Algebraic Complexity Reduction and Cryptanalysis of GOST

Nicolas T. Courtois

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

Abstract. GOST 28147-89 is a well-known block cipher and the official encryption standard of the Russian Federation. Its large key size of 256 bits at a particularly low implementation cost [37] make GOST a plausible alternative for AES-256 and 3-key triple DES. GOST is widely used, implemented in OpenSSL and other crypto libraries [31, 49], and is increasingly popular also outside its country of origin [30, 37]. In 2010 GOST was submitted to ISO, to become an international standard.

In theory 256-bit keys could remain secure for 200 years. GOST was analysed by Schneier, Biham, Biryukov, Dunkelman, Wagner, various Australian, Japanese, and Russian scientists, and all researchers seemed to agree that it looks quite secure. Though the internal structure of GOST seems quite weak compared to DES, and in particular the diffusion is not quite as good, it is always stipulated that this should be compensated by a large number of 32 rounds cf. [27, 46, 45, 3] and by the additional non-linearity and diffusion provided by modular additions [27, 38]. At Crypto 2008 the hash function based on this cipher was broken. Yet as far as traditional encryption applications with single random keys are concerned, and until 2011, no cryptographically significant attack on this algorithm was found.

In this paper we present several attacks on full 32-rounds GOST two of which are substantially faster and all of which require much less memory. Our methodology is derived from the idea of conditional algebraic attacks on block ciphers [12, 11] which can be defined as attacks in which the problem of key recovery is written as a problem of solving a large system of algebraic equations, and where the attacker makes some "clever" assumptions on the cipher which lead to an important simplification in the algebraic description of the problem, which makes it solvable in practice if the assumptions hold. Our methods work by black box reduction and allow to literally break the cipher apart into smaller pieces and reduce breaking GOST to a low data complexity software algebraic attack on 8 or less rounds. Overall we obtain more than 10 distinct attacks faster than brute force on the full 32-round GOST the best of which are summarized in Table 2, and ten more attacks on various classes of weaker or special keys. These are shown in Table 3 and in Table 5 we present five very nearly practical attacks breaking two principal 128-bit variants of GOST known from the literature.

Major update (5/2012): many results in this paper are now greatly improved, our fastest attack on 256-bit keys is now in 2^{159}.

Key Words: Block ciphers, Feistel schemes, GOST, ISO 18033, key scheduling, self-similarity, advanced slide attacks, fixed points, reflection attacks, black-box reductions, low-data complexity attacks, MITM attacks, algebraic attacks on block ciphers.

1 Publication Info and Development History

This paper describes a general methodology for block cipher cryptanalysis through a reduction to a low-data complexity key recovery attack and more than 20 different attacks on GOST obtained with this methodology.

Most of the attacks were developed in the period September 2010-February 2011 when an original 28-pages version of this paper with five distinct attacks which break GOST faster than brute force was submitted to Crypto 2011. Just a few days earlier Isobe published his "first single-key" attack on GOST to be made public [34] with time complexity of 2^{225}. Each single attack in our paper was already faster than this attack, and our fastest attack had time complexity of 2^{216} (now it has become 2^{206}) though requiring much more data. It also contained essentially all current weak-key variants and a practical attack on one 128-bit key variant.

In May 2011 a much shorter version of it was submitted to Asiacrypt 2011 and it contained **two** distinct attacks with time complexity of 2^{216}. The complexity of these two attacks is now reduced to 2^{206} and 2^{195}. In May 2012 there was a **major update of this paper which greatly reduces the time complexity of most attacks**.

With our complexity reduction methodology we reduce the complexity of GOST to a number of well-defined cryptanalysis questions with less rounds and less data, which can be studied separately, and researchers can compare results obtained by many different methods. Another alternative final step for our second attack in previously 2^{216} and now 2^{195} was presented by Shamir *et al* at FSE 2012 with complexity of about 2^{192}, see [22]. A faster single-key attack on GOST was found very recently: 2^{178} in [19].

Very interesting things come from realizing that many of our earlier weak key attacks can be transformed into attacks able to recover ordinary GOST encryption keys generated at random faster than any other attack. Initially it was 2^{186} and one of these attacks did not work exactly as claimed, but we still have two such attacks here with the cost of the best of the two going down to 2^{159} per key. Though it is no longer exactly a single key attack it certainly is a more realistic attack on GOST than 2^{192} from [22] and 2^{178} of [19]. Our 2^{159} per key is the most realistic attack on GOST encryption found so far.

Due to the substantial volume of this work, this paper will be split in several distinct papers which will be published separately. This is **the master paper** which is here for reference, to establish priority, and to show the big picture how all these attacks are related to each other. It also demonstrates that there is no single reason why GOST is an insecure cipher but rather that the self-similarity properties of GOST can be exploited in a variety of distinct more or less non-trivial ways. We break the GOST cipher faster than by brute force in a modular way, where the first step is a reduction step, the last step is a low-data complexity key recovery step, and where each step can be studied separately.

2 Impact and Significance of This Research

This paper has some serious significance both scientific and historical.

It is very rare to see a cipher submitted to ISO being broken during its standardization process, while nobody in the scientific community have expressed a slightest suspicion about its security. Over two decades less than 10 block ciphers were judged "good enough" to become a serious candidate for ISO standardisation, as GOST has become in 2010, cf. [37]. The fact that GOST was believed to be secure until 2011, and now can be broken in so many different ways is quite remarkable.

Similarly it is safe to say that nobody in the cryptographic community have ever thought that there will ever be an attack following the broad idea of software algebraic attacks [4, 10] which can break a real-life government/military/Internet standard block cipher faster than by brute force. Likewise until recently nobody would consider that low-data complexity attacks on reduced rounds of block ciphers such as in [10, 7, 21] are very important. However our paradigm of "Complexity Reduction" provides us with a large number of ways to reduce the complexity of breaking a cipher precisely to a low-data complexity attack. Some of these exploit already known fixed point, sliding, reflection and involution properties, other are totally new and non-trivial self-similarity attacks. We called it "Algebraic Complexity Reduction" because very few low-data complexity attacks are known, and a software "algebraic attack" was the initial motivation to find all these attacks, and remains one of the very few plausible last steps (can also be a meet in the middle attack, see Section 9.1 and [22]). To summarize our work brings powerful and disruptive new techniques in cryptanalysis leading to great many new attacks: new methods to achieve reduction to low-data complexity attacks, and efficient methods to deal with these low-data attacks.

It is also very rare to be able to break a real government standard cipher, used to protect classified and secret information, apparently without any limitations, cf. [29], unlike United States DES which could only be used to protect unclassified information. Though the complexity of most of our attacks is still too high to actually be able to decrypt the communications today and in practice, it is however wrong to believe that this is only academic research. It is widely believed that actual versions of GOST used in practice have additional properties which could make them much weaker than most versions studied in this paper. They could also be some of the weak versions we have already been able to break: for example for Family B of 128-bit GOST keys, given 2^{35} chosen plaintexts, we can recover keys in time 2^{81}, see Fact 43 on page 60.

The most likely impact of this research (and also other recent works on GOST from 2011 [34, 22, 13, 15–17]) is that Russia will have no other choice than to change sooner or later its national encryption standard. This can potentially cost billions of dollars in research, development, secure hardware and secure implementation development, telecommunications equipment overhauls, and maybe some upgrades in financial systems and critical data storage infrastructure.

3 Background

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. 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 [46] this set is used by the Central Bank of the Russian Federation. 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. There exists a Russian reference implementation of GOST which is a part of OpenSSL library and contains eight sets of S-boxes [31] which can be used for encryption or for the GOST Hash function. Overall, it is because of the small size of the GOST S-boxes (see [8, 10, 12, 4, 5]) that the cipher is vulnerable to "software algebraic attacks" [4, 8, 12, 42, 43, 11]. Therefore our attacks are expected to work with a similar complexity for any choice of S-boxes. In all our attacks we assume however that the S-boxes are known. Otherwise they could be recovered, see [44, 25].

It is widely known that the structure of GOST is in itself quite weak, an in particular the diffusion is quite poor, however, this is expected to be compensated by a large number of rounds [46]. 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 [35, 3], attacks for some well-chosen number of rounds [35, 1, 45], attacks with modular additions removed [3], related-key attacks [36, 24, 41], reverse engineering attacks on S-boxes [44, 25], and attacks on the hash function based on this cipher [33]. In all these attacks the attacker has much more freedom than we will allow ourselves.

In this paper we limit ourselves to the questions which pertain to the security of GOST used in encryption, with one single key chosen at random. So far no key recovery attack on full-round GOST was ever proposed. 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 [26, 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 [33, 35]. 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.

A basic assessment of the security of GOST against linear and differential cryptanalysis has been conducted in 2000 by Gabidulin *et al*, see [27]. 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 [27] the authors also estimate that, 7 rounds should be sufficient to protect GOST against differential cryptanalysis. Moreover, two Japanese researchers [45], 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. [45]), 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 it would propagate with a probability not better than 2^{-360}). The best advanced multiple differential attack proposed in [45] allows to break about 13 rounds of GOST. In 2011 better attacks of this type have been found, allowing finally to break full 32 round GOST faster than by brute force, see [16, 17, 19].

4 Preliminary Remarks on GOST

In this paper we call **a P/C pair** a pair of known **P**laintext and **C**iphertext for full GOST, or for a reduced-round version of GOST.

GOST has 64-bit block size and the key size of 256-bit keys. Accordingly:

Fact 1. 4 P/C pairs are necessary to determine the GOST key. With 4 P/C pairs we expect to get on average about one key. We get the correct key together with a list of, sometimes a few, but on average less than 1 wrong keys.

With 5 P/C pairs we are able to discard all these wrong keys in practice: the probability that just one more key works for this extra P/C pair is 2^{-64}. This is unlikely to ever happen in a single key recovery attack.

Fact 2. A brute force attack on GOST takes 2^{255} GOST encryptions on average.

Justification: We proceed as follows: we use one P/C pair and check all the possible keys. On average half way through the right key will be found. Only for an expected number of 2^{191} keys which are confirmed with the first pair, we do further comparisons. Most of these 2^{191} are **false positives**. This notion plays an important role in this paper. Here, and elsewhere, the key remark is that the total number of these false positives is small and the complexity of rejecting all the incorrect keys with additional P/C pairs is actually negligible. Indeed we have at most 2^{192} cases to be checked with another P/C pair. Then at most 2^{128} keys remain to be checked against the third P/C pair etc. Overall we need to do about $2^{255} + 2^{191} + 2^{127} + 2^{63} + 1$ GOST encryptions on average. This number is very close to 2^{255} GOST encryptions.

5 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 [4, 5, 8, 10]. The main idea was explicitly proposed by Shannon in 1949, see [47]. For DES the idea was articulated as a method of Formal Coding [32]. The best currently known attack on DES can be found in [10]: 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 [23] and ElimLin cf. [12]) or constraint satisfaction software such as SAT solvers which solve algebraic problems after conversion [9]. Many other methods have been proposed recently [42, 43] and for one problem instance many different attack techniques do usually work to some extent, see [10] and though SAT solvers do frequently solve many practical problems where Gröbner bases run out of memory, see [9], it was also shown in [9] 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). This is just a name: not all these attacks are very algebraic.

5.1 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 [10] 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 [10].

Fact 3 (Key Recovery for 4 Rounds and 2 KP). Given 2 P/C pairs for 4 rounds of GOST the 128-bit key can be recovered in time equivalent to 2^{24} GOST encryptions on the same software platform (it takes a few seconds). The memory requirements are very small. The attack works with a similar complexity for any choice of GOST S-boxes.

Justification: A detailed description of how to do it would be very tedious, and contain no new contributions compared to methods already described in [10].

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 [10] for more details. Furthermore, in our fastest attacks, and also in the fastest attacks described in [10], 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 [12]. The concatenation of all these equations describing the whole cipher or a large chunk of it is solved by various solver software.

This naive methodology is however is not enough to obtain really very good attacks on GOST. The crucial question in this type of attacks is which bits can be guessed and bits can be determined, what is the optimal guess-then-determine strategy, and how to implement such a strategy.

Fact 4 (Key Recovery for 8 Rounds and 2 KP). Given 2 P/C pairs for 8 rounds of GOST we can enumerate 2^{128} candidates for the full 256-bit key in total time equivalent to 2^{131} GOST encryptions on the same software platform. The memory requirements are negligible.

Justification: This is obtained by a hybrid MITM-Determine-Algebraic-SAT attack described in [14]. It is an attack which mimics the structure of a MITM attack with partial key guesses and uses a SAT solver to greatly reduce the time and the memory needed to perform the attack. Similarly we have:

Fact 5 (Key Recovery for 8 Rounds and 3 KP). Given 3 P/C pairs for 8 rounds of GOST we can produce 2^{64} candidates for the 256-bit key in time equivalent to 2^{110} GOST encryptions. The storage requirements are negligible and all the 2^{64} candidates can be produced in an uniform way, each of them is produced in time of 2^{46} GOST encryptions on average.

