Security Evaluation of GOST 28147-89 In View Of International Standardisation

Nicolas T. Courtois

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

Abstract. GOST 28147-89 is is a well-known 256-bit block cipher which is a plausible alternative for AES-256 and triple DES, which however has a much lower implementation cost, see [31]. GOST is implemented in standard crypto libraries such as OpenSSL and Crypto++ [25, 45], and is increasingly popular and used also outside its country of origin and on the Internet [23, 24, 31]. In 2010 GOST was submitted to ISO 18033, to become a worldwide industrial encryption standard.

Until 2011 researchers unanimously agreed that GOST could or should be very secure, which was summarized in 2010 in these words: "despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken", see [31]. Unhappily, it was recently discovered that GOST can be broken and is a deeply flawed cipher. There is a very considerable amount of recent not yet published work on cryptanalysis of GOST known to us, see [12]. One simple attack was already presented in February at FSE 2011, see [28]. In this short paper we describe another attack, to illustrate the fact that there are now attacks on GOST, which require much less memory, and don't even require the reflection property [29] to hold, without which the recent attack from [28] wouldn't work. We are also aware of many substantially faster attacks and of numerous special even weaker cases, see [12]. These will be published in appropriate peer-reviewed cryptography conferences but we must warn the ISO committees right now.

More generally, our ambition is to do more than just to point out that a major encryption standard is flawed. We would like to present and suggest a new general paradigm for effective symmetric cryptanalysis of so called "Algebraic Complexity Reduction" which in our opinion is going to structure and stimulate substantial amounts of academic research on symmetric cryptanalysis for many years to come. In this paper we will explain the main ideas behind it and explain also the precise concept of "Black-box Algebraic Complexity Reduction". This new paradigm builds on many already known attacks on symmetric ciphers, such as fixed point, slide, involution, cycling and other self-similarity attacks but the exact attacks we obtain, could never be developed previously, because only in the recent 5 years it became possible to show the existence of an appropriate last step for many such attacks, which is a low data complexity software algebraic attack. This methodology leads to a large number of new attacks on GOST [12], way more complex, better and more efficient than in [28]. One example of such an attack is given in the present paper.

Key Words: Block ciphers, Feistel schemes, key scheduling, self-similarity, reflection attacks, single-key attacks, algebraic attacks, algebraic complexity reduction, black-box reductions.

1 What Do We Know About GOST

1.1 The Official Status of GOST

GOST 28147-89 was standardized in 1989 and first it became an official standard for the protection of confidential information but the specification of the cipher remained confidential [21]. In 1994, the standard was declassified, published and also translated to English [21, 22]. It is also described in several more recent Internet standards [24, 23]. Unlike DES which could only be used to protect unclassified information, and like AES, GOST allows to protect also classified and secret information apparently without any limitations, which is explicitly stated by the Russian standard, see the first page of [22]. Therefore GOST is much more than a Russian equivalent of DES, and its large key size of 256 bits make GOST a plausible alternative for AES-256 and 3-key triple DES. The latter for the same block size of 64 bits offers keys of only 168 bits. Clearly GOST is a very serious military-grade cipher designed with most serious applications in mind. At least two sets of GOST S-boxes have been explicitly identified as being used by the two most prominent Russian banks, cf. [42, 25]). These banks need to securely communicate with tens of thousands of branches to protect assets worth many hundreds of billions of dollars against fraud.

1.2 Basic Cryptographic Specification 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. A particularity of GOST is that its S-boxes can be secret. and they can be used to constitute a secondary key which is common to a given application, further extending key size to a total of 610 bits. One set of S-boxes has been published in 1994 as a part of the Russian standard hash function specification GOST R 34.11-94 and according to Schneier [42] this set is used by the Central Bank of the Russian Federation. They also appear in more recent RFC4357 [24] as being part of the so called "id-GostR3411-94-CryptoProParamSet". A source code was included in [42] however this Schneier implementation specifies apparently a wrong (reversed) ordering of the S-boxes compared to later code contained in Crypto++ library [45]. This precise version of GOST 28147-89 block cipher is the most popular one, and it is commonly called just "the GOST cipher" in the cryptographic literature. The most complete current reference implementation of GOST which is of genuine Russian origin is a part of OpenSSL library and contains eight standard sets of S-boxes [25]. Other (secret) S-boxes could be recovered from a chip or implementation, see [39, 17].

