Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis*

Nicolas T. Courtois[1,2], Daniel Hulme[1,2], and Theodosis Mourouzis[1]

[1] University College London, UK,
[2] NP-Complete Ltd, London, UK

Abstract. One of the hardest problems in computer science is the problem of gate-efficient implementation. Such optimizations are particularly important in industrial hardware implementations of standard cryptographic algorithms [13, 17, 7, 22]. In this paper we focus on optimizing some small circuits such as S-boxes in cryptographic algorithms. We consider the notion of Multiplicative Complexity, a new important notion of complexity studied in 2008 by Boyar and Peralta and applied to find interesting optimizations for the S-box of the AES cipher [19, 22, 21]. We applied this methodology to produce a compact implementation of several ciphers. In this short paper we report our results on PRESENT and GOST, two block ciphers known for their exceptionally low hardware cost. This kind of representation seems to be very promising in implementations aiming at preventing side channel attacks on cryptographic chips such as DPA. More importantly, we postulate that this kind of minimality is also an important and interesting tool in cryptanalysis.

Key Words: Block ciphers, PRESENT, GOST, non-linearity, algebraic attacks, circuit complexity, logic-level optimization, multiplicative complexity, algebraic cryptanalysis, block cipher implementation, bitslice implementation, side-channel attacks, DPA.

* This research was supported by the European Commission under the FP7 project number 242497 ”Resilient Infrastructure and Building Security (RIBS)” and by the UK Technology Strategy Board under project 9626-58525.

Note: A short and early version of this paper was included in the electronic proceedings of 2nd IMA Conference Mathematics in Defence 2011, 20 October 2011, Defence Academy of the United Kingdom, Swindon, UK.

1 Introduction

The problems of circuit complexity is one of the hardest and yet very important problems in computer science and complexity theory. For the great majority of concrete circuits, it is not known what will be the lowest bound on their complexity, neither how to compute very good circuits efficiently. Not everybody in the industry cares about improving their gate count by a small factor, but such optimizations are particularly important in hardware implementation of standard cryptographic algorithms [13, 17, 7, 22], which in many security chips such as smart cards and RFID, will be one of the most costly components. Here even a small gain can produce measurable savings.

Many heuristic algorithms for this problem have been invented, and with a lot of computing power one can find very decent optimizations [13], but these optimizations are frequently subject to further substantial improvement. In this paper we particularly focus on optimizing the S-boxes for industrial block ciphers.

Much less known and very surprising is that this is also an important topic in cryptanalysis. As shown in [7, 10, 8] such optimizations are also very important in order to speed up so called algebraic attacks on symmetric ciphers, and in the space of attacks which require very small quantities of data, these methods lead to currently best known attacks on a few ciphers (with more data, typically faster attacks will exist). In this paper we focus mostly on 4x4 S-boxes in ciphers such as PRESENT and GOST. These ciphers are known for their exceptionally low hardware implementation cost [17]. But this is also what makes them vulnerable to algebraic cryptanalysis.

Sometimes the very existence of an attack on the cipher which would be faster than brute force will depend on a concrete circuit optimization problem, precisely because the time complexity must be fast enough to beat the brute force attacks. Our work on cryptanalysis makes extensive use of SAT solver software, both at the optimization stage, when a "compact algebraic description" of a cipher is produced, and a later solving stage, where the equations are solved to in order to compute the secret key.

2 S-box Optimization

In 2008 Boyar and Peralta introduced a new heuristic methodology to minimize the complexity of digital circuits [19, 22, 21]. It is based on the notion of Multiplicative Complexity (MC).

Multiplicative Complexity (MC) is a well-known and very deep notion of arithmetic complexity invariant w.r.t. affine transformations, which minimizes the number of non-linear elementary transformations, see [25, 18, 19]. Their main heuristics is that a two step-process based on MC appears to be able to produce very good gate efficient implementation of several famous circuits such as the AES S-box, and some other circuits related to finite fields and algebra, Several such results can be found in [22, 21]. In this paper we apply this methodology to some cryptographically significant functions $GF(2)^4 \rightarrow GF(2)^4$ (i.e. 4x4 S-boxes). We developed software which allows us to compute **optimal** representations of these S-boxes w.r.t to this methodology.

