VERSITA

TATRA MOUNTAINS
Mathematical Publications
DOI: 10.2478/v10127-012-0039-3
Tatra Mt. Math. Publ. **53** (2012), 65–79

CONTRADICTION IMMUNITY AND GUESS-THEN-DETERMINE ATTACKS ON GOST

Nicolas T. Courtois — Jerzy A. Gawinecki — Guangyan Song

ABSTRACT. GOST is a well-known government standard cipher. Since 2011 several academic attacks on GOST have been found. Most of these attacks start by a so called "Complexity Reduction" step [Courtois Cryptologia 2012] the purpose of which is to reduce the problem of breaking the full 32-round GOST to a low-data complexity attack on a reduced-round GOST. These reductions can be viewed as optimisation problems which seek to maximize the number of values inside the cipher determined at given "cost" in terms of guessing other values. In this paper we look at similar combinatorial optimisation questions BUT at the lower level, inside reduced round versions of GOST.

We introduce a key fundamental notion of **Contradiction Immunity** of a block cipher. A low value translates to working software attacks on GOST with a SAT solver. A high value will be mandatory for any block cipher to be secure. We provide some upper bounds for the Contradiction Immunity of GOST.

1. The GOST encryption standard

The Russian encryption standard GOST 28147–89 is an important government standard [18]. Its large key size of 256 bits make GOST a plausible alternative for AES-256 and 3-key triple DES. Clearly GOST is a serious cipher for serious applications and at least two sets of GOST S-boxes have been explicitly identified as being used by the most prominent Russian banks, cf. [19], [26].

The most complete current reference implementation of GOST in OpenSSL library contains eight standard sets of S-boxes [19]. The attacks we consider in this paper, work with a very similar complexity for any S-boxes.

2010 Mathematics Subject Classification: 94A60, 68P25, 90C27.
Keywords: block ciphers, cryptanalysis, GOST, DES, low-data complexity, guess-then-determine, meet in the middle, combinatorial optimization, SAT Solvers.
Partly supported by the Polish Ministry of Science as a project n0. 0 R00 0111 12 in 2012.

1.0.1. GOST and ISO standardisation

The cost of cryptography is still an important problem for the industry, for example only around 2010 Intel implemented an encryption algorithm in some of its CPUs, and not yet in all of its CPUs. It is therefore very important to notice that in addition to the very long bit keys GOST has a much lower implementation cost than AES or any other comparable encryption algorithm. For example in hardware GOST 256 bits requires less than 800 GE, while AES-128 requires 3100 GE, see [22]. Thus it is not surprising that GOST became an Internet standard, it is part of many crypto libraries such as OpenSSL [20], and is increasingly popular also outside its country of origin [23]. Hard to think about a better algorithm for the industry with its ultra-low implementation cost and 20 years of cryptanalysis efforts behind it [22]. In 2010 GOST was submitted to ISO 18033 to become a worldwide encryption standard. Less than 10 block ciphers have ever become an ISO standard. Unhappily in 2011 several key recovery attacks on GOST have been found [8], [10], [12], [14], [20].

1.0.2. Cryptanalysis of GOST

The turning point in the security of GOST was the discovery of the so called "Reflection" property [21]. Initially at Indocrypt 2008 only a weak-key attack with time complexity of 2^{192} is proposed, with large proportion of 2^{-32} of weak keys. Then in 2011 several attacks on regular GOST keys have been discovered, and more than half of these new attacks use this reflection property [16], [20] sometimes twice, three or four times [10]. Most these attacks can be described as attacks with a "Complexity Reduction" [8], [10] where from some data for the full 32 rounds GOST we obtain a certain number of pairs for 8 or less rounds of GOST. The quantity of data available after reduction is very small, for example 2, 3 or 4 pairs for a reduced cipher. In this paper we look precisely at questions pertaining to cryptanalysing 8 rounds GOST with 2, 3, 4 KP needed as a last step in numerous already known attacks on GOST [8], [9], [10].

These reductions can be viewed as optimisation problems which seek to maximize the number of values determined about some values inside the cipher which can be obtained by guessing some other values at given "cost". In this paper we consider similar optimisation questions of finding a possibly optimal guess-then-determine attack BUT at the lower level, inside reduced round versions of GOST. We postulate that there should be a phase transition between hard and easy "determine" problems, and that one can make this phase transition occur earlier by combinatorial optimization of the set of bits to "guess". These optimization questions are the main object of this paper. Many attacks which do not use any reflections have also been proposed [8], [10], [16] and also differential attacks which do not fall into the "Complexity Reduction" category. The most recent advanced differential attack on GOST has time complexity of 2^{179}, see [12], [14] which is also the best single-key attack known.