1.3 GOST Is Very Competitive

In addition to the very long bit keys GOST has a much lower implementation cost than AES or any other comparable encryption algorithm. It really costs much less than AES: for example in hardware GOST 256 bits requires less than

800 GE, while AES-128 requires 3100 GE, see [31]. More than 4 time more gates for a much lower level of security (nearly 10^{40} times lower).

Thus it is not surprising that GOST became an Internet standard [24, 23], it is part of many crypto libraries such as OpenSSL and Crypto++ [25, 45], and is increasingly popular also outside its country of origin [23, 24, 31]. In 2010 GOST was submitted to ISO to become a worldwide encryption standard. Very few crypto algorithms have ever become an international standard. ISO/IEC 18033-3:2010 specifies the following algorithms. Four 64-bit block ciphers: TDEA, MISTY1, CAST-128, HIGHT and three 128-bit block ciphers: AES, Camellia, SEED. GOST is intended to be added to the same standard ISO/IEC 18033-3.

Now it appears that never in history of industrial standardisation, we had such a competitive algorithm in terms of cost vs. claimed security level. GOST also has 20 years of cryptanalysis efforts behind it, and it appears that this claimed military-grade security level was never disputed, until now.

Update: In April 2011 [private communication] GOST was voted against by a majority of countries in an ISO vote in Singapore, but the result of this vote was later overthrown at the ISO SC27 plenary level, and thus ISO is still in the process of standardizing GOST at the time of submission of this paper.

1.4 What Experts Say About GOST

Nothing in the current knowledge and literature about GOST ever suggested that it could be insecure. On the contrary, large keys and a large number of 32 rounds make that GOST seems a plausible encryption algorithm to be used for many decades to come.

Everyone familiar with the Moore's Law, understands that, in theory 256-bit keys should remain secure for at least 200 years. GOST was widely studied by the top cryptography experts active in the area of block cipher cryptanalysis such as Schneier, Biham, Biryukov, Dunkelman, Wagner, various Australian, Japanese, German and Russian scientists, ISO cryptography experts, and all researchers always seemed to agree that it could be or should be secure. While it is widely understood that the structure of GOST is in itself quite weak, for example compared to DES, and in particular the diffusion is not quite as good, it was however always stipulated that this should be compensated by a large number of 32 rounds cf. [19, 42, 40] and also by the additional non-linearity and diffusion provided by modular additions [19, 34]. In [3], Biryukov and Wagner write: "A huge number of rounds (32) and a well studied Feistel construction combined with Shannon's substitution- permutation sequence provide a solid basis for GOST's security." In the same paper we read: "after considerable amount of time and effort, no progress in cryptanalysis of the standard was made in the open literature". Thus, so far there was no significant attack on this algorithm from the point of view of communications confidentiality: an attack which would allow decryption or key recovery in a realistic scenario where GOST is used for encryption with various random keys. In contrast, there are already many many papers on weak keys in GOST [29, 3], attacks for some well-chosen number of rounds [29,

1, 40], attacks with modular additions removed [3], related-key attacks [30, 16, 36], reverse engineering attacks on S-boxes [39, 17], and at Crypto 2008 the hash function based on this cipher was broken [27]. In all these attacks the attacker has much more freedom than we would allow ourselves here. However, as far as traditional encryption applications with random keys are concerned, until now, no cryptographically significant attack on GOST was ever found, which was summarized in 2010 in these words: "despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken", see [31].