2.1 Motivations For Achieving Low-MC and Low Gate Count

There are several good mains reasons why we want to determine and improve the complexity of various circuits, whether we consider Multiplicative Complexity (MC), or other complexity measures.

1. Lower the implementation cost in silicon.
2. Prevent Side Channel attacks such as DPA. this is due to the fact the XORs are believed easy to protect against DPA through linear secret charing techniques. Then minimizing the number of AND gates is expected to lower

the cost of general-purpose protections against side channel attacks which are developed to securely implement arbitrary digital circuits, such as for example developed in [16].

3. Algebraic Cryptanalysis of a symmetric cipher can be greatly improved if we use gate-efficient and compact representations, as demonstrated in [7, 8, 10]. This usually works only for cipher with a limited number of rounds. Then additional non-trivial higher-level "tricks" are needed to be able to really break a full cipher with many more rounds, see [8, 10, 11].
4. In symbolic computing and numerical algebra, this kind of optimizations can be applied recursively to produce asymptotically fast algorithms to solve very famous and important practical problems such as Gaussian reduction and matrix multiplication, see [5].

2.2 Bitslice Gate Complexity and Multiplicative Complexity

In this section we define two particular models for gate complexity of digital circuits.

Definition 2.2.1 (Bitslice Gate Complexity (BGC)).
Given a function $GF(2)^n \rightarrow GF(2)^m$ we define its Bitslice Gate Complexity (BGC) as the **minimum** number of 2-input gates of types XOR,OR,AND,OR needed.

Note: we do NOT allow gates of type NOR and NAND. This is a very simple model, in which the cost of all these gates is considered to be the same, and which is relevant for example in so called Bit-slice implementations of block ciphers, such as for example in [1]. However it is not an optimal model for silicon implementations, where certain gates are more costly to implement, while NOR and NAND gates are actually less costly.

Now we recall the definition of MC [25, 18, 19, 22]:

Definition 2.2.2 (Multiplicative Complexity (MC)).
Given a function $GF(2)^n \rightarrow GF(2)^m$ we define its Multiplicative Complexity (MC) as the minimum number of AND gates which need to be used to implement this function, with an unlimited number of NOT and XOR gates.

This model considers that linear operations come "for free" and ask to minimize just the number of AND gates. The problem with Bitslice Gate Complexity (BGC) is that we are not in general able to determine its value, algorithms which find such optimizations are typically random stochastic explorations of large trees of solutions [13] and we are not sure if the optimizations are final or if they can still be improved. However, as we will see in this paper, at least for small circuits, the Multiplicative Complexity (MC) can be computed **exactly** by our methods which use SAT solver software.

We have also the following fact which results directly from the definition:

Fact 1. The Multiplicative Complexity is invariant w.r.t. to multivariate affine transformations at the input and at the output. As a consequence, for a 4x4 bit S-box, its Multiplicative Complexity is the same for the whole affine equivalence class, which classes are studied in [15, 23].

2.3 Multiplicative Complexity As A Tool For Gate Complexity

Boyar and Peralta have a developed a heuristic methodology, where they optimise for of Multiplicative Complexity (MC) in order to produce also gate-efficient implementations:

1. **(Step 1)** First compute the multiplicative complexity.
2. **(Step 2)** Then optimise the number of XORs separately, see [20, 12].
3. Optional Step 3: At the end do additional optimizations to decrease the circuit depth, an possibly additional software optimizations, see [19, 22],

This methodology was then used to produce new worldwide records in gate efficient implementation of several famous circuits such as the AES S-box, and many other circuits related to finite fields and algebra, [22, 21, 21].

2.4 Our Method to Compute the Multiplicative Complexity

In this paper we focus on optimisation of functions $GF(2)^4 \rightarrow GF(2)^4$ which are immensely popular in cryptography [23]. We have implemented fully and with our own optimisation methods, both Steps 1. and 2. above.

