lemma 12: The MC for the Majority Function when $n = 7$ is 4 (cf. Figure 8)

Proof: Using our methodology we have obtained the following circuit representation. We have obtained a circuit with 23 gates in total: 4 AND gates, 1 NOT gate and 18 XOR gates.

$$t_0 = x_0 \oplus x_1, t_1 = x_0 \oplus x_2, q_0 = t_0 \wedge t_1,$$
$$t_2 = x_4 \oplus x_5, t_3 = x_3 \oplus x_4, q_1 = t_2 \wedge t_3,$$
$$p_0 = q_0 \oplus q_1, k_0 = x_1 \oplus x_2, k_1 = x_4 \oplus x_6,$$
$$k_2 = x_3 \oplus x_5, l_0 = k_0 \oplus k_1, l_1 = p_0 \oplus l_0,$$
$$t_4 = l_1 \oplus 1, t_5 = k_1 \oplus k_2, q_2 = t_4 \wedge t_5,$$
$$k_3 = x_0 \oplus x_4, t_6 = p_0 \oplus k_3, p_1 = q_0 \oplus q_2,$$
$$k_4 = x_0 \oplus x_6, t_7 = p_1 \oplus k_4, q_3 = t_6 \wedge t_7,$$
$$p_2 = q_0 \oplus q_3, o_0 = p_2 \oplus x_0$$

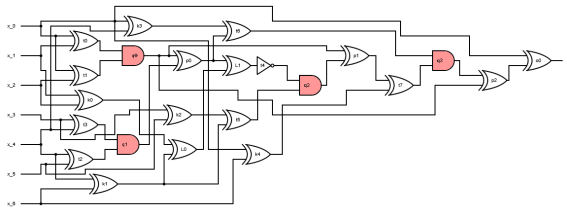

Fig. 8. Circuit Representation of Majority function on 7 inputs with optimal MC=4

■

VI. Conclusion

The construction of efficient computational circuits and the optimization of arbitrary algebraic computations over fields and rings in the general non-commutative setting is considered as one of most important problems in computer science, applied mathematics and the industry. It has numerous applications in improving various linear/polynomial/graph/language/cryptographic algorithms and an important footprint in many current applications. For example, in improving basic all-purpose linear algebra routines or in efficient implementation of important cryptographic algorithms, which are used in countless software (and hardware) systems.

In this paper, we studied two fundamental problems in the area of computational complexity. The notion of MC which minimizes the number of elementary non-linear operations (AND gates), and more general problems of gate complexity under almost any circuit complexity "metric", for example MC, gate count w.r.t. a specific set of gates, circuit depth, circuit width, etc. Following the heuristic of Boyar-Peralta, we used MC as an essential tool for optimizing potentially arbitrary algebraic computations over fields and rings and in particular for binary circuits. Additionally, we focused on the combinatorial logic optimization of general digital circuits with particular attention to small substitution boxes (S-boxes), which are massively used in the industrial cryptographic schemes and which are small enough for some very advanced methods to be invented. Thus, in many cases we managed to obtain optimal results which can no longer be improved, and we have a formal mathematical proof of these claims by an automated software proof technique with a SAT solver.

We have developed a fully automated procedure for obtaining new formulas for Matrix Multiplication (MM) problems, complex multiplication problems, multiplication of quaternions and for construction of optimal circuit representations with any given Boolean functions, with respect to any given set of basic gates. Our methodology consists of three main steps. In the first step, we formally encode these problems as polynomial equations, then convert them into a SAT problem using the Courtois-Bard-Jefferson [7] or other methods and then we solve these problems using SAT solver software. Thus, we have been able to find new formulas for multiplying two 3×3 matrices using 23 2-input multiplications [11] and also multiplying a 2×2 matrix by a 2×3 matrix using only 11 multiplications, naive multiplication needs 12. We have been able to construct several optimal circuit representations for the S-box of CTC2 cipher [3] with respect to its MC, Bitslice Gate Complexity, Gate Complexity and NAND Gate Complexity. Additionally, we constructed an optimal circuit representation with 14 gates for the PRESENT S-box, which is the best currently known [15] and we computed the exact MC of the 8 S-boxes S1-S8 used in the GOST cipher. The amazing thing is that our methodology can find EXACT circuit representations (which is very hard to obtain in the area of computational complexity). This is if our SAT solvers used in the final solving stage are complete (in a sense that they are fast enough and output UNSAT if the problem has no solution). In future works, we need to address the questions of what form or language such automated proofs could be output, shared and published.

Cryptography is always very costly, and a lot of effort is always done in order to improve the implementation of any given cipher [5], [6]. To the best of our knowledge

provably optimal circuits have never been found before for any cryptographic algorithm, and optimal digital circuits have never been yet used in industrial applications. Interestingly, if a cryptographic algorithm such as AES which tends to be present nowadays in more or less every major CPU [5] could be implemented in a provably optimal way (or at least it would be optimal for smaller components like in the present paper), people in the industry would no longer need to make their designs proprietary. The best digital designs could be developed at universities and licensed or offered to the worldwide industry or jointly developed by cooperative industrial consortia to benefit every company small and large. Thus, the (large) computational cost and research effort needed to develop such top-end (excessively good) implementations and optimizations could be amortized and justified.