1.5 Linear and Differential Cryptanalysis of GOST

In the well known Schneier textbook we read: "Against differential and linear cryptanalysis, GOST is probably stronger than DES", see [42]. A basic assessment of the security of GOST against linear and differential cryptanalysis has been conducted in 2000 by Gabidulin *et al*, see [20, 19]. The results are quite impressive: at the prescribed security of level of 2^{256}, 5 rounds are sufficient to protect GOST against linear cryptanalysis. Moreover, even if the S-boxes are replaced by identity, and the only non-linear operation in the cipher is the addition modulo 2^{32}, the cipher is still secure against linear cryptanalysis after 6 rounds out of 32. Differential cryptanalysis of GOST seems comparatively easier and have attracted more attention. In [19] the authors also estimate that, but here only w.r.t. the security level of about 2^{128} 7 rounds should be sufficient to protect GOST against differential cryptanalysis. The authors also claim that "breaking the GOST with five or more rounds is very hard". Moreover, two Japanese researchers [40], show that the straightforward classical differential attack with one single differential characteristic is unlikely to work **at all** for a large number of rounds. This is due to the fact that when we study reasonably "good" iterative differential characteristics for a limited number of rounds (which already propagate with probabilities not better than $2^{-11.4}$ per round, cf. [40]), we realize that they only work for a fraction of keys smaller than half. For full 32-round GOST such an attack with a single characteristic would work only for a negligible fraction of keys of about 2^{-62} (and even for this tiny fraction if would propagate with a probability not better than 2^{-360}).

In the same paper [40], more advanced differential attacks on GOST are described. They exploit sets of differentials which follow certain patterns, for example certain S-boxes have zero differentials, other bits have non-zero differentials. These are essentially distinguisher attacks on the weak diffusion of GOST and they differ considerably from the classical differential cryptanalysis: sets of differentials occur naturally with higher probability, and when they occur they give much less exploitable information about the secret keys. The best advanced multiple differential attack proposed in [40] allows to break between 12 and 17 rounds of GOST depending on the key, some keys being weaker. It is not clear at all, if these attacks can be extended in any way to a larger number of rounds such as full 32 rounds, because partial internal differences generated in the attack become very hard to distinguish from differences which occur naturally at random.

1.6 Sliding and Reflection Attacks

According to Biryukov and Wagner, the structure of GOST, and in particular the reversed order of keys in the last 8 rounds, makes it secure against sliding attacks [18, 2, 3]. However the cipher still has a lot of self-similarity and this exact inversion of keys allows other attacks in which fixed points are combined with a so called "Reflection" property [27, 29]. The latter attack breaks GOST only for certain keys, which are weak keys. For these keys it is possible to break GOST with a complexity of 2^{192} and with 2^{32} chosen plaintexts.

1.7 Recent Developments

A new attack which also uses reflection, and finally breaks GOST, was very recently presented at FSE 2011, see [28]. The same attack was also independently discovered by us in [12]. This attack requires about 2^{132} bytes of memory which makes it arguably worse even than slower attacks with less memory.

Many new attacks which also use reflections and even simultaneous multiple reflections, which work for most GOST keys, and which allow to really break full-round GOST with 256-bit keys, not only for some weak keys like in [29] have been recently developed, see [12]. All these attacks require much less memory, and some are substantially faster, see [12].

These new attacks can be seen as examples of a new general paradigm for block cipher cryptanalysis called "Algebraic Complexity Reduction" which generalizes these attacks, and also generalizes many other known fixed point, slide, involution and cycling attacks. Importantly, in this large family of attacks, there are attacks which allow to cryptanalyse GOST, without any reflections, and without any symmetric points which appear during the computations, see [12]. One example of such a novel yet simple attack which breaks GOST and does not use any reflections is given in this paper.

2 Algebraic Cryptanalysis and Low Data Complexity Attacks on Reduced-Round Block Ciphers

Algebraic attacks, on block and stream ciphers, can be defined as attacks in which the problem of key recovery is written as a problem of solving a large system of Boolean algebraic equations which follows the geometry and structure of a particular cryptographic circuit [5–7, 9, 15]. The main idea was explicitly proposed by Shannon in 1949, see [44]. For DES the idea was articulated as a method of Formal Coding [26]. The best currently known attack on DES can be found in [9]: it allows to break only 6 rounds of DES given only 1 known plaintext. The most efficient attacks nowadays are based on writing ciphers as systems of multivariate polynomial equations and manipulating these equations using either algebraic tools (elimination algorithms such as XL, Gröbner Bases [14] and ElimLin cf. [11]) or constraint satisfaction software such as SAT solvers which solve algebraic problems after conversion [8]. Many other methods have been proposed recently [37, 38] and for one problem instance many different attack techniques do usually work to some extent, see [9] and though SAT solvers

do frequently solve many practical problems where Gröbner bases run out of memory, see [8], it was also shown in [8] that in a few cases where both methods worked, Gröbner bases methods were actually faster. We summarize all these methods which use "solver software" to determine unknown variables inside a complex circuit of Boolean equations under the general term of Algebraic Cryptanalysis (AC).