Justification: This is obtained by another MITM-Algebraic attack, see [14].

Remark: this result can be compared to the a threshold of 2^{128} which one could obtain in a Meet-In-The-Middle (MITM) attack, cf. Fact 8. Our attack is much faster and simultaneously requires negligible storage.

Finally, we also established that:

Fact 6 (Key Recovery for 8 Rounds and 4 KP). Given 4 P/C pairs for 8 rounds of GOST we can recover the full 256-bit key in time equivalent to 2^{94} GOST encryptions with negligible memory.

Justification: In [14] we describe two attacks with complexity of 2^{94} for 4 KP. One is an excessively technical MITM attack with large memory, another is a super simple software attack with same running time and negligible memory.

Important Remark: The main object of this paper is **NOT how to** achieve and further improve various software/algebraic/MITM attacks with low data complexity and low number of GOST rounds [8, 10, 14, 22, 50] but **how** can the complexity of GOST be **reduced in the "black box" way**, so that we can ever hope to be able to apply results such as Fact 6.

6 On Conditional Algebraic Attacks on Ciphers

Algebraic attacks allow to break many stream ciphers [8, 5, 6] but for block ciphers they only work for a limited number of rounds [8, 4, 10, 11]. Additional tricks are needed to reduce the complexity of an algebraic attack.

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 [10], it turns out that the best way is to just fix say the first 20 key variables, and determine the other. This is due to the fact that in DES key variables repeat quite uniformly at random inside the algorithm. They repeat many times, and guessing a number of these variables leads to very important simplifications. In contrast other variables do not repeat. in the algebraic description, and in absence of additional properties which could be exploited, there is no particularly clever method to choose variables which would work even comparably as well as guessing these key variables.

In other ciphers, there are other highly non-trivial ways of making assumptions. In [12] 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, 25, 11] 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 [11], 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.

The attacks we present in this paper inherit the ideas of all the attacks we mention above: they take a quite non-trivial method for algebraic description of S-boxes [10], a particular method for algebraic description of addition modulo 2^n [12], and some clever tricks at the high-level description of the cipher as in [26, 2, 3, 1, 25, 11]. 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, 11]) and weak internal structure which is expected to be compensated by a larger number of rounds. But it isn't and we are able to break GOST a lot faster than brute force.

7 High-level Description of GOST and Key Observations

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	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$k_7k_6k_5k_4k_3k_2k_1\mathbf{k_0}$

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 [35]:

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

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 $\mathcal{E} \circ \mathcal{D} = \mathcal{D} \circ \mathcal{E} = Id$.

Notation. We call $\overline{X}$ the value $\mathcal{S}(X)$ where both 32-bit halves are swapped.

7.1 The Internal Reflection Property of GOST

We start with the following observation which is also used in weak attacks on GOST from [35] and to cryptanalyse the GOST hash function Crypto 2008 [33]. Both attacks also exploit fixed points, and can only work for some special (weak) keys. This property is exploited in most (but not all) of our attacks, and in a much more fundamental way: without any fixed points and for arbitrary keys.

Fact 7 (Internal Reflection Property). Consider the last 16 rounds of GOST $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ for one fixed GOST key. This function has an exceptionally large number of fixed points: applied to X gives the same value X with probability 2^{-32} over the choice of X, instead of 2^{-64} for a random permutation.

Justification: This comes from the fact that the state of the cipher after the first 8 rounds $\mathcal{E}$ is symmetric with probability 2^{-32}, and $\mathcal{D} \circ \mathcal{E} = Id$.

Remark: with 2^{32} fixed points, one can expect that one of them will be symmetric, this is exploited in our fastest attack on GOST, see Fact 13.

8 Our Reductions: Methodology and Key Steps

All the attacks described in this paper follow the following quite precise framework for conditional algebraic attacks, which deals with the fundamental question of how we can reduce the complexity of a cipher in cryptanalysis to essentially the problem of breaking the same cipher, with less rounds and less data, at the cost of some "clever" assumptions. We obtain real "black box" reductions.

First a certain number of assumptions on internal variables of the cipher, for one or several encryptions, are made. The probability that these assumptions hold for a random GOST key, needs to be evaluated. Then the probabilities

that, when our assumptions hold, certain well chosen variables in the encryption circuit(s) can be guessed by the attacker, will be estimated. Finally the combination of the assumptions and the guessed values will allow the attacker to obtain a small number of 2,3 or 4 P/C pairs for 8 rounds of the cipher.

In this reduction phase we typically have only one or two P/C pairs for the full 32-bit GOST. Then whatever is the number of P/C pairs obtained for 8 rounds the initial 1 or 2 pairs are insufficient to uniquely determine the key. Thus in all our attacks we have a number of false positives: a certain number of full 256-bit keys which will be considered and checked by the attacker, using a number of additional P/C pairs encrypted with the same key, but for the full 32 rounds. In all our algebraic attacks the total number of false positives (the line before the last in Table 2) can be neglected compared to the overall complexity. This number provides a strong and **information-theoretic** limitation to our attacks. It shows that even if we improved our algebraic key recovery software, an attack on 256-bit GOST faster than 2^{128} is very unlikely.

8.1 Synthetic Summary of All Our Reductions

The following Table 2 on p. 11 gives a synthetic summary of all key steps in the main attacks described in this paper. Weak key attacks appear in Table 3.

Notations: We call X_i, Y_i a certain number of P/C pairs for full 32-round GOST, encrypted with 1 single key (in most of our attacks this could be relaxed and they would also work if pairs come in small clusters encrypted with a single key). We call Z, A, B, C, D etc. certain state values on 64 bits.

9 Attacks On GOST Using 2^{32} Known Plaintexts

Let X_i, Y_i be the set of known plaintexts with $i = 1, 2, 3, \ldots, M$, where $M \approx 2^{32}$. Most attacks in this paper require that the Internal Reflection Property (Fact 7) holds for (at least) one i. This requires at least 2^{32} known plaintexts on average, which means that for some keys we may need a bit less, and for some keys a bit more with $M > 2^{32}$, but rarely much more.

9.1 Reflection-Meet-In-The-Middle Attacks on GOST

Our first requires needs a lot of memory. First we establish that:

Reduction 1. [From 2^{32} KP for 32 Rounds to 2KP for 8 Rounds]
Given 2^{32} random known plaintexts for GOST on average, it is possible to guess two P/C pairs for 8 rounds of GOST (having full 256-bit key) and our guess will be correct with probability 2^{-96}.

Justification: This is done as follows:

1. Let X_i, Y_i be the set of known plaintexts with $1 \leq i \leq M$ and $M \approx 2^{32}$.
2. On average there exists one index $i \leq 2^{32}$ such that $\mathcal{E}^3(X_i)$ is symmetric.
3. We call A be this 64-bit value X_i. So far we don't know i but it can be guessed and A will be immediately determined as $A = X_i$. Thus i, A can be guessed and the guess will be correct with probability about 2^{-32}.

Reduction Summary					
Reduction cf.	Red. 1 §9.1	Red. 2 §10	Red. 3 §11	Red. 4 §11.1	Red 5 §12
Type	1x Internal Reflection		2x Reflection		Fixed Point
From (data 32 R)	2^{32} KP		2^{64} KP		
Obtained (for 8R)	**2** KP	**3** KP	**3** KP	**4** KP	**2** KP
Valid w. prob.	2^{-96}	2^{-128}	2^{-96}	2^{-128}	2^{-64}

Reduction Steps			
0. Assumptions on i	$\mathcal{E}^3(X_i)$ is symmetric	$\mathcal{E}^2(X_i)$ symmetric $\mathcal{E}^3(X_i)$ symmetric	$\mathcal{E}(X_i) = X_i$
Notation for this i	Let $A = X_i$		
	$C = \mathcal{E}^2(A)$		$E = Enc_k(A)$
Observations	$C = Enc_k(X_i)$ because $\mathcal{E}^3(X_i)$ is symmetric		$\mathcal{E}(E) = \mathcal{S}(A)$
Expected ♯ of i	one such i expected for 2^{32} KP	one i on average expected for 2^{64} KP	one i $\in 2^{64}$ KP

1. Guess value	i	C symmetric	i
Determine	A, C	i, A	A, E
Correct	2^{-32}	2^{-32}	2^{-64}

2. Guess value	$B = \mathcal{E}(A)$
Observations	C symmetric cf. Fig. 1
Determine	$Z = Dec_k(B)$
Correct	2^{-64}

3. Guess value	$D = \mathcal{E}^3(A)$	$D = \mathcal{E}^3(A)$
Correct	2^{-32}	2^{-32}

Final Key Recovery					
♯ Pairs 8R	2	3	3	4	2
Pairs obtained			$Z \mapsto A$	$Z \mapsto A$	
	$A \mapsto B$	$A \mapsto B$	$A \mapsto B$	$A \mapsto B$	$A \mapsto A$
	$B \mapsto C$	$B \mapsto C$	$B \mapsto C$	$B \mapsto C$	
		$C \mapsto D$		$C \mapsto D$	$E \mapsto \mathcal{S}(A)$
Valid w. prob	2^{-96}	2^{-128}	2^{-96}	2^{-128}	2^{-64}

Last step	MITM	Guess+ Det. Hybrid MITM-Software/Algebraic			
Cases $\in$ Inside	2^{128}	2^{128}	2^{64}	2^{64}	2^{128}
Then Fact cf.	Fact 9	Fact 4	Fact 5	Fact 6	Fact 4
Time to break 8R	2^{128}	2^{131}	2^{110}	2^{94}	2^{131}
Storage bytes	2^{132}	-	-	2^{67}	-
♯ false positives	2^{224}		2^{192}	2^{128}	2^{192}
Attack time 32 R	2^{224}	2^{227}	2^{238}	$\mathbf{2^{206}}$	2^{222}	$\mathbf{2^{195}}$

Table 2. Summary of our attacks with a black box reduction of the cryptanalysis of full 32-round GOST to a key recovery attack on 8 rounds of GOST with less data.

4. Let C be the encryption of A, $C = Enc_k(A)$. Since $\mathcal{E}^3(A)$Then, as $\mathcal{E}^3(A)$ is symmetric:

$$C = Enc_k(A) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(A))) = \mathcal{D}(\mathcal{E}^3(A)) = \mathcal{E}^2(A). \quad (2)$$

   Thus we obtain one P/C pair of known texts for 16 rounds of GOST.
5. Furthermore we guess also $B = \mathcal{E}(A)$ on 64 bits.
6. Overall with probability 2^{-96} our guess i, B is correct and allows to determine the four correct values i, A, B, C. This gives two P/C pairs for 8 rounds with 256-bit key each, which was our goal: $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$.

From here we present two different methods to recover the key.

9.2 A Reflection-Meet-In-The-Middle Attack with Memory

Fact 8. Given 2 P/C pairs for 8 rounds of GOST, and a few more additional P/C pairs for full 32-rounds of GOST for verification, the correct full GOST key on 256 bits can be determined in time of $1.25 \cdot 2^{128}$ GOST encryptions, and with 2^{132} bytes of memory.

Justification: This is obtained trough a variant of a Meet-in-the-Middle attack with confirmation with additional pairs. First for the first 4 rounds and 128-bit key, in time of $4/32 \cdot 2^{128}$ GOST computations we compute 4 rounds forward for both plaintexts (which are A, B in our attack) and store 2^{128} values on $2 \cdot 64$ bits in a hash table. Then for each second half of the key on 128 bits and in total time of another $4/32 \cdot 2^{128}$ GOST computations we compute 4 rounds backwards and for each of these keys, we expect to get on average 1 corresponding first half of the key from the hash table. Thus we get 2^{128} full 256-bit keys which are checked in the real time with a few extra P/C pairs for the full 32 rounds. Most of the time only one of these is needed to reject them and it takes time of 1 GOST encryption to check. All keys are checked in total time of about $(1 + 8/32) \cdot 2^{128}$ GOST computations and with about 2^{132} bytes of memory.

Now we combine this MITM attack with our Reduction 1 and we get immediately an attack faster than brute force:

Fact 9. Given an average number of 2^{32} random known plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $1.25 \cdot 2^{96+128}$ GOST encryptions, which is $2^{30.7}$ times faster than brute force and with about 2^{132} bytes of memory.

Justification: This is straightforward. We summarize the whole attack.