ACKNOWLEDGMENT

This research was supported by the UK Technology Strategy Board in the United Kingdom under project 9626-58525.

REFERENCES

[1] N.T. Courtois, D. Hulme, and T. Mourouzis, "Multiplicative Complexity And Solving Generalized Brent Equations With SAT Solvers," in COMPUTATION TOOLS 2012, in the Third International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking, pp. 22-27, 2012.

[2] J. Patarin, N. Courtois, and L. Goubin, "Improved Algorithms for Isomorphism of Polynomials," in Advances in CryptologyEUROCRYPT'98, pp. 184-200, Springer Berlin Heidelberg, 1998.

[3] N.T. Courtois, D. Hulme, and T. Mourouzis, "Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis," in Proceedings of 2nd IMA Conference Mathematics in Defence, UK, Swindon, 2011.

[4] N.T. Courtois, D. Hulme, and T. Mourouzis, "Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis." in IACR Cryptology ePrint Archive 2011, Report 475,2012.

[5] J. Boyar, P. Matthews, and R. Peralta, "Logic Minimization Techniques with Applications to Cryptology," Journal of Cryptology, vol. 26, p. 280-312, 2013.

[6] J. Boyar, and R.Peralt, "A depth-16 circuit for the AES S-box," in IACR Cryptology ePrint Archive, Report 33,2011.

[7] G.V. Bard, N.T. Courtois, and C. Jefferson, "Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers, "ECRYPT workshop Tools for Cryptanalysis, 2007.

[8] N. Sorensson, and N. Een, "Minisat v1. 13-a sat solver with conflict-clause minimization," SAT journal pp. 53-59, 2005.

[9] R. Brent, "Algorithms for matrix multiplication," Tech. Report Report TR-CS-70-157 , Department of Computer Science, Stanford, 1970.

[10] C. Fuhs, and P. Schneider-Kamp, "Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT," in SAT 2010, Theory and Applications of Satisfiability Testing, Springer LNCS 6175, pp. 71-84, 2010.

[11] N.T. Courtois, G.V. Bard, and D. Hulme, "A New General-Purpose Method to Multiply 3x3 Matrices Using Only 23 Multiplications," in arXiv preprint arXiv:1108.2830, 2011.

[12] N. T. Courtois, and G. Bard, "Algebraic Cryptanalysis of the Data Encryption Standard," in Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007.

[13] N.T. Courtois, "Algebraic Complexity Reduction and Cryptanalysis of GOST," in IACR Cryptology ePrint Archive, Report 626, 2011.

[14] A. Edelman, "Large Dense Numerical Linear Algebra in 1994 (survey)," Journal of Supercomputer Applications. Vol. 7, p. 113128, 1993.

[15] M. Albrecht, N.T. Courtois, D. Hulme, and G. Song, "Bit-Slice Implementation of PRESENT in pure standard C," Available online at www.nicolascourtois.com, 2011.

[16] E. Prouff, C. Giraud, and S. Aumonier, "Provably Secure S-Box Implementation Based on Fourier Transform," in CHES 2006, Springer LNCS 4249, pp. 216-230, 2006.

[17] G. Bard, "Algorithms for Solving Linear and Polynomial Systems of Equations over Finite Fields with Applications to Cryptanalysis," Submitted in Partial Fulfillment for the degree of Doctor of Philosophy of Applied Mathematics and Scientific Computation, 2007.

[18] M. Soos, "CryptoMiniSat 2.5.0,"in SAT Race competitive event booklet, 2010.

[19] S. Dasgupta, C. Papadimitriou, and U. Vazirani, "Algorithms," 2nd Edition, 2006.

[20] V. Strassen, "Gaussian elimination is not optimal," in Numerische Mathematik Vol 13 pp. 354-356, 1969.

[21] D. Coppersmith, and S.Winograd, "On the asymptotic complexity of matrix multiplication," SIAM Journal Comp., Vol 11, pp 472-492 , 1980.

[22] J.D. Laderman, "A Non-Commutative Algorithm for Multiplying 3x3 Matrices Using 23 Multiplications," in Amer. Math. Soc. Vol. 82, Number 1, 1976.

[23] H. Cohn, R. Kleinberg, B. Szegedyz, and C. Umans, "Grouptheoretic Algorithms for Matrix Multiplication," in FOCS05, 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 379-390, 2005.

[24] J. Boyar, and R. Peralta, "A New Combinational Logic Minimization Technique with Applications to Cryptology," in SEA 2010, pp. 178-189, 2009.

[25] J. Boyar, P. Matthews, and R. Penalta, "On the Shortest Linear Straight-Line Program for Computing Linear Forms," in Mathematical Foundations of Computer Science 2008, pp. 168-179, Springer Berlin Heidelberg, 2008.

[26] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, and M.J.B. Robshaw,"PRESENT: An Ultra-Lightweight Block Cipher," in CHES 2007, LNCS 4727, pp. 450-466, Springer, 2007.

[27] A. Poschmann, S. Ling, and H. Wang, "256 Bit Standardized Crypto for 650 GE GOST Revisited," in CHES 2010, LNCS 6225, pp. 219-233, 2010.