The crucial feature of our implementation is that BOTH our Steps 1. and 2. are OPTIMAL, i.e. they produce the best possible optimizations which can be obtained by following these two steps. Optimality was achieved due to SAT solver software, we convert our problem to SAT and it either outputs SAT, and a solution, which we convert to a concrete circuit optimization, or it outputs UNSAT, and we are certain that there is no solution. There is third possibility, that the SAT solver software runs for a very long time and we do not have enough computing power to decide whether the result is SAT or UNSAT, but this have never happened for 4x4 S-boxes. Accordingly, we were able to produce optimal optimizations or this type for every 4x4 S-box we have ever tried. This is very rare in complexity: to be able to completely determine the best possible result.

We must say that these methods are at prototyping stage and they are so far slower than other known methods [13]. Likewise, we do not claim that we can optimise the linear parts as quickly as by recent methods described in [19, 20], but only that we can optimize to the strictest minimum possible, which probably can also be achieved in [12] by similar methods and SAT solvers. This is for linear circuits. However it seems that we are the first to apply SAT solvers also to optimize non-linear circuits.

We have also obtained some very good results on bi-linear circuits, see [5].

2.5 Provable Aspects of Our Method

Our solutions are optimal and thus proven to be impossible to improve (automated software proof with UNSAT). This is they would be provably optimal, if we had a proof of correctness of the SAT solver software.

It will also be correct, but not proven correct, if there is no bug in the SAT solver software. Such a bug, where a problem which is SAT is claimed to be UNSAT by another solver, will quickly and easily be found, because we have a portfolio of many different SAT solver software, and regularly check these results by at least a few SAT solvers. Even if we assume the presence of bugs in this software, one can consider that our proofs are "probabilistic proofs", but still the probability of error can be easily made as small as desired.

Thus we achieve a proof of impossibility when our program outputs UNSAT for smaller sizes. 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. However what is missing is also a proof that all our conversions are correct and preserve correctness. This can be done in future research. Such proofs would not be published in scientific papers, but rather as lengthy computer files, which should come together with a formal system able to efficiently check the correctness of such proofs. This is a major topic for further research which would require one to develop a whole new formal language and software to manipulate it.

2.6 An Alternative Method to Compute the Multiplicative Complexity

We can note that for Step 1, and only for 4x4 S-boxes, there is a simple and alternative method to compute the of Multiplicative Complexity (MC), in step 1, following the work on classification and equivalence of 4x4 S-boxes [15, 23]. It is as follows:

1. Determine another S-box for which our S-box is an affine equivalent of another S-box, for which the MC was already computed.
2. The affine equivalence can be determined by methods of [2] which are actually essentially the same methods which have been proposed at the same conference 10 years earlier [4] in a slightly different context.

3 Optimizing the PRESENT S-box

The PRESENT S-box is defined as $\{12, 5, 6, 11, 9, 0, 10, 13, 3, 14, 15, 8, 4, 7, 1, 2\}$. We will number the least significant bits starting from 1.

Theorem 3.0.1. The Multiplicative Complexity of the PRESENT S-box is exactly 4.

Proof: For 3 AND gates our thoroughly designed and tested system outputs UNSAT. This could be converted to a formal proof that the Multiplicative Complexity is at least 3. We have obtained an automated proof of this fact which takes a few seconds on a PC and can reproduced and checked. For 4 AND gates, our system outputs SAT and a solution. Further optimisation 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 example we have obtained an implementation of the PRESENT S-box with 25 gates, 4 AND, 20 XOR, 1 NOT which is optimal w.r.t our Boyar-Peralta 2-step methodology but not optimal in overall gate complexity. 25 gates are still not very satisfactory.

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 gate 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 (!):

```
T1=X2^X1; T2=X1&T1; T3=X0^T2; Y3=X3^T3; T2=T1&T3; T1^=Y3; T2^=X1;
T4=X3|T2; Y2=T1^T4; T2^=~X3; Y0=Y2^T2; T2|=T1; Y1=T3^T2;
```

Applications. This implementation is used in our recent bit-slice implementation of PRESENT, see [1]. 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 does **not** contradict the Boyar-Peralta heuristic to the effect that some of the best possible gate-efficient implementations are very closely related to the notion of multiplicative complexity. 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 Fig 1. in [22].