1. Let X_i, Y_i be the set of 2^{32} known plaintexts.
2. As in Reduction 1, on average there is one index i such that $\mathcal{E}^3(X_i)$ is symmetric. Then a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ can be guessed and the guess will be correct with probability 2^{-96}.
3. In each of 2^{96} cases we apply our MITM attack from Fact 8. Thus we check 2^{96+128} cases and need to perform an equivalent of $1 + 8/32$ full encryptions per case, where the cost of each pre-computation of 2^{128} cases is amortized over each interval containing 2^{128} cases. In each of 2^{96} cases i, B we we

obtain exactly one key, which is checked with on average one and at most a few additional P/C pairs X_i, Y_i. Overall only one correct key is obtained in this attack.

Summary and Discussion. This attack requires 2^{32} known plaintexts, and the running time is $1.25 \cdot 2^{96+128}$ GOST encryptions, which is $2^{30.7}$ times faster than brute force. The storage requirements are however very important: about 2^{132} bytes of fast memory, which need basically to work at the speed of encryption with only 4 rounds of the cipher.

Important: If we consider that today the memory of 2^{30} has a cost comparable to 2^{60} computations, it is possible to believe that the cost of 2^{128} of memory at some moment in the future may be as high as to be equivalent to 2^{256} in computing power. In this case, it is possible to believe that we do **not yet** have a valid attack on GOST. Happily, we are going now to present a more convincing attack, and later also attacks which are strictly faster attacks and yet with very low storage requirements.

Related work: Takanori Isobe from Japan have discovered another MITM attack on GOST in 2011 [34]. However both attacks are **not the same but different**. Our attack is much simpler and slightly faster. It is also true that our attack has been found before their attack was submitted to FSE 2011 in late 2010. In their attack they guess values after 4 and 12 rounds and do a MITM attack on 4+4 rounds, and use an equivalent key technique. In our attack is much simpler we guess one value after 8 rounds and do a MITM attack on 8+8 double rounds in parallel.

More recently, there is more work on MITM attacks on various number of GOST rounds of GOST [50] and on reducing the memory requirements of these attacks [22].

9.3 A Reflection-MITM-Algebraic Attack

The attack is very similar and only the last step changes.

Fact 10. Given an average number of 2^{32} random known plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $2^{96+131} = 2^{227}$ GOST encryptions, which is 2^{28} times faster than brute force. The memory required is negligible.

Justification: This is again straightforward:

1. Again given 2^{32} KP X_i, Y_i, we use the Reduction 1, and on average there is one index i such that $\mathcal{E}^3(X_i)$ is symmetric. Then a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ can be guessed and the guess will be correct with probability 2^{-96}.
2. In each of 2^{96} cases we apply Fact 4 which allows to enumerate 2^{128} keys in total time of 2^{131} GOST computations. The total time spent in this step is $2^{96+131} = 2^{227}$ GOST computations.
3. Overall in this attack we will check 2^{96+128} full keys on 256-bits, most of them being false positives. Each is checked with on average one and at most a few additional P/C pairs X_i, Y_i. The total time spent in this step can be neglected.

Summary. This attack requires 2^{32} known plaintexts, and the running time is 2^{227} GOST encryptions. The storage requirements are negligible.

This attack, though slower, is arguably much better than the previous one and one in [34], which required an inacceptable amount of storage. In what follows we are going to describe better attacks, with lower complexity, and still with negligible storage.

10 A Different Reflection-Algebraic Attack With 2^{32} KP

In this attack the security of GOST will be reduced to the problem of breaking 8 rounds of GOST with 3 known plaintexts (instead of 2 in earlier attacks, see Table 2). It does no longer use the meet-in-the-middle approach.

Reduction 2. [From 2^{32} KP for 32 Rounds to 3KP for 8 Rounds]
Given 2^{32} random known plaintexts for GOST on average, it is possible to obtain three P/C pairs for 8 rounds of GOST and our guess will be correct with probability 2^{-128}.

Justification: Again let X_i, Y_i be our set of approx. 2^{32} known plaintexts. Then:

1. As in Reduction 1, on average there is one index i such that $\mathcal{E}^3(X_i)$ is symmetric. Then a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ can be guessed and the guess will be correct with probability 2^{-96}.
2. We call $D = \mathcal{E}^3(A)$ the value which is symmetric by definition of A. Unlike in our previous attack we also guess D. Overall we get i, A, B, C, D and our guess will be correct with probability 2^{-128}. We have:

$$B = \mathcal{E}(A) \tag{3}$$

$$C = \mathcal{E}^2(A) \tag{4}$$

$$D = \mathcal{E}^3(A) \tag{5}$$

$$D = \mathcal{S}(\mathcal{E}^3(A)) \tag{6}$$

$$C = \mathcal{D}(D) = Enc_k(A) \tag{7}$$

Now we will describe a full attack with complete key recovery.

Fact 11. Given an average number of 2^{32} random knowns plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $2^{128+120}$ GOST encryptions, which is 2^{16} times faster than brute force. The memory required is negligible.

1. We use the Reduction 2 and given 2^{32} KP we obtain a 5-tuple i, A, B, C, D and correct with probability 2^{-128} and 3 P/C pairs for 8 rounds of GOST: $B = \mathcal{E}(A)$ from (3), $C = \mathcal{E}(B)$ from (4), and $D = \mathcal{E}(C)$ from (5). These 8 rounds of GOST depend however on the full 256-bit key. And these 3 P/C pairs do not uniquely determine the key. Moreover, only 64 bits of information about the key are available from one single value i and the information contained in the 3 P/C pairs for 8 rounds of GOST above is largely based on attacker's guesses, and will only be confirmed after a large number of candidates for the full 256-bit GOST key will be generated, and checked against some 4 additional P/C pairs X_j, Y_j for $j \neq i$, see Fact 1.

2. Following Fact 5 (cf. page 7) in each of 2^{128} cases tried and on average, and in total time equivalent to 2^{120} GOST encryptions we obtain 2^{64} candidates for the GOST key k.
3. For each of the 2^{128} cases i, A, B, C, D we get from the program of Fact 5 a uniform enumeration of 2^{64} keys. We get an enumeration of 2^{192} 6-tuples i, A, B, C, D, k. Where k is a candidate for the full 256-bit key. These 6-tuples contain about 2^{192} different candidates for the GOST key k. Each 6-tuple is generated in time of 2^{56} GOST encryptions on average (cf. Fact 5).
   These 6-tuples are checked with 4 extra additional P/C pairs, for example the previous ones X_{i-1}, Y_{i-1} etc. With 5 P/C pairs total, only the right key will be accepted, and the probability that a wrong key is accepted in our attack is 2^{-64}, see Fact 1.
4. Thus we reject all the 2^{192} 6-tuples i, A, B, C, D, k except the correct one which contains the full 256-bit key of the cipher.

Summary. Overall our attack requires 2^{32} known plaintexts, time is 2^7 times faster than brute force which requires 2^{255} GOST encryptions on average. It requires negligible storage, except for the 2^{32} known P/C pairs.

11 Attacks On GOST Using 2^{64} Known Plaintexts

Now we are going now to describe a better attack where we are still going to reduce the security of GOST to the problem of breaking 8 rounds of GOST with 3 known P/C pairs where our guess will be valid with a higher probability. This however will be obtained at a price of 2^{64} known plaintexts (instead of 2^{32} KP). This larger quantity of data is required if order to find cases where the internal reflection (cf. Fact 7) occurs twice, in order to be able to analyse several encryptions at the same time, which will reduce the number of false positives.

First we consider special plaintexts which are likely to occur in practice:

Assumption 1. Let A be such that both $\mathcal{E}^2(A)$ and $\mathcal{E}^3(A)$ are symmetric.

Fact 12 (Key Property). There is on average one value A which satisfies Assumption 1 above. For 63% of all GOST keys at least one such A exists.
Justification: We have 2^{64} possibilities, each time the probability is 2^{-64}. Such a value A exists for $1-(1-1/N)^N \approx 63\%$ of all GOST keys where $N = 2^{64}$.
Remark: For 37 % of keys this attack fails but our earlier attacks requiring only 2^{32} KP still work.

Reduction 3. [From 2^{64} KP for 32 Rounds to 3KP for 8 Rounds]
Given 2^{64} known plaintexts for GOST, it is possible to obtain three P/C pairs for 8 rounds of GOST and our guess will be correct with probability 2^{-96}.
Justification: will be provided below.

1. Let X_i, Y_i be the set of all the 2^{64} known plaintexts.
2. On average there exists one index such that both $C = \mathcal{E}^2(X_i)$ $D = \mathcal{E}^3(X_i)$ are symmetric values on 64 bits. Then since $D = \mathcal{E}^3(A)$ is symmetric we have

$$Enc_k(A) = C = \mathcal{E}^2(A) \tag{8}$$

So far we don't know neither i nor A, C, D. However since from our Key Assumption on i the value of $C = \mathcal{E}^2(X_i)$ must be a symmetric value on 64-bits, we can limit ourselves to select C among all symmetric ciphertexts, guess $C = Y_i$ and our guess is true with probability 2^{-32}. Let $A = X_i$ be the corresponding plaintext. We have a triple i, A, C which is correct with probability 2^{-32}.

3. Then we guess B and get a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ and our guess will be correct with probability 2^{-96}.
   As in our first two attacks we don't try to guess D.
4. This gives exactly 2 P/C pairs for 8 rounds $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$.
5. One extra pair will be obtained by decrypting B as follows. We define Z as $Z = Dec_k(B) = \mathcal{E}^{-3}(\mathcal{S}(\mathcal{E}(B)))$. We have
$$Z = Dec_k(B) = \mathcal{E}^{-3}(\mathcal{S}(C)) = \mathcal{E}^{-3}(C) = \mathcal{E}^{-2}(B) = \mathcal{E}^{-1}(A) = \mathcal{D}(A).$$
   Where we used our assumption that $C = \mathcal{E}^2(A)$ is symmetric, and we get that $Z = Dec_k(B) = \mathcal{D}(A)$. Thus we get our 3-rd pair $A = \mathcal{E}(Z)$. This decryption is done in constant time if we assume that all the pairs X_i, Y_i are stored using a hash table.
6. Thus we determine i, Z, A, B, C and we get 3 known P/C pairs for 8 rounds of GOST, and our guess is valid with probability 2^{-96}.

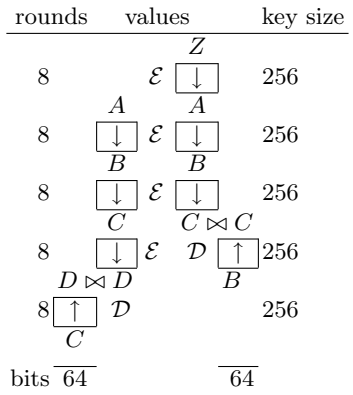

Fig. 1. An attack on GOST with double reflection

Thus we obtain the following result:

Fact 13. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{26} GOST encryptions. The storage required is 2^{64} times 8 bytes.

Justification: As above we get 3 known P/C pairs for 8 rounds of GOST, and our guess is valid with probability 2^{-96}. For each of the 2^{96} cases i, A, B, C we get from the program of Fact 5 a uniform enumeration of 2^{64} keys. Thus we get

an enumeration of 2^{160} 5-tuples i, A, B, C, k. Where k is a candidate for the full 256-bit key. These 5-tuples contain about 2^{160} different candidates for the GOST key k. Each 5-tuple i, A, B, C, k is generated in time of 2^{46} GOST encryptions on average (cf. Fact 5). These 4-tuples are checked with 4 extra additional P/C pairs. We reject all the 2^{160} 4-tuples i, D, B, k except the correct one. Total cost is about $2^{160+46} = 2^{206}$ GOST encryptions.

11.1 Alternative Attacks with Reduction to 4 Pairs

If we look at Reduction 3 it is possible to see that by guessing D we are able to obtain 4 pairs with a degraded probability as follows:

Reduction 4. [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}.

If we compare Fact 6 to 5 we gain a factor of about 2^{16} however we lose a factor of 2^{-32} to obtain 4 pairs instead of 3. This attack is summarized in the next to next to last column in Table 2 and we obtain 2^{222} GOST encryptions overall. In Appendix we present three other and different methods to obtain 4 pairs given 2^{64} KP with the same and even slightly better success probability.

12 A Simple Fixed Point Attack With 2^{64} KP

So far all single-key attacks on GOST ever found exploited the internal reflections [34] and in our best attack on GOST we use this reflection property twice. However there is another very simple attack on GOST which does not use any reflection and where no symmetric 64-bit values appear. This shows that GOST is broken independently of reflection attacks [34, 35].

Reduction 5. [From 2^{64} KP for 32 Rounds to 2KP for 8 Rounds]
Given 2^{64} known plaintexts for GOST, it is possible to obtain two P/C pairs for 8 rounds of GOST and our guess will be correct with probability 2^{-64}.

Justification: This attack is very simple and appears in the last column of Table 2. Let A be a fixed point of $\mathcal{E}$. One on average such value exists. Then let $E = Enc_k(A)$, and since A is a fixed point for 8 rounds, and $Enc_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E}^3$ after 24 rounds we still have A, and we obtain an additional pair for 8 rounds $\mathcal{E}(E) = \mathcal{S}(A) = \overline{A}$. Both these pairs are jointly valid with probability 2^{-64}, when A is correct.