2.1 Algebraic Attacks - Application to GOST

GOST is a Feistel cipher with 32 rounds. In each round we have a round function $f_{k_i}(X)$ with a 32-bit sub-key k_i. Each round function contains a key addition modulo 2^{32}, a set of 8 bijective S-boxes on 4 bits, and a simple rotation by 11 positions. We need to to find a way to represent the cipher as an algebraic system of equations in such a way that it can efficiently be solved. It can be seen as encoding the problem of key recovery as an instance of an NP-hard problem. Both methods for encoding ciphers as such problems, and advanced heuristic algorithms for solving such problems are in constant evolution and are constantly improved. We have developed several efficient methods for formal encoding of GOST block cipher in the spirit of [9] and a lot of complex encoding, conversion and solver software for algebraic cryptanalysis. Our current best method for GOST is pretty much the same as the best known encoding method for DES described in [9].

Fact 1 (Key Recovery for 8 Rounds and 4 KP). Given 4 P/C pairs for 8 rounds of GOST we can find the full 256-bit key in time equivalent to 2^{120} GOST encryptions on the same software platform. The storage requirements are negligible.

Justification: We encode the S-boxes as an algebraic system of I/O relations (equations which relate Inputs and Outputs of these S-boxes), in a very similar way as for DES, see [9] for more details. Furthermore, in our fastest attacks, and also in the fastest attacks described in [9], we use about 20 additional variables per S-box, which allow equations be more more sparse. In order to encode the addition modulo 2^{32} we follow the first method described in [11]. The concatenation of all these equations describing the whole cipher or a large chunk of it is solved by various solver software. Given the fact that GOST has "weak diffusion" and that overall GOST is "not too complex" compared to any other block cipher (see [31] for the questions of gate-efficient implementation of GOST) we expect that to some extent our systems are solvable in practice. This is confirmed by our computer simulations.

3 On Conditional Algebraic Attacks on Ciphers

Algebraic attacks allow to cryptanalyse quite a few stream ciphers see [7, 6] but for block ciphers they only work for a limited number of rounds, see [7, 5, 9, 10]. Additional tricks are needed to reduce the complexity of an algebraic attack. This section deals with prior art and can be omitted in the first reading.

Conditional algebraic attacks, which could also be called Guess-Then-Algebraic attacks, make some, more or less clever assumptions on the internal variables of

the cipher of key bits, and determine all the other variables. The goal is to simplify the system of equations in such a way that it becomes solvable in practice. There are many methods to achieve that, some work locally, some with larger pieces of the cipher computation circuit.

In many cases, for example for DES [9], it turns out that the best way is to just fix say the first 20 key variables, and determine the other. In other ciphers, there are other highly non-trivial ways of making assumptions. In [11] the authors study the concept of (Probabilistic) Conditional Describing Degree of addition modulo 2^n. The main idea is that certain linear equations can be added as assumptions about the internal state of the cryptosystem, and they may produce a larger number of additional linear equations **simultaneously** true with high probability.

A different and powerful method to achieve this type of simplification, at a higher level, is to use self-similarity of the cipher and individual components of it. Many ciphers have important high-level self-similarity properties. This is exploited in slide attacks and in an increasing number of more sophisticated self-similarity attacks [1, 3, 17, 10] some of which exploit fixed points and have nothing to do with slide attacks. In many of these attacks the last step can be an Algebraic Cryptanalysis (AC) step. For example in one Slide-Algebraic Attack 1 on the KeeLoq block cipher [10], the attacker guesses 16 bits of the key and one pair of the plaintexts to be a so called "slid pair", where the two encryptions coincide with a shift by 64 rounds. This leads to an algebraic problem of a much smaller size and allows to break the cipher.

Our attacks and those in [12] inherit the ideas of all the attacks we mention above: they take a quite non-trivial method for algebraic description of S-boxes [9], a particular method for algebraic description of addition modulo 2^n [11], and some "clever" assumptions made at the high-level description of the cipher as in [18, 2, 3, 1, 17, 10, 12]. Our attacks on GOST bear some resemblance to certain known attacks on KeeLoq: both GOST and KeeLoq are ciphers relatively small block size compared to key size, imperfect periodicity (cf. [2, 3, 1, 10]) and weak internal structure which is expected to be compensated by a larger number of rounds. But it isn't and one is able to break ful 256-bit GOST 2^{39} times faster than brute force, and some variants can be broken in practice, see [12].