4 The GOST S-boxes

We consider the main standard and most widely known version of the GOST block cipher, known as "GostR3411_94_TestParamSet" in [14]. and also known as the one used by the Central Bank of the Russian Federation [17]. By running the same method and programs we obtained the following result:

Theorem 4.0.2. The Multiplicative Complexity of the eight GOST S-boxes S1,S2,S3,S4,S5,S6,S7,S8 is exactly equal to respectively 4,5,5,5,5,5,4,5.

Related Work: We can compare this to the results in Table on page 226 of [17] where we see that these 8 S-boxes are also on average more expensive than the PRESENT S-box in the sense of Gate Equivalent (GE) cost, (the GOST S-boxes cost 23.5 GE on average per S-box while the PRESENT S-box appears to require about 27 GE). In Table 3 in [17] we see that PRESENT S-box is better against linear and differential cryptanalysis. However in our Multiplicative Complexity (MC) metric, in our Bitslice Gate Multiplicative Complexity (BGC) metric, and also in the strict GE cost metric in [17], it is clear that the complexity of the PRESENT S-box is always lower and therefore we conjecture that PRESENT S-box will be weaker than the GOST S-boxes, against many types of algebraic cryptanalysis such as attacks described in [10, 11]. Thus it is probably a bad idea to use the GOST cipher with PRESENT S-boxes as proposed in [17].

4.1 Additional Standard GOST S-boxes

Remark: In the future works we will publish much more results for all the 64 known GOST S-boxes and their inverses, and also other optimizations of these S-boxes, and also the exact application of these results in cryptanalysis. The table below contains some preliminary results.

Table 1. Multiplicative Complexity 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

We believe that this table gives some first and early indications which versions of GOST will be more secure against algebraic cryptanalysis, this however requires much more extra work.

4.2 Affine Equivalence and Quality of GOST S-boxes

The S-boxes of many ciphers have been carefully chosen to be very good with respect to linear and differential cryptanalysis. Researchers who have studied 4-bit S-boxes [15, 23] have found that there only 16 'optimal' S-boxes w.r.t. differential and linear attacks, plus their affine/linear equivalents. Many other well-known S-boxes such as the PRESENT S-box studied here, all the 8 S-boxes in Serpent cipher and all their 8 inverses, and many other, are affine equivalents of one of these 16 optimal S-boxes, see [15, 23].

In contrast, very few of the known GOST S-boxes are affine equivalents of these 16 optimal S-boxes. In the following table we give for each GOST S-box and each inverse S-box its affine equivalence class.

The equivalence class is either

1. It can be of the type G_i with $i = 0..16$ for equivalents of so called 'optimal' S-boxes from [15],
2. We denote by Lu1 the second Lucifer S-box S1, see [23].

Table 2. Affine equivalence of known GOST S-Boxes and their inverses

S-box Set Name	$S1$	$S2$	$S3$	$S4$	$S5$	$S6$	$S7$	$S8$
GostR3411_94_TestParamSet	36	02	03	04		06	35	08
- their inverses		02	03	04		06		08
GostR3411_94_CryptoProParamSet			$Lu1$	14	G_{10}		G_8	
- their inverses			$Lu1$	14	G_{10}		G_8	
Gost28147_TestParamSet	21	21			25			28
- their inverses	21	21			25			28
Gost28147_CryptoProParamSetA	31	32	33	G_8	35	36	37	38
- their inverses	31	32	33	G_8			37	38
Gost28147_CryptoProParamSetB	G_{13}	G_{13}	G_{13}	G_{11}	G_7	G_7	G_{11}	G_6
- their inverses	G_{13}	G_{13}	G_{13}	G_{11}	G_7	G_7	G_{11}	G_6
Gost28147_CryptoProParamSetC	G_7	G_4	G_6	G_{13}	G_{13}	G_6	G_{11}	G_{13}
- their inverses	G_7	G_4	G_6	G_{13}	G_{13}	G_6	G_{11}	G_{13}
Gost28147_CryptoProParamSetD	G_{13}	G_{13}	G_{13}	G_4	G_{12}	G_4	G_{13}	G_7
- their inverses	G_{13}	G_{13}	G_{13}	G_4	G_{12}	G_4	G_{13}	G_7
GostR3411_94_SberbankHashParamset			74	75	76		78	
- their inverses			74	75	78		76	

3. or it is an arbitrary string of integers, where we use the same string only where we want to signify that we have an equivalent S-box elsewhere in the same table (these string so of integers are purely conventional and have no meaning outside this table).