This can be used to break GOST directly with the same complexity as in Fact 11:

Fact 14. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{195} GOST encryptions. The storage required is 2^{64} times 8 bytes.

Justification: We combine Reduction 5 and Fact 4. The number of false positive keys will be 2^{192} and can be neglected.

Important Remark This attack will work for about 63 % of all GOST keys for which $\mathcal{E}$ has a fixed point. For the remaining 37 % of Family B keys this

attack fails, but the attack described in Section 11 will still work with roughly the same complexity, this for 63 % of these 37 % of keys.

This paper provides strong motivation for doing more research on this type solver technology, which is known to be able to break more or less any cipher with a limited number of rounds, see [10], and the main contribution here is to be able to reduce the security of a cipher with 32 rounds to the security of the cipher with 8 rounds.

12.1 An Improved Fixed Point Attack With 2^{64} KP

An alternative final step for the same reduction and attack was very recently proposed in [22]: the overall running time is then reduced from 2^{195} (and in older versions of this paper it was 2^{216}) to about 2^{192}.

13 Discussion

13.1 Interesting Points About Our Attacks

We obtained several attacks which break the full 32-round GOST faster than by brute force and require small storage. Crucial ingredients in these attacks are:

A) A self-similarity property of the full 32-rounds GOST which allows to reduce the problem of breaking a 32 rounds cipher to a problem of breaking a cipher reduced to 4 or 8 rounds. Most our attacks use the Internal Reflection Property (cf. Fact 7) and an overall small number of iterations of the 8-round block $\mathcal{E}$. We heavily rely on the fact that the same large encryption block with the same key is repeated, we call it **"strong self-similarity"**. The attack described in Section 10 is a very innovative new type of attack on block ciphers based on strong self-similarity and reflection, yet it is not a slide attack [26, 2, 3, 1] neither it exploits fixed points like in [11, 35]. In addition in Appendix A and Appendix D we show two attacks on GOST which are faster than brute force, and don't use any reflection.

B) The second necessary ingredient is the existence of efficient and low data complexity [10, 21] key recovery attacks on reduced-rounds of GOST. For example 8 rounds of GOST with 256-bit key can be broken in time of 2^{120} GOST encryptions and only 3 KP, cf. Fact 5. This is possible due to several factors. First, the diffusion in the cipher is poor, which is known to play an important role in this type of algebraic attack. Secondly, both the GOST S-boxes (mainly due to their size, see [8, 10, 12, 4, 5], much less due to any particular choice of S-boxes), and the addition modulo 2^{32} contribute to a circuit of GOST which is overall not too complex compared to any other comparable cipher, see [37], and this also makes it vulnerable to Algebraic Cryptanalysis [4, 8, 12, 42, 43, 11].

It is worth noticing that we do not exploit any other property or weakness of GOST other than A) and B) above, and we are able from these properties to construct a dozen of very diverse and rather non-trivial attacks on GOST.

13.2 Algebraic Complexity Reduction vs. Black Box Reductions

All except one algebraic complexity reductions in this paper are black box reductions. However the concept of an algebraic complexity reduction is more general and reductions do not have to be black box. Another example of attack with algebraic complexity reduction which is not a black box reduction is the Slide-Algebraic Attack 2 in [11], here the attacker ends up with two instances of a slightly different cipher with a shared key.

13.3 Beyond GOST

This paper demonstrates the power of algebraic cryptanalysis (AC) which is a disruptive technique in cryptanalysis leading to great many new attacks. In theory algebraic attacks allow to break more or less any cipher, provided it is "not too complex", however it turns out that it can break only a few rounds, up to about 6, 7 or 8, of modern ciphers such as DES [10] or GOST (this paper). Then, if the cipher has special properties at the high level, like in KeeLoq [11] or in GOST (this paper), one can do indeed much more.

In particular self-similarity properties are very powerful because they allow a dramatic reduction in the overall complexity of the algebraic description of the problem. In what we call strong self-similarity whole very large blocks of the cipher are eliminated in one step by assuming that all the bits inside are identical to the whole internal state of another large block. We can also note that if we had approximate self-similarity, this type of attack would work extremely well. with an additional factor in time complexity.

Many block ciphers have various self-similarity properties and a relatively simple key schedule. In the past these have been overlooked or used only in very special attacks such as attacks which exhibit weak keys or attacks which use several related keys. If keys are generated at random, these attacks have a negligible impact on the real life applications of ciphers as far as confidentiality is concerned. In this paper however, in a similar way as for example in various attacks on KeeLoq [11] and elsewhere [26, 2, 3, 1], we show that these quite strong properties are dangerous and really allow to break ciphers. This is by a black box reduction to a software algebraic attack on a reduced-round sub-component of a bigger cipher. This will work but only if the key schedule is indeed "not too complex", and allows large blocks of the circuit to be identical with high probability, and if the algebraic attacks are powerful enough to work below a certain threshold.

We contend that this combination of self-similarity attacks and algebraic attacks is quite unique. In the most basic form of slide attacks [26, 2, 3] (which is maybe the simplest form of self-similarity attacks) the attacker reduces the security of a cipher for a large number of rounds to a security of essentially the same cipher with much less rounds, and he is able to generate a large (or unlimited) quantity of known P/C pairs for a simpler component. Thus many different attacks are applied in the literature as the last step of a slide attack, and not surprisingly there are so many different attacks on KeeLoq, see [11]. However in advanced self-similarity attacks like in this paper and in [11], one can generate only a very limited number quantity of P/C pairs for the smaller component.

For example we can have 3, and with a lot more effort we can have 4, cf. Section 11.1, but probably by no means we could have 5, which would already require a number of assumptions which cannot be afforded by the attacker (or weak keys cf. Fact 25). Very few cryptographic attacks are able to deal with such small quantities of encrypted data: brute force attacks, guess then determine attacks, meet-in-the middle attacks and notably algebraic attacks (cf. also [21]). Then if the components of the cipher have low resistance to algebraic attacks, below a certain threshold, an algebraic attack with self-similarity could potentially become the best attack known on the given cipher. This is what we currently obtain for GOST. As algebraic attacks develop, this could happen for some other block ciphers with over-simplistic key schedule and strong-self similarity.

13.4 Should GOST Become An International Encryption Standard?

In 2010 GOST was submitted to ISO to become a worldwide encryption standard. From the cryptography research point of view we have broken GOST, and many algorithms have been rejected by various standardization bodies for much less than an actual key recovery attack faster than brute force. Does it mean that GOST should not be used?

In practice, in a pragmatic perspective, GOST with full 256-bit keys generated at random remains still impossible to decrypt in practice. It remains a particularly economical cipher in terms of gate count complexity in hardware implementation, cf. [37], and thus suitable for resource-constrained environments such as smart cards and RFID.

From the standardization point of view however, given the fact that academic standards for block ciphers tend to be very high, and a provision should be made for further improvements in cryptanalysis, GOST should not be used in applications which require high security. In particular, it should never be used by banks (at least two sets of GOST S-boxes have been explicitly identified as being used by Russian banks cf. [46, 31]). Very few encryption algorithms have ever been standardized by ISO. The 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. To summarize, it is clear that ISO should not standardize GOST, as this algorithm is structurally flawed, and does not provide the security level required by ISO.

14 Conclusion

The Russian encryption standard GOST is implemented in OpenSSL and other crypto libraries [31, 49], used by Russian banks, and increasingly also on the Internet. It appears that GOST has a lower gate count than any comparable cipher, cf. [37]. In 2010 GOST was submitted to ISO to become an international standard. Given the 256-bit key size of GOST, and the large number of 32 rounds, GOST is expected to remain secure for many decades to come. Until 2011, no shortcut attack allowing to recover individual GOST keys faster than brute force was found.

The general idea of Algebraic Cryptanalysis (also known as the method of "Formal Coding") has been around for more than 60 years [47, 32]. 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 [8, 5, 6]. However for block ciphers only a few rounds can be broken, see [10]. Only one full-round block cipher KeeLoq could so far be shown to be weak enough, to be broken using an algebraic attack [11]. This was due to self-similarity of large encryption blocks in this cipher. The same sort of self-similarity is found in the Russian GOST. Can we break full 32-round GOST by an algebraic attack?

In this paper we introduce a general framework which allows one to reduce an attack on a block cipher with many rounds to an attack on a cipher with less rounds. We call it **Algebraic Complexity Reduction**. In order to achieve our complexity reduction we need to solve a certain combinatorial puzzle. With well-chosen equalities on internal values, we are able to literally break the cipher apart into smaller pieces. This greatly reduces the complexity of the cipher as a circuit. Unhappily this methodology leads to very small quantity of data available for the reduced cipher. Very few low-data complexity attacks are known: mostly software "algebraic attacks" and meet in the middle attacks (MITM). Our best attacks are now obtained by combination of these techniques. To summarize our work brings powerful and disruptive new techniques in cryptanalysis leading to great many new attacks: new methods to achieve reduction to low-data complexity attacks, and new methods to deal with these low-data attacks. For regular GOST keys in this paper we proposed five non-trivial black-box complexity reductions on full GOST which are summarized in Table 2 on page 11.

In this paper we considerably enlarge the spectrum of self-similarity attacks on block ciphers. Our attacks generalize many already known fixed point, sliding, reflection and involution attacks. We are are able to exploit similarities of individual sub-blocks and their inverses. We are the first to propose attacks with double and even triple reflection. We are able to relax the conditions necessary in slide attacks [26, 2, 3, 1, 25] in nearly arbitrary ways. We use fixed points in innovative ways, different than in previous fixed point attacks (cf. [11, 35]). We have found four distinct valid attacks on GOST which don't use any reflections [34, 35] whatsoever.

In this paper we present more than 10 different new attacks on GOST which are faster than brute force plus more than 10 additional attacks on special weaker key classes. For single key attacks, six of our attacks are shown in Table 2 on page 11 and many other in the Appendix. The fastest single key attack on GOST in this paper requires 2^{64} known plaintexts and 2^{195} GOST computations.
The fastest single-key attack on GOST known is in 2^{178} cf. [19].

The most recent results (May 2012 update of this paper and [22]) show that the final step of our attacks can be greatly improved. Since 2012 we can observe the the best current attacks are not anymore experimental and simple software algebraic attacks. We have now discovered better attacks which require a much more careful analysis of the internal connections inside the cipher. In most cases the best (updated) final step for our attacks is now a software attack with SAT solver where the first step works as a low-memory multi-dimensional MITM-like attack and where very low memory requirements and faster running times are achieved due to an experimental software attack.

In addition we have done an extensive study of new weak key classes in GOST which are made possible by our methodology. The results are summarized in Table 3 on page 46. Many of these new weak key classes occur with probability which is high enough, or/and lead to significantly faster attacks than with regular keys. In fact two of these attacks **beat best regular attack on running time** also when the keys are not weak. We consider a realistic scenario of encryption with a population of devices with random diversified keys, with weak keys occurring naturally. Then we can break full GOST in overall **total** time of about $\mathbf{2^{159}}$, which includes checking all the keys and breaking one of the weaker keys, see Appendix L. It becomes the best attack ever found on GOST in the multiple-key scenario. Ciphers are not used in practice with single keys, on the contrary, and our 2^{159} is arguably a better and more realistic attack than any other attack on GOST ever found, including 2^{192} by Shamir *et al* [22] and 2^{178} by Courtois [19].

In this paper, in addition we also present four very nearly practically feasible attacks on some major variants of GOST. Both "natural" methods of using GOST with 128-bit keys which were previously suggested in the literature [3] are shown to be broken. With the "inversed" method we can recover arbitrary 128-bit keys within 2^{81} GOST encryptions and given 2^{35} CP, cf. Fact 43. With the "direct" method we can identify and break only certain (weak) keys in overall time of 2^{65} GOST encryptions, and 2^{32} CP, cf. Fact 33 in Appendix M. Which again leads to two attacks on **arbitrary random** 128-bit keys which can be recovered in total time of 2^{66} GOST encryptions per key. These and some additional results are summarized in Table 5 on page 61.

All our attacks are expected to work for any choice of GOST S-boxes. Since most of our attacks require very large quantities of data and the time complexities remain astronomical, they do not threaten practical applications of GOST with random 256-bit keys. However our methods and results may have a very significant impact on decryption of messages encrypted with some weaker variants of GOST.