Now it is important to see that the work of cryptanalyst for GOST (and also many other ciphers) can be split into two independent tasks. The first task is to achieve and further improve this type of software attacks, see [7, 9, 15]. this area is very technical and requires a lot of programming an optimisation. The second task, is to see **how** can the complexity of GOST be **reduced** and moreover this will frequently be a real **"black box" reduction**, so that we can ever hope to be able to apply results such as Fact 1. We call it "Algebraic Complexity Reduction" and we claim that that this area contains a plethora of new combinatorial cryptanalysis tricks, which are going to produce a very large number of non-trivial new cryptanalytic attacks in the near future (some twenty different attacks are already given in [12]).

4 Algebraic Complexity Reduction

The idea stems from conditional algebraic attacks on symmetric ciphers and it also generalizes many already known structural high-level attacks on symmetric ciphers. The main idea is as follows: In order to reduce the attack complexity, we exploit the self-similarity of the cipher 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. We call this process **Algebraic Complexity Reduction**. 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. It 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. Solving this type of optimisation problems is going to create, as we anticipate, new important NP-hard problems of cryptographic importance, and developing formal mathematical proofs that certain optimizations have no solution, is going to again, as we anticipate, to create a whole new area in provable security of symmetric ciphers against algebraic attacks.

4.1 Black-Box Reductions

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 [12] for a detailed discussion.

Algebraic Complexity Reduction applies principally to ciphers, which have a lot of **self-similarity**, especially for larger components. In block ciphers this will be due for example to a very simple key schedule. First a certain number of assumptions on internal variables of the cipher, for one or several encryptions, are made. Then, if the assumptions hold, certain well chosen variables inside the encryption circuit(s) may be guessed or determined by the attacker. The key point is to do it in such a way as to minimise the costs and to maximise the benefits. In order to achieve an actual complexity reduction we need to solve a certain non-trivial combinatorial puzzle and optimisation problem, and it is not clear at all if such a puzzle will have a solution for any given cipher. Finally the combination of the assumptions the guessed values and determined values, will allow the attacker to obtain a small number of for example 4 P/C pairs for, for example for 8 rounds of the cipher, which will be true with a certain probability, for example 2^{-96}.

Then comes the final key recovery step which given the very small quantity of data obtained is most likely an algebraic attack. Our paradigm, especially in its Black-Box version, allows to very neatly **split the task** of the cryptanalysts

in two independent tasks. The performance of each task can be studied independently. **Isn't Algebraic Complexity Reduction already known**, in many different forms? In a sense yes. Many well-known attacks such as fixed point, slide, involution and cycling attacks will also lead to "Algebraic Complexity Reduction" attacks. However, as more advanced attacks of this type are developed, the quantity of data available in the last step of the attack decreases, as seen in [10, **?**,12] and also in this paper. Therefore the importance of Algebraic Cryptanalysis and similar **low-data complexity** attacks [9, 13] is likely to increase in the future. In fact many of such attacks **would never been discovered, or never seen as valid cryptanalytic tools**, because **only in the recent 5 years** it became possible to design and implement an appropriate last step (cf. Fact 1) for many such attacks. Today's cryptanalysts need to embrace the paradigm of Algebraic Complexity Reduction to be able to have a better visibility of what can be done, and to drive specialization among cryptanalysts, handling separate tasks in advanced attacks.

In what follows we will present just one attack which illustrates very well the concept of Algebraic Complexity Reduction, with a real black box reduction, and which also allows to break GOST.

5 High-level Structure of GOST

GOST is a Feistel cipher with 32 rounds. In each round we have a round function $f_k(X)$ with a 32-bit key which uses a 32-bit segment of the original 256-bit key which is divided into eight 32-bit sub-keys $k = (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$.