We observe that many GOST S-boxes, especially the first set, which is the oldest set of GOST S-boxes known, are equivalent to their own inverses.

This table also shows very clearly that there was a historical evolution of GOST S-boxes towards boxes of type G_i which are optimal against linear and differential cryptanalysis [15]. Most of more recent S-boxes which appear in OpenSSL [14] are one of the G_i. Following [15] 12 out of these 'optimal' S-boxes are self-equivalent. Only 8 of these 12, namely $G_4, G_6, G_7, G_8, G_{10}, G_{11}, G_{12}, G_{13}$, occur in our table for GOST.

5 Multiplicative Complexity of Whole Ciphers

It appears that, from here we are able to **provably minimize** the number of non-linear gates in a whole given cipher, to a proven lower bound.

In order to do this we need to look at any other existing non-linear components of the cipher, and also compute their Multiplicative Complexity (MC).

Then we also need to prove that the Multiplicative Complexity is not reduced by the combination.

Such a reduction is not always very likely, but if it occurs, it could be considered as a potential structural flaw in the cipher. It could be seen as a sign that somewhat the designers have maybe "wasted" the computational resources in hardware, for a given security level. Alternatively, it could also be a source of potential shortcuts to implement the cipher more efficiently.

6 Multiplicative Complexity of Whole GOST Cipher

We would like to minimize the number of non-linear gates in the whole given cipher. We sketch how this can be done for the GOST block cipher. We basically need to compute the Multiplicative Complexity (MC) for each component and add them.

6.1 Modular Addition

In addition to S-boxes, the GOST cipher uses addition modulo 2^{32}. The interesting question is what is the multiplicative complexity of this operation.

In order to optimize this addition modulo 2^{32} we follow the first method described in [9]. Let us consider three n-bit words $(x_{n-1}, \ldots, x_0)$, $(y_{n-1}, \ldots, y_0)$ and $(z_{n-1}, \ldots, z_0)$ with z_0 being the low-order bit. The modular addition

$$(x, y) \mapsto z = x \boxplus y \mod 2^n$$

can be described the following way by $(*)$ and $(*')$, using new variables that are carry bits, represented by the $(n-1)$-bit word $c = (c_{n-1}, \ldots, c_1)$:

$$(*) \begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + c_1 \\ z_2 = x_2 + y_2 + c_2 \\ \vdots \\ z_i = x_i + y_i + c_i \\ \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + c_{n-1}, \end{cases} \qquad (*') \begin{cases} c_1 = x_0 y_0 \\ c_2 = x_1 y_1 + (x_1 + y_1) c_1 \\ \vdots \\ c_i = x_{i-1} y_{i-1} + (x_{i-1} + y_{i-1}) c_{i-1} \\ \vdots \\ c_{n-1} = x_{n-2} y_{n-2} + (x_{n-2} + y_{n-2}) c_{n-2} \end{cases}$$

We claim that:

Theorem 6.1.1. The Multiplicative Complexity (MC) of the addition modulo 2^n is exactly $n - 1$.

Proof: This is is not obvious at the first sight, it may seem that it is $2(n-1)$. However in characteristic 2 we have:

$$xy + (x + y)c = (x + c)(y + c) + c$$

which allows to reduce the number of multiplications to 1 in each line: we obtain $(x_{i-1} + c_{i-1})(y_{i-1} + c_{i-1}) + c_{i-1}$. Thus we have established it is at most $n - 1$.

To prove it is at least $n - 1$ we observe that the algebraic degree of the ANF of the last output bit z_{n-1} as a function of the x_i and the y_i is always $n-1$. This is easy to see from the formulas because each new carry c_i contains a multiplication of the previous carry c_{i-1} by new independent variables. Therefore the ANF degree of z_{n-1} is $n - 1$ and at least $n - 1$ multiplications are needed to compute it, and therefore at least $n - 1$ multiplications are needed overall.