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. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
5. 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`
6. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
7. Nicolas Courtois *CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited* Available at `http://eprint.iacr.org/2007/152/`.
8. 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.
9. 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/`.
10. 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/`.
11. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
12. 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.
13. Nicolas Courtois: *Security Evaluation of GOST 28147-89 In View Of International Standardisation,* in Cryptologia, Volume 36, Issue 1, pp. 2-13, 2012. An earlier version which was officially submitted to ISO in May 2011 can be found at `http://eprint.iacr.org/2011/211/`.
14. Nicolas Courtois: *Low-Dimensional Key Recovery Attacks On 8 Rounds of GOST,* preprint, May 2012.
15. Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* In CECC 2011, 11th Central European Conference on Cryptology, post-proceedings in preparation.
16. Nicolas Courtois, Michał Misztal: *First Differential Attack On Full 32-Round GOST,* in ICICS'11, pp. 216-227, Springer LNCS 7043, 2011.
17. Nicolas Courtois, Michał Misztal: *Differential Cryptanalysis of GOST,* In Cryptology ePrint Archive, Report 2011/312. 14 June 2011, `http://eprint.iacr.org/2011/312`.
18. Nicolas Courtois: *Faster Attacks on Full GOST,* A short presentation given at FSE 2012 rump session, available at `http://fse2012rump.cr.yp.to/9c19b743f2434a74b3a0d3e281b52b01.pdf`.

19. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* In Cryptology ePrint Archive, Report 2012/138. 15 March 2011, `http://eprint.iacr.org/2012/138`.
20. Nicolas Courtois, Theodosis Mourouzis: *Black-Box Collision Attacks on the Compression Function of the GOST Hash Function,* appears in 6th International Conference on Security and Cryptography SECRYPT 2011.
21. 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/`.
22. Itai Dinur, Orr Dunkelman and Adi Shamir: *Improved Attacks on Full GOST,* in FSE 2012 proceedings, early version from October 2011 available at `http://eprint.iacr.org/2011/558/`.
23. 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.
24. 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.
25. Soichi Furuya: *Slide Attacks with a Known-Plaintext Cryptanalysis,* In ICISC 2001, LNCS 2288, 2002, pp. 11-50.
26. 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.
27. 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
28. 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 [29].
29. An English translation of [28] by Aleksandr Malchik with an English Preface co-written with Whitfield Diffie, was published in 1994 `http://www.autochthonous.org/crypto/gosthash.tar.gz` or at `193.166.3.2/pub/crypt/cryptography/papers/gost/russian-des-preface.ps.gz`
30. 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`
31. 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/`
32. J. Hulsbosch: *Analyse van de zwakheden van het DES-algoritme door middel van formele codering,* Master thesis, K. U. Leuven, Belgium, 1982.
33. 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.
34. Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011.
35. Orhun Kara: *Reflection Cryptanalysis of Some Ciphers,* In Indocrypt 2008, LNCS 5365, pp. 294-307, 2008.

36. John Kelsey, Bruce Schneier, David Wagner: *Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES,* In Crypto'96, pp. 237-251, LNCS 1109, Springer, 1996.
37. 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.
38. 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.
39. Random Permutation Statistics – wikipedia article, 22 January 2008, available at `http://en.wikipedia.org/wiki/Random~permutation~statistics`.
40. J.-J. Quisquater and Y. Desmedt and M. Davio: *The Importance of 'good' Key Scheduling Schemes (How to make a secure DES scheme with* $\leq$ *48 bit keys?,* In Crypto'85, LNCS 218, pp. 537–542, Springer, 1985.
41. Vladimir Rudskoy: *On zero practical significance of Key recovery attack on full GOST block cipher with zero time and memory,* Preprint 31-Mar-2010, `http://eprint.iacr.org/2010/111`
42. Igor Semaev: *Sparse Algebraic Equations over Finite Fields,* SIAM J. Comput. 39(2): 388-409 (2009).
43. 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/`
44. Markku-Juhani Saarinen: *A chosen key attack against the secret S-boxes of GOST,* unpublished manuscript, 1998.
45. 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.*
46. *Bruce Schneier: Section 14.1 GOST,* in *Applied Cryptography,* Second Edition, John Wiley and Sons, 1996. ISBN 0-471-11709-9.
47. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in particular page 704.
48. Niklas Sörensson, Niklas Eén, Mate Soos: CryptoMiniSat 2.92, an open-source SAT solver package based on earlier MiniSat software, at `http://www.msoos.org/cryptominisat2/`
49. Wei Dai: Crypto++, a public domain library containing a reference C++ implementation of GOST and test vectors, `http://www.cryptopp.com`
50. Bo Zhu and Guang Gong: Multidimensional Meet-in-the-Middle Attack and Its Applications to GOST, KTANTAN and Hummingbird-2, Cryptology ePrint Archive: `eprint.iacr.org/2011/619/`

A An Alternative Reduction With 2^{64} KP and Without Internal Reflections

In this paper we presented 3 attacks with Internal Reflection and two more attacks where such reflection occurs twice. Reflection occurs frequently because the last 16 rounds of GOST have a large number of 2^{32} fixed points, which is instrumental in most of our attacks. Here we provide yet another, very surprising method to obtain 4 pairs given 2^{64} KP, and with the same success probability of 2^{-128} as in Reduction 4. The method however is very different and this will be the only attack in this paper which **does NOT use any internal reflection** and where no symmetric 64-bit values appear. This attack is rather a new and peculiar form of a slide attack, and it is somewhat reminiscent of certain fixed point attacks [11], except it uses points of type $\mathcal{E}(D) = \overline{D}$ where by definition $\overline{D}$ is the value on 64-bits with both 32-bit halves exchanged. Similarly as for other attacks in this paper the black box reduction stage is non-trivial: it is not clear if such attacks should exist at all for any given block cipher. This attack is also described in [13].

Here we consider plaintexts with another very peculiar property:

Assumption 2 (Assumption W). Let A be such that $\mathcal{E}(D) = \overline{D}$ where D is defined as $D = \mathcal{E}^3(A)$.

Again, it is possible to see that:

Fact 15 (Property W). Given 2^{64} KP there is on average one value A which satisfies the Assumption W. For 63% of all GOST keys at least one such A exists.

This property has some very important consequences:

Fact 16 (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. 2.
2. $Enc_k(B) = C$ This can be seen on the left hand side of Fig. 2.

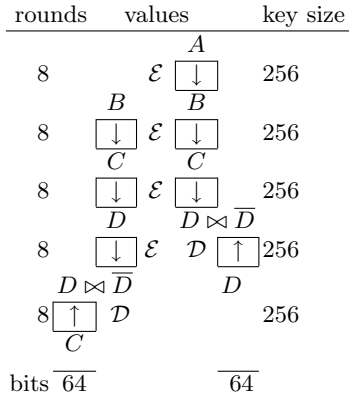

Fig. 2. An alternative attack with reduction to 4 pairs and no internal reflection

This leads directly to our new reduction:

Reduction 6. [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. 2. For each (A, B) this computation of (C, D) is done in constant time if we assume that all the pairs X_i, Y_i are stored using a hash table.

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

A.1 How to Use This Reduction 6 to Break GOST

Both our previous Reduction 4 and this new Reduction 6 described here achieve exactly the same result by a different method.

Thus that attack which uses this reduction will also be exactly the same as the attack given in Section 11 with additional guess of D, as described in Section 11.1 which is also summarized in the next to last column in Table 2 and the complexity is exactly the same: 2^{222} GOST encryptions.

Summary. Overall our attack requires 2^{64} known plaintexts, time is 2^{22} times faster than brute force. The storage required is for the 2^{64} known P/C pairs.

A.2 Can One Do Better?

In Section 11 of this paper we presented one attack which obtains 3 KP for 8 rounds which will be correct with probability 2^{-96} and 4 KP could be obtained in Section 11.1 with probability 2^{-128}, and by a second alternative method in this Appendix A. An interesting question is whether these results can be improved. For typical GOST keys the answer is probably no. However these probabilities can be further quite substantially improved for particular (still quite large) classes of weak keys. For example if we have a diverse population of GOST keys where 2^{-32} will be weak, for such weak keys one can then obtain 4 KP for 8 rounds which will be correct with probability 2^{-64} instead of 2^{-128}, see Fact 25. Moreover this probability can be reduced to almost certainty, about 2^{-1} if we allow a further reduction in the number of weak keys considered, and 2^{-64} of all keys will be weak, see Fact 28.

A.3 Generalizations of Reduction 6

This attack does not use the peculiar (weak) nature of $\mathcal{S}$ which is not only an involution (it is equal to its own inverse function) but has as many as 2^{32} fixed points, which we do not require. It only exploits the (weaker) property that the last 16 rounds of GOST are an involution. Then it still uses some sort of fixed points, for the function $\mathcal{S} \circ \mathcal{E}$, which does not have a particularly large number of such fixed points, just 1 on average. It is possible to see that this attack would also work for any cipher defined as $Enc_k = \mathcal{D} \circ \mathcal{F} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E}$ where $\mathcal{F}$ is an arbitrary involution which could also depend on some cryptographic key, potentially even a key independent on the key used in $\mathcal{E}$, thus increasing the key space. In this sense it clearly is a stronger and more general attack.

B One More Reduction To 4 KP Without Internal Reflections

There is also another method to get the same result as in Reduction 6,

We combine Reduction 5 to get 2 pairs for 8 rounds, then we use the amplification property of Fact 18 to get one more pair for 16 rounds, and we guess 64-bits in the middle of it. Thus we get 4 pairs for 8 rounds which are with very high probability distinct.

As before, this is yet another attack on GOST faster than brute force. However as we will see below, it is better and more productive to just apply Reduction 5 twice, which will lead to a slightly faster attack.

C Another Cheaper Reduction With 2^{64} KP and Without Internal Reflections

In this paper we presented three black-box reductions allowing to produce 4 P/C pairs for 8 rounds of GOST, given 2^{64} KP, and at the price of making an assumption which holds with probability 2^{-128}: these are Reduction 4, Reduction B and Reduction 6.

In this section and in the next section we present two more such reductions. which both have a slightly better success probability: 2^{-127} instead of 2^{-128}. All these reductions lead to attacks which will in 2^{222} or 2^{221} faster than brute force, (like in Section 11.1 and the next to last column in Table 2). However these are not the fastest of our attacks, see Section A.2. Therefore their interest is (for now) purely academic. It is also another two attacks which do NOT use any internal reflection. All these reductions are very different and work for a majority but not all GOST keys, for example for 63 % of keys, or less, and different reductions work for different keys, and therefore they complement each other.

This reduction is very simple and we essentially need to apply Reduction 5 twice:

Reduction 7. [From 2^{64} KP for 32 Rounds to 4KP for 8 Rounds]
We assume that $\mathcal{E}$ has two fixed points, which occurs with probability about 26%.
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^{-127}.

Justification: Let $\mathcal{E}$ be such that it has two or more fixed points, which occurs with probability $1-(1-1/N)^N-\binom{N}{1}(1-1/N)^{N-1}(1/N)^1 \approx 1-2/e \approx 26\%$, where $N=2^{64}$, see [11, 39]. We can apply Reduction 5 twice and guess two fixed points A and A' for $\mathcal{E}$. However the probability to guess 2 fixed points for $\mathcal{E}^2$ is only about 2^{-127} instead of 2^{-128}, this is because if A, A' is a correct guess on 128 bits, A', A is also correct.

Again this can be used to break GOST directly in the same way as before, we apply Fact 6 and compute the key in time of 2^{94} GOST encryptions.

Fact 17. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{232} GOST encryptions. The storage required is 2^{64} times 8 bytes.

Summary. Thus we obtained another attack with 2^{64} KP, but time is now 2^{23} times faster than brute force.

D Yet Another Cheaper Reduction To 4 KP

Here is another black-box reduction allowing to produce 4 P/C pairs for 8 rounds of GOST making an assumption which also holds with probability 2^{-127}. All these attacks work for a different (quite large) fraction of all GOST keys, but not all GOST keys, and complement each other. It can be seen as a slight variant of Reduction C which gives in some cases identical, and in some cases different cases (there is a non-trivial intersection of both attacks).

Reduction 8. [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^{-127}.

Justification: We consider the initial 16 rounds $\mathcal{E}^2$. On average it has 2 fixed points and it is easy to see that points of order two for $\mathcal{E}$ come in pairs. Similarly as before, the probability to guess 2 fixed points for $\mathcal{E}^2$ is about 2^{-127} instead of 2^{-128}, this is because if X, Y is a correct guess on 128 bits, Y, X is also correct.

Then we proceed as follows: This gives us immediately Z and T as shown on Fig. 3. For each (X, Y) this computation of (Z, T) is done in constant time if we assume that all the pairs X_i, Y_i are stored using a hash table.

Thus we obtained 4 pairs for 8 rounds of GOST:
$X \mapsto Y, Y \mapsto X, Z \mapsto \overline{Y}, T \mapsto \overline{X}$.