One 32-bit sub-key is used in each round, and their exact order is as follows:

rounds	1 8	9 16	17 24	25 32
keys	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$k_7 k_6 k_5 k_4 k_3 k_2 k_1 \mathbf{k_0}$

Table 1. Key schedule in GOST

We write GOST as the following functional decomposition (to be read from right to left) which is the same as used at Indocrypt 2008 [29]:

$$Enc_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E} \tag{1}$$

Where $\mathcal{E}$ is exactly the first 8 rounds which exploits the whole 256-bit key, $\mathcal{S}$ is a swap function which exchanges the left and right hand sides and does not depend on the key, and $\mathcal{D}$ is the corresponding decryption function with $\mathcal{E} \circ \mathcal{D} = \mathcal{D} \circ \mathcal{E} = Id$.

6 How To Break GOST

We describe a relatively simple attack on GOST. It is by far not the best attack on GOST, see [12], but it is a good illustration for our general methodology. It consists of two stages. We have a black box reduction stage and key recover stage. We proceed as follows. We consider plaintexts with a very peculiar property:

Assumption 1 (Assumption W). Let A be such that $\mathcal{E}(D) = \overline{D}$ where D is defined as $D = \mathcal{E}^3(A)$.

This kind of event is very likely to happen in the real life.

Fact 2 (Property W). Given 2^{64} KP there is on average one value A which satisfies the Assumption. For 63% of all GOST keys at least one such A exists.

Remark: For the remaining 37 % of keys this attack fails. However many other attacks still work, see [12].

This property has some very important consequences:

Fact 3 (Consequences of Property W). If A satisfies the Assumption W above and defining $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ we have:

1. $Enc_k(A) = D$. This is illustrated on the right hand side of Fig. 1.
2. $Enc_k(B) = C$ This can be seen on the left hand side of Fig. 1.

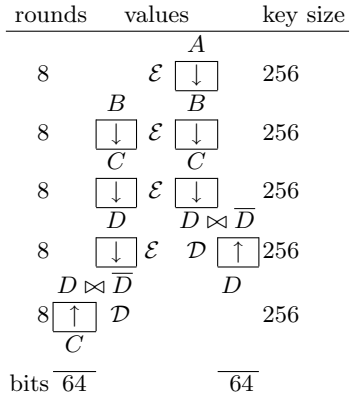

Fig. 1. A black-box "Algebraic Complexity Reduction" from 32 to 8 rounds of GOST

This leads directly to our new reduction:

Reduction 1. [From 2^{64} KP for 32 Rounds to 4 KP for 8 Rounds]
Given 2^{64} known plaintexts for GOST, it is possible to obtain four P/C pairs for 8 rounds of GOST and our guess will be correct with probability 2^{-128}.

Justification: Given 2^{64} known plaintexts, there is on average one value $A = X_i$ with Property W. We guess A and B and our choice is correct with probability 2^{-128}. This gives us immediately C and D as shown on Fig. 1. For each (A, B) this computation of (C, D) is done in constant time if we assume that all the 2^{64} pairs X_i, Y_i are stored using a hash table.

Thus we obtained 4 pairs for 8 rounds of GOST:
$A \mapsto B, B \mapsto C, C \mapsto D, D \mapsto \overline{D}$.

Resulting Attack. If we combine this with Fact 1 we get an attack which breaks GOST given 2^{64} known plaintexts, time is 2^8 times faster than brute force. The storage required is for the 2^{64} known P/C pairs. GOST is broken.

7 Conclusion

GOST was designed to provide a military level of security and to last 200 years. Most major block cipher encryption experts have studied GOST, and in 2010 the consensus was still to say that "despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken", see [31]. In 2010 GOST was submitted to ISO 18033 to become a worldwide encryption standard.

The general idea of Algebraic Cryptanalysis has been around for more than 60 years [44, 26]. Yet only in the last 10 years several efficient software tools for solving various NP-hard problems involved have been developed, while numerous specific vulnerabilities leading to efficient attacks of this type have been found. A number of stream ciphers are indeed broken [7, 6, 15]. However only one block cipher KeeLoq could so far be shown to be weak enough, to be broken using an algebraic attack [10]. In this paper we break another important real-life block cipher GOST. It is **the first time in history that a standard government block cipher is broken by an algebraic attack**.