6.2 Application to Cryptanalysis of GOST

It appears that we are able to **provably minimize** the number of non-linear gates in a whole given cipher such as GOST, to a proven lower bound. In future works we will show how this type of optimization is used to break the full-round block cipher GOST, see [10, 11].

Currently no theory is able to give recommendations about how to produce the fastest algebraic attack on a given cipher, and there are many competing techniques, see [7]. We conjecture that the possibility to reduce the Multiplicative Complexity (MC) of the whole cipher to the lowest possible number, and also other metrics of circuit complexity, will play an important role in finding the best possible attacks in algebraic cryptanalysis.

7 Conclusion

In this paper we explore some ideas recently proposed by Boyar and Peralta to optimize the AES S-box [19, 22] in order to see if they can be applied to other cryptographic S-boxes. They central notion is that of Multiplicative Complexity (MC) which minimizes the number of elementary non-linear operations (AND gates) at the cost of linear operations, which can be optimized separately, as a second step. We have implemented both these steps in an innovative way, where each problem is converted to a satisfiability problem and solved by SAT solver software. This type of methodology was previously applied to optimize linear circuits [12] and bi-linear circuits [5] but it appears it is for the first time it is used to optimise circuits of arbitrary algebraic degree. The key interesting point is that many SAT solvers will be able to detect when the problem is not solvable, leading to results which are proven to be optimal, a rare thing in complexity.

Thus we are able to compute Multiplicative Complexity (MC) **exactly**, for all sufficiently small circuits, and also to optimize the linear parts exactly. Our method is practical though rather slow, so far we have been able to optimize every 4x4 S-box we tried, but not many larger S-boxes. Yet it is a unique and very powerful method, because all the results are optimal and one could produce and publish a formal mathematical proof (automatically found by the software) that they cannot be improved.

Furthermore we have applied this notion to derive efficient implementations of the S-boxes in two ciphers, PRESENT and GOST. In the case of PRESENT it happens that the Boyar-Peralta heuristics [19, 22] works extremely well, and the best possible gate-efficient optimization we could find also contains the (optimal) lowest possible number of non-linear gates(!). However GOST S-boxes have on average higher Multiplicative Complexity (MC) and yet lower implementation cost, so this heuristics is unlikely to be always the best method to optimise a circuit. Clearly better optimizations are likely to use a few more non-linear gates, as also seen for AES, cf. Fig 1 in [22].

Interestingly, from here we are able to **provably minimize** the number of non-linear gates in a whole given cipher such as PRESENT or GOST, to a rather unexpectedly low number such as 5 per S-box. This has two sorts of applications in cryptography. First, such optimizations are important in synthesis of implementations of circuits secure against side-channel attacks, which is an important and hot research topic, see for example [16].

Moreover, in future works we will show how S-box optimizations greatly help to break the full-round block cipher GOST and its many variants [14, 17]. It is extremely rare to see a real-life block cipher which can be broken faster than brute force. This however requires a lot of additional work, see [10, 11].

In particular, we have seen that PRESENT S-box has, and this on multiple accounts, a lower complexity than many of the GOST S-boxes. Thus we expect that future research will show, that against many types of algebraic cryptanalysis such as in [10, 11], the PRESENT S-box is strictly weaker than majority of GOST S-boxes. Therefore it is probably a bad idea to use the GOST cipher with PRESENT S-boxes as proposed in [17].