2. The internal structure of GOST

GOST is a block cipher with a simple Feistel structure, 64-bit block size, 256-bit keys and 32 rounds. Each round contains a key addition modulo 2^{32}, a set of 8 bijective S-boxes on 4 bits, and a simple rotation by 11 positions.

GOST has 32 identical rounds such as the one described on Fig. 1 below. They differ only by the subsets of 32 key bits which they use. GOST has a weak key schedule which is the main source of all the attacks on full 32-round GOST [8], [10]–[14], [16], [20]. In this paper however we only look at up to 8 rounds of GOST which have independent 32-bit keys which do not repeat, therefore we can ignore the GOST key scheduling totally.

We number the inputs of the S-box Si for $i = 1, 2, \ldots, 8$ by integers from $4i + 1$ to $4i + 4$ out of $1 \ldots 32$ and its outputs are numbered according to their final positions after the rotation by 11 positions: for example the inputs of S6 are $20, 21, 22, 23$ and the outputs are $32, 1, 2, 3$.

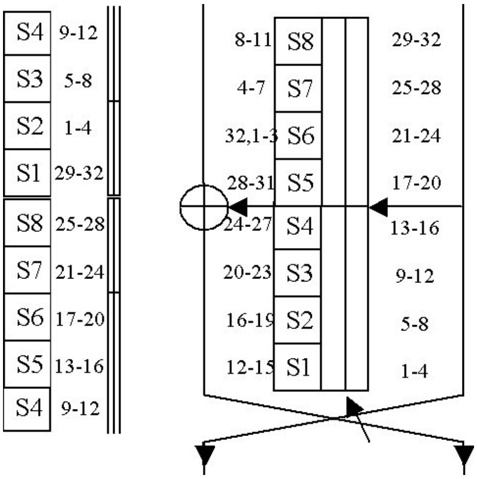

FIGURE 1. One round of GOST and connections in the following round.

On our picture Fig. 1 the $\boxplus$ denotes the addition modulo 2^{32}. At the left margin on Fig. 1 we also show S-box numbers in the next round, which is very helpful, to see which bits are successfully determined in our attacks on GOST. In a great simplification, in most cases, one S-box in one round affects essentially only two consecutive S-boxes in the next round. Additional propagation is obtained due to the Feistel structure and due to carries in the modular addition.

3. Algebraic cryptanalysis with complexity reduction

Following [8], [10] the work of cryptanalyst who wants to cryptanalyse GOST can be split into two independent tasks. First task is how to achieve a software algebraic attack on a reduced-round version as discussed in the previous section. The second question is if and **how** the complexity major variants of full GOST with 32 rounds can ever be **reduced** to a problem of breaking a cipher with much less rounds. In fact only in the recent 5 years it became possible to design and implement an appropriate last step for many such attacks which is a software algebraic attack, a Meet-In-the-Middle (MITM) attack, or other low-data complexity attack. This last step is the main focus in this paper.

3.1. Reductions and black-box reductions

The main idea is as follows [8], [10]: In order to reduce the attack complexity, we exploit the self-similarity of the cipher (due for example to a weak key schedule) and add some well-chosen assumptions which produce interesting and sometimes quite non-trivial consequences due to the high-level structural properties of the cipher, which makes cryptanalysis problems smaller, simpler and easier to solve. In this process we need to minimise the costs (in terms of probability that our assumptions hold) and to maximise the benefits (in terms of the number and the complexity of interesting relations which hold under these assumptions).

This process is called **Algebraic Complexity Reduction**, see [8], [10]. In most cases what we get is to compute (guess or determine) many internal values inside one or several decryptions, and literally break the cipher apart into smaller pieces. In particular we have **Black-Box Algebraic Complexity Reductions** where we obtain real black-box reductions, to for example the same cipher with strictly less rounds (and less data) again at the cost of some well-chosen assumptions. Most but not all reductions we are aware of are real “black box” reductions, see [10] for a detailed discussion. The notion of Algebraic Complexity Reduction creates new important **optimisation problems** in symmetric cryptanalysis: which deals with the fundamental question of how we can reduce the complexity of a cipher in cryptanalysis to a simpler problem, with a limited quantity of data, and with greatly reduced complexity, and this in the best possible (optimal) way while many interesting and non-trivial solutions will exist. One example of a Black-Box Algebraic Complexity Reduction from 2^{64} KP for 32 rounds of GOST, to 4 KP for 8 rounds of GOST, can be found in [8] and many more in [10].