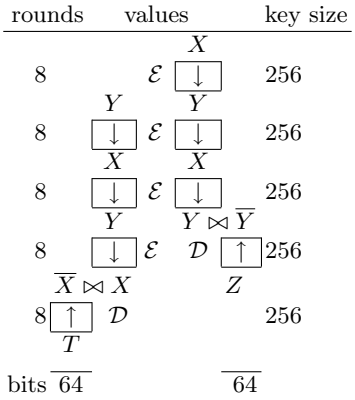

Fig. 3. A slightly cheaper alternative attack with no internal reflection

Resulting Attack. Again, if we combine this with Fact 5 we get an attack which breaks GOST given 2^{64} known plaintexts, time is also 2^{23} times faster than brute force (as in Fact 17). The storage required is for the 2^{64} known P/C pairs.

E Involution Property For 16 Rounds of GOST and Amplification Property for Full GOST

We discovered a peculiar amplification-like property of GOST. which is closely related to most of the attacks described in this paper and reminiscent of slide attacks [26, 2] and yet it is different than any slide attack known to us. It is based on the fact that the last 16 rounds of GOST are **an involution**, i.e. a special sort of permutation where all cycles are of length 1 or 2.

Basic slide attacks [26, 2, 3] where the encryption process is assumed to be perfectly periodic, and to be a straightforward periodic iteration of one single key-dependent function f_k, operate as follows. The attacker assumes that he knows one P/C pair for this function f_k, and then uses the sliding property to obtain an additional P/C pair for f_k. This process can be iterated and generate many P/C pairs for f_k. In contrast, in more advanced self-similarity attacks like in this paper and in [11], some of which are considered to be "advanced" slide attacks, and some of which use some special points such as fixed points, the process cannot be continued and one can generate only a very limited number quantity of P/C pairs for the smaller component.

In GOST cipher the periodicity which is very helpful in slide attacks [26, 2, 3] is deeply broken by the inversion of keys which occurs in the last 8 rounds. However an analogous property for GOST still exists.

Fact 18 (Amplification Property for Full 32-rounds GOST). For any X, Y we have:

$$Y = \mathcal{E}^2(X)$$
$$\Updownarrow$$
$$Enc_k(X) = \mathcal{E}^2(Dec_k(Y)).$$

Given access to both encryption and decryption oracles for the full GOST $Enc_k(\cdot)$ For each P/C pair for 16 rounds of GOST $Y = \mathcal{E}^2(X)$ such that $\mathcal{E}^3(X)$ is not symmetric, the attacker can obtain another **different** P/C pair $Z = \mathcal{E}^2(T)$ for 16 rounds of GOST with $Z = Enc_k(X)$ and $T = Dec_k(Y)$.

Justification: This property is due to the fact that the last 16 rounds of GOST are an involution. From our initial pair $Y = \mathcal{E}^2(X)$ such that X we obtain a pair $Y, Enc_k(X)$ for the last 16 rounds of GOST which is $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$. Indeed:

$$Enc_k(X) = \left(\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}^3\right)(X) = \left(\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}\right)(Y)$$

However this function $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ representing the last 16 rounds of GOST is an involution, therefore also $Enc_k(X), Y$ is a valid pair:

$$Y = \left(\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}\right)(Enc_k(X))$$

Now we decrypt both sides to obtain:

$$Dec_k(Y) = \left(\mathcal{D}^3 \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{D} \circ \mathcal{S} \circ \mathcal{E}\right)(Enc_k(X)) = \mathcal{D}^2(Enc_k(X))$$

Therefore we have:

$$Enc_k(X) = \mathcal{E}^2(Dec_k(Y))$$

Now we need to see under which condition the pair Z, T is distinct from the initial pair $Y = \mathcal{E}^2(X)$. This happens if and only if $Enc_k(X) \neq Y$. Equivalently when

$$(\mathcal{D} \circ \mathcal{S} \circ \mathcal{E})(Y) \neq Y$$

which in turn is equivalent to $\mathcal{S}(\mathcal{E}(Y)) \neq \mathcal{E}(Y)$ which occurs if and only if $\mathcal{E}(Y) = \mathcal{E}^3(X)$ is **not** symmetric.

E.1 Additional Remarks on Amplification in GOST

In Fact 18, each time $\mathcal{E}^3(X)$ is not symmetric, and given one pair for 16 rounds $Y = \mathcal{E}^2(X)$ we obtain another distinct pair for 16 rounds $Z = \mathcal{E}^2(T)$ where by definition $Z = Enc_k(X)$ and $T = Dec_k(Y)$.

Unhappily this process cannot be iterated. By reasoning from one assumption on 16 bits we can infer at maximum one another distinct pair, and then we immediately enter a cycle of length 2:

$$Y = \mathcal{E}^2(X)$$
$$\vdash$$
$$Enc_k(X) = \mathcal{E}^2(Dec_k(Y))$$
$$\vdash$$
$$Enc_k(Dec_k(Y)) = \mathcal{E}^2(Dec_k(Enc_k(X)))$$

The third pair is identical to the first. Moreover it was already shown that if $\mathcal{E}^3(X)$ is symmetric, and only in this case, all these pairs for 16 rounds are identical (we have a fixed point in our inference process of Fact 18).

Remark: In many attacks studied in this paper we have $\mathcal{E}^3(X)$ which is symmetric, a reflection occurs in the involution function $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$. In these attacks this method of Fact 18 does exceptionally not work and gives the same pair X, Y. In most other cases, and with overwhelming probability of $1 - 2^{-32}$, $\mathcal{E}^3(X)$ is not symmetric, and our method of Fact 18 is guaranteed to work.

E.2 The Amplification Paradox

Our amplification property is not very dangerous, no attack really exploits it.

Now imagine that we have a second property like this. Then we could combine both these properties to generate an unlimited number of P/C pairs for 16 rounds, probably the whole code-book, starting from one single assumption on 16 rounds, which is quite affordable to make for the attacker (64 bits need to be guessed). And maybe even generate 2^{32} pairs, make some of the inputs repeat by the birthday paradox, realize that the predicted outputs are different, which would prove that there was a contradiction, proving that the initial assumption on 64 bits was incorrect (this cannot be guaranteed). We call this "amplification paradox": one such property is not very dangerous, two would be a source of very powerful attacks, which transform the security of GOST with 32 rounds and broken/imperfect periodicity, to 16 rounds of GOST with perfect periodicity, which will be therefore much easier to break by various slide fixed point, cycling and other attacks.

F Black-box Reductions from 32 to 16 Rounds

Most black-box reductions in this paper are reductions from 32 to 8 rounds. In this section we study very briefly the question of black-box reduction to 16 rounds. We don't propose any new attack, but such reductions give important insights about the structure of GOST cipher, and structure the space of other reductions studied in this paper. In fact our basic reduction presented below, underpins most (but not all) of our reductions from 32 to 8 rounds and can be seen as a first step in these more complex reductions and the resulting attacks.

The key question to be asked for the cipher such as GOST is: what is the cost, in terms of success probability in our guess, and data complexity, of obtaining 1 P/C pair for 16 rounds? Similarly what will be the cost of obtaining more pairs?

F.1 Black-box Reductions from 32 to 16 Rounds

One such reduction is already present in Step 1. in Table 2 and underlies all the attacks which are summarized in this table. For completeness, we recall this reduction in its basic form:

Reduction 9. [From 2^{32} KP for 32 Rounds to 1KP for 16 Rounds]
Given an average expected number of 2^{32} known plaintexts for GOST, it is possible to obtain one P/C pair for 16 rounds of GOST and our guess will be correct with probability 2^{-32}.

Justification: The full justification is already given in Reduction 1 in Section 9.1, and also used in Reduction 2 in Section 10. We guess i. The reflection occurs with probability 2^{-32} in which case $\mathcal{E}^3(A)$ is symmetric where $A = X_i$. Then we obtain our pair for 16 rounds as follows. Let $C = Enc_k(A)$ then $C = Enc_k(A) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(A))) = \mathcal{D}(\mathcal{E}^3(A)) = \mathcal{E}^2(A)$.

Remark: One we get a pair for 16 rounds, one can be tempted to apply our Amplification method given by the Fact 18. However here, quite exceptionally, we are in the case in which it does not work, which is also because an internal reflection occurs. Therefore it is not trivial to obtain 2 pairs for 16 rounds.

F.2 More Black-box Reductions from 32 to 16 Rounds

By Applying Reduction 9 twice we immediately obtain that:

Reduction 10. [From 2^{33} KP for 32 Rounds to 2KP for 16 Rounds]
Given an average expected number of 2^{33} known plaintexts for GOST, it is possible to obtain two P/C pairs for 16 rounds of GOST and our guess will be correct with probability 2^{-64}.

F.3 Slight Improvement Based On Amplification

There is another simple method to obtain two pairs for 16 rounds, where our guess will be correct with probability 2^{-64}. We simply guess 1 pair, X, Y, and apply the Amplification property of Fact 18.

$$Y = \mathcal{E}^2(X)$$
$$\vdash$$
$$Enc_k(X) = \mathcal{E}^2(Dec_k(Y))$$

However this method as described here seems less interesting that Reduction 10 above. It seems to require 2^{64} KP to be able to either encrypt X or decrypt Y. Happily we are able to propose a non-trivial variant of this method which requires only 2^{32} KP, which will be a strict improvement compared to 2^{33} KP required by Reduction 10.

Reduction 11. [From 2^{32} KP for 32 Rounds to 2KP for 16 Rounds]
Given an average expected number of 2^{32} known plaintexts for GOST, it is possible to obtain two P/C pairs for 16 rounds of GOST and our guess will be correct with probability 2^{-64}.

Justification: Given a set of 2^{32} KP, $X_i \mapsto Y_i$ for 32 rounds, the probability that there exists i, j such that $X_j = \mathcal{E}^2(Y_i)$ is close to 1. We guess i, j and we obtain two pairs without any further access to encryption/decryption oracles by the Amplification method. Following Fact 18 we have the following (not totally obvious) result:

$$X_j = \mathcal{E}^2(Y_i)$$
$$\vdash$$
$$Y_j = \mathcal{E}^2(X_i).$$

F.4 Potential Applications

Under certain conditions, our Reduction 10 as well as Reduction 11 could allow attacks faster than brute force on the full 32-round GOST:

Fact 19 (Hypothetic Attack with Reduction to 16 rounds). If there exists an attack on 16 rounds of GOST which allows to recover the key given only 2 P/C pairs for 16 rounds (with 2^{128} false positives generated during this process checked later against additional P/C pairs) which is faster than 2^{192} full GOST encryptions, then it can be transformed into an attack on the full-round GOST faster than brute force.

Justification: This is obvious given Reduction 10 or Reduction 11 which would multiply the complexity of our key recovery attack by a factor of 2^{64}.

In the next section we revisit the question of reductions which result in P/C pairs for 16 rounds and we are going to develop much more interesting applications for these reductions.

G Algebraic Complexity Reduction and Chosen Plaintext or Chosen Ciphertext Attacks

Now we are going to do something quite unique, compared to other reductions described in this paper. All the other reductions are reductions to 2,3,4 or 5 known P/C pairs for 4 or 8 rounds. However the outcome of a Black-Box Algebraic Complexity Reduction can also be for example 4 chosen plaintexts (or chosen ciphertexts) for 8 rounds, this if we are able to do the reduction in such a way that the attacker can chose the plaintexts or the ciphertexts, which is in general much harder to achieve than a reduction which produces just some known pairs. Of course we are not going to obtain a choice of specific plaintexts or ciphertexts by the attacker with certainty, but, as in all our reductions, an attack in which some pairs are freely chosen by the attacker, but the result is only correct with some probability. With this probability, for example 2^{-32}, we should obtain all the relations and characteristics it is claimed to have to hold simultaneously. In other cases, for example with probability $1 - 2^{-32}$ we still have chosen plaintexts, but the result is incorrect, which, as usual, needs to be taken into consideration of false positives in the attack. And again, if the total number of these false positives is small enough, we don't need to worry about them, even if each of them needs to be checked against additional pair for 32 rounds.

A care also needs to be taken in such attacks that the attacker can have a sufficient supply of cases to choose from, because only with some small probability the choices of the attacker are actually used in the "effective" part of the attack execution, which is the one which finds the correct key.

G.1 Chosen Plaintext Reductions From 32 to 16 Rounds

Now we can look at the reductions from 32 to 16 rounds in the previous sections in the new light. Reductions in which the attacker is able to choose both plaintexts, will be much more interesting than other reductions.

We are going to revisit Reduction 9 and Reduction 10.

Reduction 12. [From 2^{32} KP for 32 Rounds to 1CP for 16 Rounds]
Given an average expected number of 2^{32} CP for GOST, it is possible to obtain one P/C pair for 16 rounds of GOST where the plaintext is freely chosen by the attacker, sampled from a probability distribution, or from another source such as an oracle, and our guess (and the corresponding ciphertext value after 16 rounds we obtained for the chosen plaintext) will be correct with probability 2^{-32}.