One simple MITM-Reflection attack on GOST was already presented at FSE 2011 conference, see [28]. In this short paper we describe just one another attack, to illustrate the fact that there is now many other attacks on GOST, many of which are faster (see [12]) and all of which including the one presented here, are algebraic attacks which require fundamentally much less memory and create an infinitely more possibilities for the attacker to break the cipher in various ways. Also, in this paper, we already establish that one does **not** need the reflection property [29] to break GOST.

Clearly GOST is deeply flawed, in more than one way, and **GOST does not provide the security level required by ISO**. A plethora of other attacks following our general idea and paradigm for symmetric cryptanalysis, called "Algebraic Complexity Reduction" is given in [12]. With this framework which we amply describe here and illustrate with one attack, we ambition to considerably enlarge the spectrum of self-similarity attacks on block ciphers.

We must also report some facts, known to us, and the reader will excuse us for not being able to give more details now, but this is very important for the sake of the still ongoing process at the time of writing of ISO standardisation. There is much more than just a "certificational" attack on GOST faster than brute force [28]. In fact to standardize GOST now would be really dangerous and irresponsible. This is because **some of our attacks are feasible in practice**. Some GOST keys can indeed be decrypted in practice, which are either weak keys, or for particular natural versions of GOST. See [12] and our forthcoming publications on the same topic, for a detailed discussion of cases in which this will be possible. It appears that also that it is **for the first time in history that a major standardized block cipher intended to provide a military-grade level of security and intended to protect also classified and secret documents, for the government, large banks and other organisations, is broken by a mathematical attack.**

References

1. Eli Biham, Orr Dunkelman, Nathan Keller: *Improved Slide Attacks,* In FSE 2007, LNCS 4593 Springer 2007, pp. 153-166.
2. A. Biryukov, D.Wagner: *Slide Attacks,* In proceedings of FSE'99, LNCS 1636, pp. 245-259, Springer, 1999.
3. Alex Biryukov, David Wagner: *Advanced Slide Attacks,* In Eurocrypt 2000, LNCS 1807, pp. 589-606, Springer 2000.
4. Christophe De Cannière: *GOST article,* In ENCYCLOPEDIA OF CRYPTOGRAPHY AND SECURITY 2005, pp. 242-243.
5. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
6. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, LNCS 2656, pp. 345-359, Springer. An extended version is available at `http://www.minrank.org/toyolili.pdf`
7. 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.
8. Gregory V. Bard, Nicolas T. Courtois and Chris Jefferson: *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/`.
9. 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/`.
10. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
11. 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.
12. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* 17 February 2011, 28 pages, original preprint submitted to Crypto 2011. MD5 Hash is d1e272a75601405d156618176cf98218. SHA-1 Hash is 6C16C46E 00AFD74B 3ED4949B 7766D5BF 6EC7DDBB. The fastest attack on full-round 256-bit GOST presented in this paper has a time complexity of 2^{216}. The paper also contained one nearly-practical attack on a well-known practical variant of GOST which allows to break some keys in practice. Many more important attacks were added later, current version has 54 pages, to be published soon, probably wil be split in several pieces.
13. Charles Bouilleguet, Patrick Derbez, Orr Dunkelman, Nathan Keller, Pierre-Alain Fouque: *Low Data Complexity Attacks on AES,* Cryptology ePrint Archive, Report 2010/633. `http://eprint.iacr.org/2010/633/`.
14. Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),* Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002, ACM Press.
15. Gwenolé Ars, Jean-Charles Faugère: *An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner Bases* INRIA research report, available at https://hal.ccsd.cnrs.fr/.
16. Fleischmann Ewan, Gorski Michael, Huehne Jan-Hendrik, Lucks Stefan: *Key recovery attack on full GOST block cipher with zero time and memory,* Published as ISO/IEC JTC 1/SC 27 N8229. 2009.