Algebraic Complexity Reduction is a sort of umbrella paradigm which generalizes many already known fixed point, sliding, reflection and involution attacks. One is able to exploit similarities of individual sub-blocks and their inverses.

We have reflection attacks [21] but also have new attacks with double triple and quadruple reflection [10]. We are able to relax the conditions necessary in slide attacks [1] in nearly arbitrary ways. We have many new sorts of self-similarity attacks, cf. [10]. though reflection attacks [10], [21] are related to fixed point attacks which are an important attack for ciphers with block size being smaller than the key size, cf. [4], [5].

3.2. Optimizing the reductions: amplification

Reductions can be compared in terms of the number of pairs obtained, the resulting reduced number of rounds, success probability, and in terms of plaintext complexity, see [8], [10]. A key property of these reductions is the process of so called **Amplification** which is inspired by [6, Section 6.3].

Definition 1 (Amplification, Informal)**.** The goal of the attacker is to find a reduction where he makes some assumptions at a certain initial cost, for example they are true with probability 2^{-X} or work for certain proportion 2^{-Z} of keys. Then the attacker can in constant time determine many other internal bits inside the cipher to the total of Y bits.

We are only interested in cases in which the values X and Z are judged realistic for a given attack, for example $Z < 32$ and $X < 128$.

We call amplification the ratio $A = Y/X$.

The amplification is an important question in algebraic cryptanalysis which was previously discussed in [6]. We should note that there are some difficulties in defining this ratio formally:

(1) We claim that we need specific definitions for each individual cipher and for each specific attack method. For example Y can be the total number of linear equations obtained with the ElimLin algorithm [3], [6], [7] after adding a well-chosen set of X linear equations on the internal bits inside the cipher.

(2) Intuitively, the higher, this amplification coefficient A is, while X and Z remain below a certain threshold, the stronger and more surprising is the attack obtained.

(3) With higher values of X, the amplification can also be higher, however the attacker must limit the size of X for the whole the attack to remain fast enough overall.

(4) It is very difficult to know if an attack with given parameters may exist.

Example. In the most basic slide attack based on periodicity in the key scheduling [1] the amplification is unlimited. From one “slid pair”, we can obtain another “slid pair”, then another, etc. However in more advanced sliding attacks which operate with imperfect periodicity, the amplification can be limited, or occur only after the attacker guesses a few key bits, see [4], [5].

Another example where the amplification is exceptionally high is the Weak Key Family 3 in [10]. In this case the amplification is very very high: $X = 1$ and $Y = 256$ while $Z = 64$, see [10]. This can only be obtained for some weak keys in GOST with $Z = 64$. In spite of the additional of factor of 2^{64} implied by this value of Z the amplification is very high and overall it leads to very efficient attacks on certain GOST keys, see [9], [10]. The present paper is mostly motivated by the question how one can identify the suitable sets of key bits for such attacks.

3.3. Working at the low level

The amplification is easy to define when as in many initial steps in the cryptanalysis of GOST [10], we deal with black boxes. It is harder but possible to define at the low level, when we look at a complete functional description of a cipher. Here the question is what is the best possible software attack with tools such as the ElimLin algorithm [3], [6], [7] SAT solvers [2], [15], Gröbner bases [17] and other [24]. In all these algorithms we observe the phenomenon of Amplification in various forms. For example we can study and count linearly independent linear equations and try to amplify their number by the ElimLin algorithm, see [3], [6], [7].

When the ElimLin algorithm is itself the last step of the attack, or if the SAT solver is the last step of the attack, this amplification phenomenon becomes very important. We observe an avalanche-like phenomenon where more and more new linear equations are generated in the ElimLin algorithm, until the system is solved. Similarly, with SAT solver there is a point of phase transition where the problem becomes really easy to solve. If we want to understand algebraic cryptanalysis we need precisely to work on this face transition phenomenon itself. What happens after this threshold when the problem is just very easy to solve is less important.