Justification: We guess i, for which the reflection occurs at round 24 of encryption of X_i. This happens with probability 2^{-32} in which case $\mathcal{E}^3(A)$ is symmetric where $A = X_i$ and we obtain our pair for 16 rounds as usual, it is $A, C = Enc_k(A)$.

Again by Applying Reduction 12 twice and exactly in the same way as before, twice we immediately obtain that:

Reduction 13. [From 2^{33} KP for 32 Rounds to 2CP for 16 Rounds]
Given an average expected number of 2^{33} known plaintexts for GOST, it is possible to obtain two P/C pairs for 16 rounds of GOST and our guess will be correct with probability 2^{-64}.

Remark. We could have also produced a reduction to 1 or 2 chosen ciphertexts in the same way, and some other. We leave it for further research.

H Conjugation Property of GOST

We discovered another peculiar property of GOST. It is not clear if this property leads to any attacks on GOST. We recall our decomposition of GOST encryption function:

$$Enc_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E} \tag{9}$$

Fact 20 (Conjugation Property). The cycle structure of Enc_k is exactly the same as of $\mathcal{S} \circ \mathcal{E}^2$. In particular Enc_k has i points of order j if and only if $\mathcal{S} \circ \mathcal{E}^2$ has i points of order j.

In particular, given 2^{64} KP, the cycle structure of $\mathcal{S} \circ \mathcal{E}^2$ can be computed by the attacker.

H.1 One Potential Application

This property implies that if we had a large proportion of the code-book of the first 16 rounds of GOST, we could immediately see if this code-book is authentic or not, which is related to Amplification Paradox of Section E.2 above: if we are able to derive from one single pair for 16 rounds, a large number of pairs for 16 rounds, then some of them could lead to a contradiction with the cycle structure of $\mathcal{S} \circ \mathcal{E}^2$ which is always known to the attacker.

H.2 An Actual Application

Another application will be to detect that the GOST key is weak. Then the statistics on the cycle structure and other properties of $\mathcal{S} \circ \mathcal{E}^2$ may provide circumstantial evidence, or disprove an assumption, that a given GOST device has keys with a particular structure (which may make GOST weaker and easier to cryptanalyse). For example for keys of Family B studied in Section N, it is possible to see that (cf. Fact 35) the function $\mathcal{S} \circ \mathcal{E}^2$ is an involution, which is very easy to detect with 2 CP. If this is not the case, we can be certain that a given key is not in Family B. Similarly, we can use this property to easily disprove that a given key belongs to other weak key families. In the following sections we will precisely study some such families of weak keys in GOST.

I Approximate Reflection in GOST

It is possible to show the following property:

Fact 21 (Approximate Internal Reflection Property). Consider the (bijective) function $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ for one fixed GOST key. Consider the difference between X and the value obtained when this function is applied:

$$X \oplus \mathcal{D}(\mathcal{S}(\mathcal{E}(X))).$$

and look at the 50 bits out of 64 which are at 0 in 0x8070070080700700. The probability that these 50 bits are at 0 is at most about 2^{-49} instead of 2^{-50} expected for a random permutation.

Justification: The basic justification is as follows. Let $Y = \mathcal{E}(X)$. We consider the difference between Y and $Y' = \mathcal{S}(Y)$. This value $Y \oplus Y' = Y \oplus \mathcal{S}(Y)$ is a symmetric value with both halves equal. The probability that such a value has the 50 bits at 0 at all the 50 positions which are at 0 in 0x8070070080700700 is high and equal to the probability that the left hand side has 25 bits at 0, which is 2^{-25}.

Let

$$Z = \mathcal{D}(\mathcal{S}(\mathcal{E}(X))).$$

We have

$$\mathcal{E}(Z) = \mathcal{S}(\mathcal{E}(X)).$$

and for this couple of applications of $\mathcal{E}$ we have here an output difference of type 0x8070070080700700 with probability at least 2^{-25}, and thus we also have an input difference of type 0x8070070080700700 with probability at least $2^{-25-25} = 2^{-50}$. But this can also occur by accident with probability 2^{-50}. Overall we expect it occurs with probability of about $2^{-50} + 2^{-50} = 2^{-49}$.

Remark: This is a very weak property, knowing that exact reflection occurs with probability 2^{-32}.

J Some Interesting Weak Key Attacks on GOST

Weak keys offer a considerable degree of extra freedom to the attacker. One basic attack of this type has been published [35]. This attack breaks GOST for weak keys which occur with probability 2^{-32}. For these keys the attack allows to break GOST with a time complexity of 2^{192} and given 2^{32} chosen plaintexts.

Let d denote the density of keys for which a given attack works, defined as the probability that the attack will work for a key chosen uniformly at random. Up till now we didn't deal with weak keys, and had very large values of $d \geq 0.63$. We should note however that the probability of $d = 2^{-32}$ is still quite large and should be considered as quite realistic. Given that the population of our planet is about 2^{33}, and one person can use during their life many cryptographic keys, an attack with $d = 2^{-32}$ should be considered as semi-realistic: it is plausible to assume that at some moment in the future 2^{32} different GOST keys will be used worldwide, making one of these keys vulnerable to the attack from [35]. This type of weak-keys which are frequent enough to occur in the real life are worth studying.

Given the fact that even without weak keys, we have been able to find 6 different attacks on GOST faster than brute force, the reader can imagine that there exists a plethora of interesting weak keys attacks on GOST. We are just going to exhibit some examples which we found interesting, either because they have a large d, or low complexity. We start by recalling the method of [35].

J.1 Weak Key Family 0

Fact 22 (Weak Keys Family 0, $d = 2^{-32}$, Reduction to 1 KP for 8R). We define the Weak Keys Family 0 by keys such that $\mathcal{E}$ has a fixed point A which is symmetric, i.e. $\overline{A} = A$. This occurs with density $d = 2^{-32}$.

For every key in Weak Keys Family 0, given 2^{32} chosen plaintexts for GOST, we can compute A and obtain 1 P/C pair for 8 rounds of GOST correct with very high probability of about 2^{-1}.

Justification: If A is a symmetric value such that $\mathcal{E}(A) = A$ then $Enc_k(A) = A$. However there are also, on average, about one values for which $Enc_k(A) = A$, as every permutation of 64 bits has about one fixed point which occurs by accident, not due to the internal structure. Thus we obtain 1 P/C pair for 8 rounds of GOST $\mathcal{E}(A) = A$, which is correct only with high probability of about 1/2.

J.2 Key Recovery With Family 0

Now in [35], this method of Fact 22 is used to recover keys with time complexity of 2^{192} and negligible memory. This is very hard to improve because the attack uses only 1 KP for 32 rounds, and there are 2^{192} keys for which this pair is correct, and all these keys must be checked against additional P/C pairs for the full 32-rounds. (for 128-bit keys, see Section M). In the next section we will introduce another family of weak keys, where we will be able at last to improve the time complexity of the attack.

J.3 Weak Key Family 1 (Different Than Earlier Family 1)

Important: we thank the anonymous referee for pointing out that our initial Family 1 attack did not work as claimed.

Now we are going to exhibit another family of weak keys, with the same density but with a possibility to obtain more P/C pairs and improve the time complexity of the attack. We will also require 2^{64} KP instead of 2^{32} CP.

Fact 23 (Weak Keys Family 1, $d = 2^{-32}$, Reduction to 3 KP for 8R). We define the Weak Keys Family 1 by keys such that if A is a fixed point of Enc_k for the full 32 rounds, A is not symmetric (cf. Family 0) and $F = \mathcal{E}^4(A)$ is symmetric. This occurs with density of about $d = 2^{-32}$.

For every key in Weak Keys Family 1, given 2^{64} KP for GOST, we can compute A, B and obtain 1 P/C pairs for 8 rounds of GOST correct with probability of about 2^{-2}. Furthermore we can compute A, B, C and get 3 P/C pairs for 8 rounds correct with probability of about 2^{-66}.

Justification: On average we have one fixed point for the whole cipher, and the probability that is $F = \mathcal{E}^4(A)$ is symmetric is $d = 2^{-32}$ taken over all possible GOST keys. For these keys we proceed as follows:

1. First we observe that since A is a fixed point and very few other fixed points exist for the permutation Enc_k, we can obtain A in time 2^{64} and our guess will be correct with very high probability of about 2^{-1}.
2. Then we observe that if B is defined as $B = \mathcal{E}(A)$ then we have:

$$B = \mathcal{E}(A) = \mathcal{E}(Enc_k(A)) = \mathcal{E}(\mathcal{D}(\mathcal{S}(\mathcal{E}^3(A)))) = \mathcal{S}(\mathcal{E}^3(A)) = \mathcal{S}(\mathcal{E}^2(B))$$

   therefore we have

$$\mathcal{E}^2(B) = \overline{B}.$$

3. Let C be defined as $C = \mathcal{E}^2(A)$. We have $\mathcal{E}(C) = \mathcal{E}^2(B) = \overline{B}$.
4. Additionally $\mathcal{E}(\overline{B}) = F$ which is assumed to be symmetric. Then we can use an internal reflection and we get that

$$Enc_k(B) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(B))) = \mathcal{D}(\mathcal{S}(\mathcal{E}(\overline{B}))) = \mathcal{D}(F) = \overline{B}.$$

   This means that we do not have to guess B because it satisfies $Enc_k(B) = \overline{B}$ and can be determined with probability about 2^{-1} with 2^{64} KP.
5. We get 1 pair for 8 rounds: $B = \mathcal{E}(A)$ correct with probability about 2^{-2}.
6. We guess C and get 3 pairs $B = \mathcal{E}(A)$, $C = \mathcal{E}(B)$, $\overline{B} = \mathcal{E}(C)$ correct with probability about 2^{-66}.
7. We do NOT guess the symmetric value F, (this would lead to 4KP for the price of about 2^{-98}).

Now we need to examine the consequences of this reduction with 3 P/C pairs.

Fact 24 (Key Recovery for Weak Keys Family 1, $d = 2^{-32}$).
One can recover the keys for the Weak Keys Family 1 with 2^{64} KP, running time of 2^{186} GOST encryptions and with negligible memory.

Justification: We use Fact 5 with the 3 P/C pairs obtained, and in total time equivalent to 2^{120} GOST encryptions we obtain 2^{64} candidates for the GOST key k. This needs to be multiplied by 2^{66} attempts to guess A, B, C.

The number of false positives is 2^{128} because we mount this attack with two pairs obtained from $Enc_k()$: one such that $Enc_k(A) = A$ and another such that $Enc_k(B) = \overline{B}$. The time to reject all the false positive keys with additional P/C pairs for the full 32-round GOST can be neglected in comparison to our 2^{186} GOST encryptions, which is the overall total time for this attack.

Discussion: 2^{192} per weak key, vs 2^{192} per randomly generated key. This attack has a complexity of less than 2^{192} for on weak key. However if some key are weak and some are strong, the attack will fail most of the time. Overall for arbitrary 256-bit keys generated at random we would obtain one key in total time of about 2^{186+32} GOST encryptions per key effectively found. In order to obtain an arguably better attack than [22] we would need to obtain an attack which achieves really less than 2^{192} per key, including the time to examine all the keys (great majority being immune to this attack). Surprisingly, this is possible and in what follows we are going to exhibits two such attacks (cf. later Table 3).

J.4 Weak Key Family 2

In this section we exhibit another family of weak keys, with the same density $d = 2^{-32}$ but with more extensive possibilities. This attack can be seen as an extension of our two attacks of Section 11, and Section 11.1, both based on Reduction 3, where by requiring that B is also symmetric, (which happens only for weak keys but the probability of these keys is quite large of 2^{-32}) we will be able to simultaneously improve the probability of our guess being true, from 2^{-128} to 2^{-64} to obtain 4 P/C pairs for 8 rounds, and reduce the data complexity back to 2^{32} chosen ciphertexts, and we will also be able to obtain, for the first time ever, (up to) 5 pairs for 8 rounds.

Fact 25 (Weak Keys Family 2, $d = 2^{-32}$, Getting 3,4 and 5 KP for 8R). We define the Weak Keys Family 2 by keys such there exists A such that all the three values $\mathcal{E}(A)$, $\mathcal{E}^2(A)$ and $\mathcal{E}^3(A)$ are symmetric. This occurs with density $d = 2^{-32}$. For every key in Family 2, we have the following reductions:
-with 2^{32} CC we obtain 3 P/C pairs for 8 rounds of GOST correct with $P = 2^{-64}$,
-with 2^{64} KP we obtain 4 P/C pairs for 8 rounds of GOST correct with $P = 2^{-64}$,
-with 2^{32} CC we obtain 4 P/C pairs for 8 rounds of GOST correct with $P = 2^{-96}$,
-with 2^{64} KP we obtain 5 P/C pairs for 8 rounds of GOST correct with $P = 2^{-96}$.