References

1. Martin Albrecht, Nicolas T. Courtois, Daniel Hulme, Guangyan Song: *Bit-Slice Implementation of PRESENT in pure standard C,* v1.5, 26/08/2011, open-source code available at `https://bitbucket.org/malb/algebraic_attacks/src/tip/present_bitslice.c`
2. A. Biryukov, C. De Cannière, A. Braeken, and B. Preneel: *A toolbox for cryptanalysis: Linear and affine equivalence algorithms.* In Eurocrypt 2003, LNCS 2656, pp. 3350, Springer, 2003.
3. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe: *PRESENT: An Ultra-Lightweight Block Cipher,* In CHES 2007, LNCS 4727, pp. 450466, Springer, 2007.
4. Jacques Patarin, Nicolas Courtois, Louis Goubin: *Improved Algorithms for Isomorphism of Polynomials*; Eurocrypt'98, LNCS 1403, Springer, pp.184-200,
5. Nicolas T. Courtois, Gregory V. Bard and Daniel Hulme: A New General-Purpose Method to Multiply 3x3 Matrices Using Only 23 Multiplications, At `http://arxiv.org/abs/1108.2830`.
6. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4, LNCS 3373, pp. 67-83, Springer, 2005.
7. Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007. Preprint available at `eprint.iacr.org/2006/402/`.
8. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
9. Nicolas Courtois and Blandine Debraize: *Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0.,* In ICICS 2008, 10th International Conference on Information and Communications Security, 20 - 22 October, 2008, Birmingham, UK. In LNCS 5308, pp. 328-344, Springer, 2008.
10. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* Preprint available at `http://www.nicolascourtois.com/papers/gostac11.pdf`.
11. Nicolas Courtois: *Security Evaluation of GOST 28147-89 In View Of International Standardisation,* document officially submitted to ISO in May 2011, At `http://eprint.iacr.org/2011/211/`.
12. Carsten Fuhs and Peter 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.
13. B. R. Gladman, software for efficient boolean function decompositions for the eight Serpent S boxes and their inverses, available at `http://gladman.plushost.co.uk/oldsite/cryptography_technology/serpent/index.php`.
14. A Russian reference implementation of GOST implementing Russian algorithms as an extension of TLS v1.0. is available as a part of OpenSSL library. The file gost89.c contains eight different sets of S-boxes and is found in OpenSSL 0.9.8 and later: `http://www.openssl.org/source/`
15. Gregor Leander, Axel Poschmann: *On the Classification of 4 Bit S-Boxes, In Proceedings of WAIFI'07, 1st international workshop on Arithmetic of Finite Fields.*
16. *Svetla Nikova, Vincent Rijmen, Martin Schläffer: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches,* Special Issue on Hardware and Security of Journal of Cryptology, 27 pages, 2011. `http://homes.esat.kuleuven.be/~snikova/JOC_2011.pdf`

17. Axel Poschmann, San Ling, and Huaxiong Wang: *256 Bit Standardized Crypto for 650 GE GOST Revisited,* In CHES 2010, LNCS 6225, pp. 219-233, 2010.
18. Joan Boyar, René Peralta, Denis Pochuev: *On the multiplicative complexity of Boolean functions over the basis (AND, XOR, 1),* In Theor. Comput. Sci. 235(1): 43-57 (2000).
19. Joan Boyar, René Peralta: *A New Combinational Logic Minimization Technique with Applications to Cryptology.* In SEA 2010: 178-189.
    An early version was published in 2009 at `http://eprint.iacr.org/2009/191`. It was revised 13 Mar 2010.
20. Joan Boyar, Philip Matthews, René Peralta: *On the Shortest Linear Straight-Line Program for Computing Linear Forms,* In MFCS 2008: 168-179.
21. Web page with all circuit minimialisation results obtained at Yale University, `http://cs-www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html`.
22. Joan Boyar and Rene Peralta; *A depth-16 circuit for the AES S-box,* `http://eprint.iacr.org/2011/332`
23. Markku-Juhani O. Saarinen: *Cryptographic Analysis of All 4 x 4 - Bit S-Boxes,* In SAC 2011, August 2011 Toronto, Canada, Springer LNCS. A version is available at `eprint.iacr.org/2011/218/`.
24. I. Schaumuller-Bichl: *Cryptanalysis of the Data Encryption Standard by the Method of Formal Coding,* In Cryptography, Proc. Burg Feuerstein 1982, LNCS 149, T. Beth editor, Springer-Verlag, 1983.
25. Claus-Peter Schnorr: *The Multiplicative Complexity of Boolean Functions,* In Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, AAECC-6, LNCS 357, pp. 45-58, 1988.