In this paper we focus more specifically on cryptographic attacks with SAT solvers, and on GOST, which is a nice example of a weak government standard cipher with relatively poor diffusion. There are two main approaches in SAT cryptanalysis or two main algorithms to break a cipher with a SAT solver:

(1) **The SAT Method:** Guess X bits and run a SAT solver which, if the assumption on X bits is correct takes time T. Abort all the other computations at time T. The total time complexity is about $2^X \cdot T$.

(2) **The UNSAT Method:** Guess X bits and run a SAT solver which, if the assumption on X bits is incorrect finds a contradiction in time T with large probability $1 - P$ say 99 %.

With a small probability of $P > 0$, we can guess more key bits and either find additional contradictions or find the solution.

The idea is that if P is small enough the complexity of these additional steps can be less then the $2^X \cdot T$ spent in the initial UNSAT step.

(3) **A Mixed UNSAT/SAT Attack:** In practice maybe P is not as small as we wish, and therefore we may have a mix of SAT and UNSAT method: where the final complexity will be a sum of two terms none of which can be neglected. We will see a very nice example how a combined attack can be better than any of SAT and UNSAT methods in isolation in Section 6.

3.4. Contradiction immunity and SAT immunity

If we want to qualify the resistance of a cipher against the two attacks described above, it is natural to define the two following numbers:

Definition 2 (Contradiction immunity or UNSAT immunity)**.** We define the Contradiction Immunity of a given cipher and for $M = 1$ plaintext/ciphertext pairs of the cipher as being the smallest possible number of key bits which can be fixed so that given $M = 1$ KP we can obtain a contradiction with probability at least 50 % by just examining the logical consequences of these key bits. We require this contradiction to be found in a very short time, less than 1 second for the best SAT solver available.

Similarly we define:

Definition 3 (SAT Immunity or satisfaction immunity)**.** We define the SAT Immunity of a given cipher and for M plaintext/ciphertext pairs of the cipher as being the smallest possible number of key bits which can be fixed so that given M KP we can compute the secret key by the best available SAT solver in a relatively short time, say less than 1000 seconds.

Discussion: These notions are as precise as they can be. They depend on software used, but not excessively. Because we can only hope to provide upper bounds for this quantity by concrete "attacks" with concrete software, it makes sense to use (each time) the best available software, and improve these bounds slightly as the software improves. Importantly, we should consider that the first notion is much more robust and more fundamental: it expected to depend only on the connections between the components with the "optimal" subset of key bits, we do not expect that the contradiction will be found be examining too many other bits, but just by simple step-by-step local analysis. We also expect that the time to finding a contradiction will be essentially zero and will not depend too much on the software used. In contrast, the SAT Immunity can only be determined by somewhat "solving" the whole cipher, with the avalanche effect. Unless we are able to determine all the bits in the whole cipher, we do not know if the cipher is really solvable. It is safe to say that nobody really understands the complexity of SAT solvers in practice. Our experience shows that the results for the second notion will depend a lot on the SAT solver software used and where some software works well, some other does not seem to work at all (!).

A small technicality is that in order to determine the key uniquely in many ciphers with key size bigger than block size, it is necessary to use some $M > 1$ while for the first notion, for finding contradictions, we can frequently limit to considering the case where $M = 1$.

3.5. Applications of UNSAT/SAT immunities

Cryptanalysis. The main idea is that these two numbers will allow to evaluate the security of the cipher against cryptanalytic attacks with a SAT solver. Upper bounds we obtain do translate, more or less, as we will see, into concrete attacks with complexity of about 2^X. The two figures will also indicate which of the three strategies: SAT/UNSAT/Mixed is more likely to work.

Design of block ciphers. It is easy to see that the designer of a cipher can very effectively lower-bound these quantities. This will be achieved by making sure that each S-box in each round influences as many S-boxes as possible in the next round. This is not all very different than designing a cipher which is provably resistant to linear and differential cryptanalysis. Interestingly, Schneier once claimed that "Against differential and linear cryptanalysis, GOST is probably stronger than DES" [26]. Therefore we should also expect that Contradiction Immunity of DES and GOST are comparable. Happily, similar attacks with SAT solvers have been developed for both DES [3] and GOST [15]. In fact, it is obvious that the diffusion in DES is much better than in GOST and so is the Contradiction Immunity in DES. However we need to be careful about drawing any conclusions and direct comparisons do not mean much. If the contradiction immunity is 78 for 8 out of 32 rounds of GOST with 3 KP and 256-bit keys, is it better or less good, than contradiction immunity being 20 for 6 rounds out of 16 of DES with 1 KP and 56-bit keys? It is very hard to say.