17. Soichi Furuya: *Slide Attacks with a Known-Plaintext Cryptanalysis,* In ICISC 2001, LNCS 2288, 2002, pp. 11-50.
18. E. K. Grossman, B. Tuckerman: *Analysis of a Weakened Feistel-like Cipher,* 1978 International Conference on Communications, pp.46.3.1-46.3.5, Alger Press Limited, 1978.
19. Vitaly V. Shorin, Vadim V. Jelezniakov and Ernst M. Gabidulin: *Linear and Differential Cryptanalysis of Russian GOST,* Preprint submitted to Elsevier Preprint, 4 April 2001
20. V.V. Shorin, V.V. Jelezniakov, E.M. Gabidulin *Security of algorithm GOST 28147-89,* (in Russian), In Abstracts of XLIII MIPT Science Conference, December 8-9, 2000.
21. I. A. Zabotin, G. P. Glazkov, V. B. Isaeva: *Cryptographic Protection for Information Processing Systems,* Government Standard of the USSR, GOST 28147-89, Government Committee of the USSR for Standards, 1989. In Russian, translated to English in [22].
22. An English translation of [21] by Aleksandr Malchik with an English Preface cowritten with Whitfield Diffie, can be found at `http://www.autochthonous.org/crypto/gosthash.tar.gz`
23. Vasily Dolmatov, Editor, RFC 5830: *GOST 28147-89 encryption, decryption and MAC algorithms,* IETF. ISSN: 2070-1721. March 2010. `http://tools.ietf.org/html/rfc5830`
24. V. Popov, I. Kurepkin, S. Leontie: *RFC 4357: Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms,* IETF January 2006. `http://tools.ietf.org/html/rfc4357`
25. 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/`
26. J. Hulsbosch: *Analyse van de zwakheden van het DES-algoritme door middel van formele codering,* Master thesis, K. U. Leuven, Belgium, 1982.
27. Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak and Janusz Szmidt: *Cryptanalysis of the GOST Hash Function,* In Crypto 2008, LNCS 5157, pp. 162 - 178, Springer, 2008.
28. Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, Fast Software Ecnryption, Springer LNCS, 2011.
29. Orhun Kara: *Reflection Cryptanalysis of Some Ciphers,* In Indocrypt 2008, LNCS 5365, pp. 294-307, 2008.
30. John Kelsey, Bruce Schneier, David Wagner: *Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES,* In Crypto'96, LLNCS 1109, Springer, 1996.
31. 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.
32. J. Pieprzyk and L. Tombak, *Soviet Encryption Algorithm,* Preprint 94-10, Department of Computer Science, The University of Wollongong, 1994
33. C. Charnes, L. O'Connor, J. Pieprzyk, R. Savafi-Naini, Y. Zheng: *Further comments on GOST encryption algorithm,* Preprint 94-9, Department of Computer Science, The University of Wollongong, 1994.
34. C. Charnes, L. O'Connor, J. Pieprzyk, R. Savafi-Naini, Y. Zheng: *Comments on Soviet encryption algorithm,* In Advances in Cryptology - Eurocrypt'94 Proceedings, LNCS 950, A. De Santis, ed., pp. 433-438, Springer, 1995.

35. J.-J. Quisquater and Y. Desmedt and M. Davio: *The Importance of 'good' Key Scheduling Schemes (How to make a secure DES scheme with* ≤ 48 *bit keys?,* In Crypto'85, LNCS 218, pp. 537–542, Springer, 1985.
36. Vladimir Rudskoy: *On zero practical significance of Key recovery attack on full GOST block cipher with zero time and memory,*
37. Igor Semaev: *Sparse Algebraic Equations over Finite Fields,* SIAM J. Comput. 39(2): 388-409 (2009).
38. Haavard Raddum and Igor Semaev: New Technique for Solving Sparse Equation Systems, ECRYPT STVL website, January 16th 2006, available also at `eprint.iacr.org/2006/475/`
39. Markku-Juhani Saarinen: *A chosen key attack against the secret S-boxes of GOST,* unpublished manuscript, 1998.
40. Haruki Seki and Toshinobu Kaneko: *Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, Selected Areas in Cryptography, Douglas R. Stinson and Stafford E. Tavares, editors, LNCS 2012, pp. 315323, Springer, 2000.*
41. *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.
42. Bruce Schneier: *Section 14.1 GOST,* in *Applied Cryptography,* Second Edition, John Wiley and Sons, 1996. ISBN 0-471-11709-9.
43. Bruce Schneier, *The GOST Encryption Algorithm,* Dr. Dobbs Journal, Vol. 20, No. 2, 1995.
44. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in particular page 704.
45. Wei Dai: Crypto++, a public domain library containing a reference C++ implementation of GOST and test vectors, `http://www.cryptopp.com`