4. Application to DES

In this section we give some basic results for DES obtained by the methods of [3] with however a better SAT solver CryptoMiniSat 2.92 [25]. Unhappily in DES the key bits are spread more or less uniformly in different rounds, and they tend to repeat many times. Therefore it is difficult to choose really good sets of bits and for now we just report upper bounds obtained when choosing the key bits at random, and letting the SAT solver to do the job.

Fact 1. *The Contradiction Immunity is at most 44 for 8 rounds of DES.*

Fact 2. *The SAT Immunity is at most 34 for 8 rounds of DES and 1 KP.*

These two results were obtained with the gate-efficient encoding described in [3], and with CryptoMiniSat 2.92 [25]. We found no attack on 8 rounds of DES and 1 KP which would be faster than brute force. For ultra low-data complexity attacks, 8 rounds of DES seem already secure or secure enough.

5. Application to GOST

In this paper we are going to provide some results on the Contradiction Immunity and SAT Immunity of GOST. These results are constructive upper bounds.

For a long time we thought that the Contradiction Immunity of 8 rounds of GOST was about 128. The reason for that can be found on Fig. 4 in [14]: it is possible to see that GOST splits very neatly into two nearly independent ciphers with 128-bit key each, which are only loosely connected. With this idea it is easy to understand why a software/algebraic attack on 8 rounds of GOST with complexity of 2^{120} seems plausible and natural. However recently we found that it is much lower than 128, much closer to 64. This was the main motivation for writing this paper.

Notation, cf. Fig. 2: We denote by $S^1 3$ just 1 higher ranking bit at S-box 1 in a given round. Similarly we denote by $S_3 3$ the 3 lower ranking bits of S3.

Fact 3. *The Contradiction Immunity for 8 rounds of GOST is at most 78.*

Justification: We have constructed and tried many different sets aiming at a contradiction in the middle. Our best set is as follows (cf. Fig. 2): 0-15,47-58, 64-70,111-114,128-130,175-182,192-202,239-255. The contradictions can be found in time of 0.06 s with CryptoMiniSat 2.92 software [25] with probability of about 50 %. It is easy to see that they could be obtained in essentially constant time by a dedicated algorithm with some small precomputed tables.

Remark. In fact we come remarkably close to 68 bits. If we consider the set of 68 bits later shown on Fig. 3 and also used in [9], we we achieve about 39 % UNSAT with CryptoMiniSat 2.92. It is remarkably close but it does not achieve 50 % required. This may seem strange, but in order to achieve 50 %, many more key bits are needed, cf. Fact 3 above. This is because in GOST it makes a lot of sense to guess key bits for full S-boxes, and S-boxes active in one round call for other S-boxes being also active.

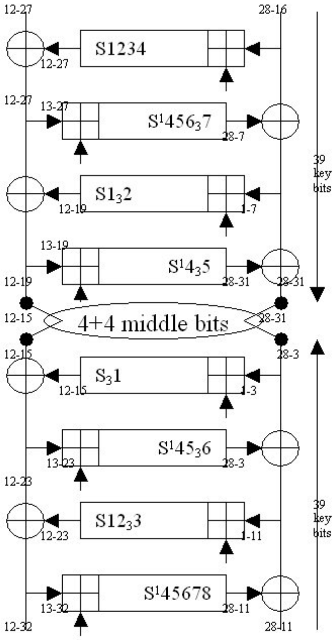

FIGURE 2. Our best set of 78 bits for UNSAT.

5.1. SAT immunity of GOST

Unhappily, it turns out that a set which is good for UNSAT is typically NOT good at SAT. No SAT solver software we dispose of is able to find the missing bits if the 78 bits of Fig. 2 are fixed. Happily we have found sets which are very good at SAT and they are in fact smaller than 78. Our best result is as follows:

FACT 4. *The SAT immunity for 8 rounds of GOST and 4 KP is at most 68.*

Justification: We use the following set of bits depicted on Fig. 3 0-15,51-55, 64-66,128-130,179-183,192-207,224-231,244-255 which is also used in [9]. All the remaining 256-68 bits can be determined in time of about 400 seconds using GOST encodings described in [15] and with CryptoMiniSat 2.92 [25].

From here a naive "SAT strategy" attack on GOST would be to run a SAT solver for 400 seconds 2^{68} times. This would be about 2^{99} GOST encryptions.

Further improvement. In order to improve this attack, the interesting question is if we can obtain contradictions when one of the 2^{68} cases is incorrect, and for what proportion of cases, and how long would it take. In other terms, we would like our set of 68 bits or possibly a smaller subset, to be good at UNSAT, which is the case as we have previously seen. In our attack with 4 KP we want to find a contradiction for all the 4 pairs simultaneously. This will be easier than contradiction with 1 KP we studied previously. This leads to the following improved attack which mixes the SAT and UNSAT strategies and which is also described in [9].

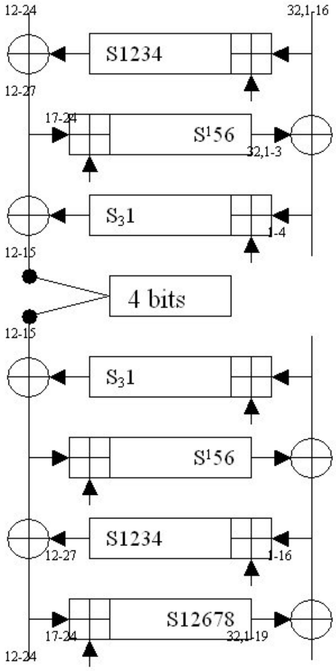

FIGURE 3. Our best set of 68 bits for SAT.

6. A mixed attack with 4 KP

FACT 5. *Given 4 KP for 8 rounds of GOST the full 256-bit key can be found in time of about* 2^{94} *GOST computations and negligible memory.*

Justification: As in [9] we proceed as follows.

(1) We use our set of 68 bits as on Fig 3.

(2) We run the software 2^{68} times for all possible assignments of the 68 bits.

(3) Computer simulations with the timeout of 7 seconds, a proportion of $1-2^{-5}$ of cases on 68 bits terminates with UNSAT within 2 s on average.

(4) Overall, we only need to run a proportion of 2^{-5} of all the 2^{68} cases for as many as 400 seconds, in other cases it simply terminates automatically within 2 s which is 2^{23} GOST encryptions on the same CPU.

(5) Assuming that all the other cases run for 400 s (some still terminate earlier) our conservative estimate of the attack time is $2^{68+23} + 2^{68+31-5} \approx 2^{94}$ GOST computations.

6.1. Application to full 32-round GOST

Following [9], [10], this can be transformed into an attack on GOST in the multiple key scenario. If there is a suitable population of at least 2^{64} different keys generated at random, then one can find one valid 256-bit key, in time of about 2^{94} GOST encryptions for one (weaker) key. This can be converted into attacks on full GOST not in just one way, on the contrary there are tens of ways of doing it which is outside of the scope of this paper, we refer to [8], [9], [10].

7. Conclusion

In this paper we introduce a new notion of Contradiction Immunity and a related notion of SAT Immunity. These definitions lead to new computational optimization problems in cryptography, which can be seen as looking for an optimal software guess-then-determine attack. We provide our best optimizations found which were constructed following a sort of meet-in-the middle strategy. Our key result is that the Contradiction Immunity for the GOST cipher is quite low, about 78, for 8 rounds, instead of 120 which we initially expected. We can compare GOST to DES: for 8 rounds of DES the Contradiction Immunity is relatively to key size, high enough to prevent attacks with a SAT solver.

The main outcome of this research is not just providing a bound on the two Immunity figures, but to provide concrete sets of bits based on which we can build concrete attacks on the given cipher. Theses sets are fundamental in being able to break 8 rounds of GOST in time of about 2^{94} and which extend to attacks on full GOST in a variety of ways, see [9], [10].

We postulate that the designers of ciphers should insure that they have sufficiently high Contradiction Immunity. This is **not** very hard to achieve and amounts to applying all the already known methods for achieving better diffusion: making sure that each S-box in each round influences as many S-boxes as possible in the next round, which is achieved by the well known tools such as the avalanche criteria, wide-trail strategy, MDS codes, larger S-boxes, etc.

Our notion of Contradiction Immunity is not exactly a well-defined mathematical concept. It clearly depends a lot on the software used and even more for the SAT Immunity. Therefore, on one side it is possible to consider that our work has a methodological flaw and is not well founded. However in cryptography problems to formally define certain concepts are not uncommon, for example security claims will depend on how the reference Turing Machine is built, and we do not know what is a secure hash function, we can only define a family of hash functions. In fact, on the other side, this property could be seen not as a flaw, but as an important feature. Using concrete software greatly increases the usefulness of our notions as comparison metric for different ciphers, as in fact we are able to substantiate our comparisons with actual working and highly optimized software key recovery attacks. This paper allows to built a concrete metric to measure the relative strength of various ciphers against realistic guess-then-determine attacks.

REFERENCES

[1] BIRYUKOV, A.—WAGNER, D.: *Advanced Slide Attacks*, in: Advances in Cryptology–EUROCRYPT '00, 19th Internat. Conf. on the Theory and Appl. of Cryptographic Tech., Bruges, Belgium, 2000 (B. Preneel, ed.), Lecture Notes in Comput. Sci., Vol. 1807, Springer, Berlin, 2000, pp. 598–606.

[2] BARD, G. V.—COURTOIS, N. T.—JEFFERSON, CH.: *Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers*, `http://eprint.iacr.org/2007/024/`. A working windows distribution with source code is available at: `http://www.nicolascourtois.com/software/CourtoisBardJefferson_public_distribution.zip`

[3] COURTOIS, N.—BARD, G. V.: *Algebraic cryptanalysis of the data encryption standard*, in: Cryptography and Coding, 11th IMA Internat. Conf. (S. Galbraith, ed.), Cirencester, UK, 2007, Lecture Notes in Comput. Sci., Vol. 4887, Springer, Berlin, 2007, pp. 152–169; `eprint.iacr.org/2006/402/`.

[4] COURTOIS, N.—BARD, G. V.—WAGNER, D.: *Algebraic and slide attacks on KeeLoq*, in: Fast Software Encryption, 15th Internat. Workshop—FSE '08 (K. Nyberg, ed.), Lausanne, Switzerland, 2008, Lecture Notes in Comput. Sci., Vol. 5086, Springer, Berlin, 2008, pp. 97–115.

[5] COURTOIS, N.—BARD, G. V.—BOGDANOV, A.: *Periodic ciphers with small blocks and aryptanalysis of KeeLoq*, Tatra Mt. Math. Publ. **41** (2008), 167–188.

[6] COURTOIS, N.—DEBRAIZE, B.: *Algebraic description and simultaneous linear approximations of addition in Snow 2.0.*, in: 10th Internat. Conf. on Information and Commun. Security—ICICS '08 (L. Chen et al., eds.), Birmingham, UK, 2008, Lecture Notes in Comput. Sci., Vol. 5308, Springer, Berlin, 2008, pp. 328–344.

[7] COURTOIS, N. T.—SEPHERDAD, P.—SUSIL, P.—VAUDENAY, S.: *ElimLin algorithm revisited*, in: Fast Software Encryption—FSE '12, 19th Internat. Workshop (A. Canteaut, ed.), Washington, 2012, Lecture Notes in Comput. Sci., Vol. 7549, Springer, Berlin, pp. 306–325.

[8] COURTOIS, N.: *Security evaluation of GOST 28147-89 in view of international standardisation*, Cryptologia **36** (2012) 2–13.

[9] COURTOIS, N.: *Low complexity key recovery attacks on GOST block cipher*, Cryptologia **37** (2013) (to apear).

[10] COURTOIS, N.: *Algebraic complexity reduction and cryptanalysis of GOST*, Preprint, 9 December 2012, `http://eprint.iacr.org/2011/626`.

[11] COURTOIS, N.—MISZTAL, M.: *Aggregated differentials and cryptanalysis of PP-1 and GOST*, in: 11th Central European. Conference on Cryptology—CECC '11, Debrecen, Hungary, 2012, Period. Math. Hungar. **65** (2012), 11–26.

[12] COURTOIS, N.—MISZTAL, M.: *First differential attack on full 32-round GOST*, in: 13th Internat. Conf.—ICICS '11 (S. Qing et al., eds.), Beijing, China, 2011, Lecture Notes in Comput. Sci., Vol. 7043, pp. 216–227.

[13] COURTOIS, N.—MISZTAL, M.: *Differential cryptanalysis of GOST*, Cryptology ePrint Archive, Report 2011/312, 14 June 2011, `http://eprint.iacr.org/2011/312`.

[14] COURTOIS, N.: *An improved differential attack on full GOST*, in: Cryptology ePrint Archive, Report 2012/138, 15 March 2012, `http://eprint.iacr.org/2012/138`.

[15] COURTOIS, N. T.—HULME, D.—MOUROUZIS, TH.: *Solving circuit optimisation problems in cryptography and cryptanalysis*, in: (informal) proceedings of SHARCS '12, Workshop, Washington, USA, pp. 179–191, `http://2012.sharcs.org/record.pdf`. Earlier preprint is available at, `http://eprint.iacr.org/2011/475`, and an abridged version appears in the electronic proceedings of the 2nd IMA conference Mathematics in Defence 2011, UK.

[16] DINUR, I.—DUNKELMAN, O.—SHAMIR, A.: *Improved attacks on full GOST*, in: Fast Software Encryption—FSE '12, 19th Internat. Workshop, Washington, USA, 2012, Lecture Notes in Comput. Sci., Vol. 7549, Springer, Berlin, 2012, pp. 9–28; early version available at `http://eprint.iacr.org/2011/558/`.

[17] FAUGÈRE, J.-CH.: *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5)*, in: Proc. of the Internat. Symp. on Symbolic and Algebraic Computation—ISSAC '02, New York, NY, 2002, ACM Press, New York, pp. 75–83.

[18] SHORIN, V. V.—JELEZNIAKOV, V. V.—GABIDULIN, E. M.: *Linear and differential cryptanalysis of Russian GOST*, Preprint submitted to Elsevier Preprint, 4 April 2001.

[19] ZABOTIN, I. A.—GLAZKOV, G. P.—ISAEVA, V. B.: *Cryptographic protection for information processing systems*, Government Standard of the USSR, GOST 28147-89, Government Committee of the USSR for Standards, 1989. ed States DES can be used ONLY for unclassified documents.

[20] 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 at `http://www.openssl.org/source/`

[21] ISOBE, T.: *A single-key attack on the full GOST block cipher*, in: Fast Software Encryption—FSE '11, 18th Internat. Workshop (A. Joux, ed.), Lyngby, Denmark, 2011, Lecture Notes in Comput. Sci., Vol. 6733, Springer, Berlin, 2011, pp. 290–305.

[22] KARA, O.: *Reflection cryptanalysis of some ciphers*, in: Progress in Cryptology–INDOCRYPT '08, 9th Internat. Conf. on Cryptology in India (R. Chowdhury et al., eds.), Kharagpur, India, 2008, Lecture Notes in Comput. Sci., Vol. 5365, Springer, Berlin, 2008, pp. 294–307.

[23] POSCHMANN, A.—LING, S.—WANG, H.: *256 bit standardized crypto for 650 GE GOST revisited*, in: Workshop on Cryptographic Hardware and Embedded Systems—CHES '10 Santa Barbara, California, 2010, Lecture Notes in Comput. Sci., Vol. 6225, Springer, Berlin, 2010, pp. 219–233.

[24] CHARNES, C.—O'CONNOR, L.—PIEPRZYK, J.—SAVAFI-NAINI, R.—ZHENG, Y.: *Comments on Soviet encryption algorithm*, in: Advances in Cryptology–EUROCRYPT '94 (A. De Santis, ed.), Lecture Notes in Comput. Sci., Vol. 950, Springer, Berlin, 1995, pp. 433–438.

[25] SEMAEV, I.: *Sparse algebraic equations over finite fields*, SIAM J. Comput. **39** (2009), 388–409.

[26] SÖRENSSON, N.—EÉN, N.—SOOS, M.: CryptoMiniSat 2.92, an open-source SAT solver package based on earlier MiniSat software, `http://www.msoos.org/cryptominisat2/`.

[27] SCHNEIER, B.: *Section 14.1 GOST (2nd ed.)*, in: Applied Cryptography, John Wiley and Sons, New York, 1996.

Received August 30, 2012

Nicolas T. Courtois
University College London
Gower street
London
UNITED KINGDOM
E-mail: n.courtois@cs.ucl.ac.uk

Jerzy A. Gawinecki
Military University of Technology
Warsaw
POLAND
E-mail: jgawinecki@wat.edu.pl

Guangyan Song
University College London
Gower street
London
UNITED KINGDOM
CISCO Systems
London
UNITED KINGDOM
E-mail: guangyan.song@gmail.com