Justification: See Fig. 4. This can be seen as an extension of Fig. 1 except that since B is also symmetric and accordingly, we could add another column to the right where Y is encrypted to obtain in turn $Y, Z, A, B, B, A = Enc_k(Y)$.

We have three encryptions with internal reflection $C = Enc_k(A)$, also $B = Enc_k(Z)$, and $A = Enc_k(Y)$ where due to the internal reflection we have $C = \mathcal{E}^2(A)$, $B = \mathcal{E}^2(Z)$, and $A = \mathcal{E}^2(Y)$.

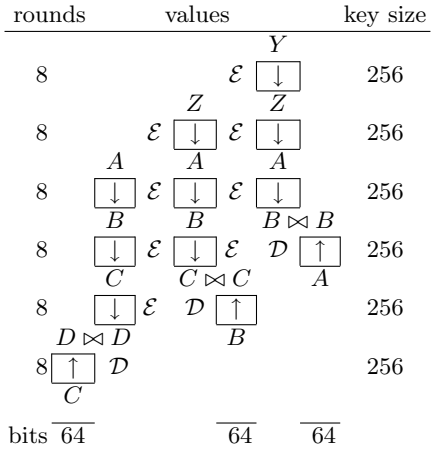

Fig. 4. A weak-key black-box reduction which gives up to 5 pairs for 8 rounds

There are two interesting attack scenarios. In all cases we start by guessing C and B which are symmetric and therefore, to decrypt these and obtain respectively A and Z we need only 2^{32} CC. However if we also want to decrypt A to obtain Y, we need 2^{64} KP.

We proceed as follows:

1. We guess B, C and our guess is correct with probability 2^{-64}.
2. We determine A, Z by decrypting B and C which are both symmetric.
3. We get 3 pairs $\mathcal{E}(Z) = A$, $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$ and our guess is correct with probability 2^{-64}.
4. Furthermore, if we also decrypt A we get also one additional pair $\mathcal{E}(Y) = Z$, at the price of 2^{64} KP because A is not symmetric.
5. Going one step backwards, we don't decrypt A but also guess D which is symmetric, We get 4 pairs $\mathcal{E}(Z) = A$, $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$, $\mathcal{E}(C) = D$ and our guess is correct with probability 2^{-96}.
6. Now if we combine guessing D and decrypting A, we get 5 pairs given 2^{64} KP and our guess is correct with probability 2^{-96}.

Out of four possibilities given in Fact 25, we will use two in order to obtain two new weak key attacks:

Fact 26 (Key Recovery for Family 2, 2^{32} CC, $d = 2^{-32}$).
One can recover the keys for the Weak Keys Family 2 with 2^{32} CC, running time of 2^{184} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the first reduction of Fact 25 and of Fact 5 which allows to enumerate a set of solutions and the time is 2^{64+120} GOST encryptions. The total number of full 256-bits keys which are false positives which need to be checked against additional P/C pairs for the full 32 rounds of the cipher is comparatively smaller, about 2^{128}, which is unlikely to influence the overall complexity of the attack which will be 2^{184} GOST encryptions.

Similarly, with just one more pair $\mathcal{E}(Y) = Z$, which is obtained with the same key density and the same success probability, but at the price of 2^{64} KP, we obtain:

Fact 27 (Faster Key Recovery for Family 2, 2^{64} KP, $d = 2^{-32}$).
One can recover the keys for the Weak Keys Family 2 with 2^{64} KP, running time of 2^{184} GOST encryptions and with negligible memory.

Justification: Here we replace 3 KP by 4 KP and Fact 5 with 2^{120} by Fact 6 with 2^{94}. We need a total time of $2^{64+94} = 2^{158}$ GOST encryptions.

J.5 Weak Key Family 3

In this section we explore if better attacks exist, and in particular attacks with complexity less than 2^{128}, at the price of further decreasing the density of weak keys to 2^{-64}.

Fact 28 (Weak Keys Family 3, $d = 2^{-64}$, Getting 4 KP for 8R). We define the Weak Keys Family 3 by keys such there exists A such that $\mathcal{E}(A) = \overline{A}$, $\mathcal{E}^2(\overline{A}) = A$. This occurs with density $d = 2^{-64}$. For every key in Family 3, we have the following: with 2^{64} KP we obtain 4 P/C pairs for 8 rounds of GOST, correct with probability of roughly about $P = 2^{-1}$.
Justification: We proceed as follows:

1. First we observe that A is a fixed point for $Enc_k(\cdot)$. Indeed

$$Enc_k(A) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(A))) = \mathcal{D}(\mathcal{S}(\mathcal{E}^2(\overline{A}))) = \mathcal{D}(\mathcal{S}(A)) = \mathcal{D}(\overline{A}) = A.$$

   Therefore given 2^{64} KP we can identify A. Due to other possible fixed points, our guess will be correct with probability roughly about $P = 2^{-1}$.
2. Moreover if we define $B = \mathcal{E}(\overline{A})$ we have $A = \mathcal{E}(B)$ and

$$Enc_k(\overline{A}) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(\overline{A}))) = \mathcal{D}(\mathcal{S}(\mathcal{E}(A))) = \mathcal{D}(\mathcal{S}(\overline{A})) = \mathcal{D}(A) = B.$$

   Therefore we can determine B from A.

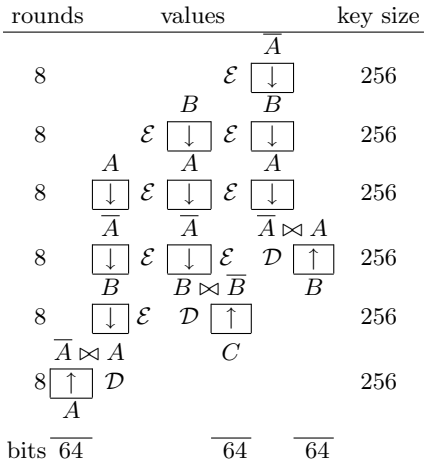

Fig. 5. Weak Key Family 3 which gives 4 pairs for 8 rounds

3. Moreover, if we encrypt B we obtain another interesting value C defined as:

$$Enc_k(B) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(B))) = \mathcal{D}(\mathcal{S}(\mathcal{E}^2(A))) = \mathcal{D}(\mathcal{S}(\mathcal{E}(\overline{A}))) = \mathcal{D}(\mathcal{S}(B)) = \mathcal{D}(\overline{B}) = C.$$

   with the property that $\mathcal{E}(C) = \overline{B}$.
4. Overall our triple A, B, C will be correct with probability about $P = 2^{-1}$. We get 4 P/C pairs for 8 round which are $\mathcal{E}(A) = \overline{A}$, $\mathcal{E}(\overline{A}) = B$, $\mathcal{E}(B) = A$ $\mathcal{E}(C) = \overline{B}$ and these are correct with probability 2^{-1}.

Fact 29 (Key Recovery for Weak Keys Family 3, $d = 2^{-64}$).
One can recover the keys for the Weak Keys Family 3 with 2^{64} KP, running time of 2^{95} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 28 and Fact 6 for 4 KP. This reduction manages to exploit the information obtained from as many as three different encryptions for $Enc_k()$ for $A, \overline{A}$ and B. It is therefore possible to see that in this attack, the total number of false positives which need to be checked against additional P/C pairs for the full 32 rounds is only 2^{64}.

K Summary of Our Weak Key Attacks

In Table 3 we can compare our attacks with weak keys compared to selected other attacks with regular keys

It is important to note that with regular keys, we had no attack below 2^{192}. One of the barriers to achieve really fast attacks are false positives. With attacks such all our black box reduction attacks, only if we manage to simultaneously exploit similarities inside two encryptions for 32 rounds and convert them to a number of simpler P/C pairs for 8 rounds we will have less than 2^{192} false positives. And this condition is necessary, but not sufficient. Amazingly enough, we obtained not only several attacks with time being below 2^{192} per weak key, but two of them are below 2^{192} per key for arbitrary keys generated at random.

Finally and furthermore, an interesting question now is whether attacks below 2^{128} can exist for some 256-bit GOST keys. We see that this can be achieved with Family 3 and we are below 2^{100} per weak key with $d = 2^{-64}$ and overall below 2^{160} per key for random keys.

Reduction cf.	Red.1 §9.1		Family 0	Red. 3 §11	Family 2	Family 2	Family 3	Fam. 0'§M
Key size	256							128
Keys density d	0.63		2^{-32}	0.63	2^{-32}		2^{-64}	$2^{(-128)-32}$
From (data 32 R)	2^{32} KP		2^{32} CP	2^{64} KP	2^{32} CC	2^{64} KP		2^{32} CP
Obtained (for 8R)	**2** KP		**1** KP	**3** KP	**3** KP	**4** KP	**4** KP	1(4R)
Valid w. prob.	2^{-96}		2^{-1}	2^{-96}	2^{-64}	2^{-64}	2^{-1}	2^{-1}
Storage bytes	2^{132}	-	-	2^{67}	-	2^{67}		-
♯ False positives	2^{192}			2^{128}		2^{64}		2^{64}
Time to break 8 R	2^{128}	2^{131}	2^{192}	2^{110}	2^{120}	$\mathbf{2^{94}}$	$\mathbf{2^{94}}$	$\mathbf{2^{64}}$(4R)
Attack time 32 R	2^{224}	2^{227}	2^{192}	2^{206}	2^{184}	$\mathbf{2^{158}}$	$\mathbf{2^{95}}$	$\mathbf{2^{65}}$
Cost of 1 key, if	2^{225}	2^{228}	2^{223}	2^{207}	2^{226}	$\mathbf{2^{190}}$	$\mathbf{2^{159}}$	2^{66}
key diversity $\geq$	$2^{0.7}$		2^{32}	$2^{0.7}$	2^{32}		2^{64}	2^{32}

Table 3. Selected weak key attacks compared to the best regular attacks

L How to Transform A Weak-Key Attack Into A Regular Attack on Random 256-bit Keys With Total Time Of 2^{190} and 2^{159}

If we are dealing with the problem of key recovery of a **single** fixed 256-bit GOST key, then our best attack in Table 2 on page 11. requires 2^{64} known plaintexts has the time complexity of 2^{195}, which can be further reduced to 2^{192} as shown by Shamir *et al* in [22]

However, in this paper we have also two arguably better attacks. We look at the last line in Table 3 above, for two of the attacks we have that the ratio Time / d is less than 2^{192}. This actually **does mean** that GOST key can be recovered in overall total time of 2^{159} which is substantially less than our best attack 2^{195} of Table 2 and also better than the most recent and best ever found attacks on GOST with 2^{192} of [22] and 2^{178} of [19]. Here is how.

Fact 30 (Key Recovery for A Diverse Population of Keys, $d = 2^{-64}$). If we have a diverse population of at least 2^{64} different keys, with access to 2^{64} KP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{159} GOST encryptions.

Justification: We apply the Fact 29 to 2^{64} random devices. For each device with probability about half, if the key is weak, and Following Fact 28, we obtain 4 P/C pairs for 8 rounds of GOST. If the key is not weak we still obtain 4 pairs but they are wrong. Then in each case we run the attack which takes 2^{95} GOST encryptions. In one case on average, the attack will work and output a valid key which can be checked with additional pairs for that device.

Remark: Here the cost of one key is equal to the attack time divided by density, and this is the total complexity of a realistic attack which breaks one key out of many. Though it is not a single attack it is more realistic than the best single key attacks on GOST [22, 19], because it will actually recover one full 256-bit key with a total cost of 2^{159}.

Similarly we have, in an even more realistic scenario, frequent enough to maybe occur in the real life:

Fact 31 (Key Recovery for A Diverse Population of Keys, $d = 2^{-32}$). If we have a diverse population of at least 2^{32} different keys, with access to 2^{64} KP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{190} GOST encryptions.

Justification: We apply Fact 27 to each of the 2^{32} devices, and we recover one key out of 2^{32} in total time of 2^{32+158}.

M Attacks on GOST With Repeated 128-bit Keys

In this section we study the security of GOST with 128-bit keys. Unhappily, though we found many different attacks which allow to break 256-bit keys, the sheer cost ot the final key recovery step and the existence of false positives makes that our previous attacks are very far from being able to break 128-bit keys. In this section we show that there are weak keys which allow very efficient attacks and these keys exist with probability high enough to be a practical concern in a population of diverse keys, leading to a compromise of certain keys in practice.

We assume that the GOST key is such that the same 128-bit key is repeated twice. We ignore if such a variant of GOST is used in practice but this method is one of the interesting special variants of GOST explicitly considered on page 594 of [3]. Our attack uses a modified variant of the attack from [35] which we named Family 0, see Fact 22. In the new version called Family 0' we will consider fixed points for 4 rounds instead of 8 rounds. Let $\mathcal{F}$ be the first 4 rounds with 128-bit key. If the key is repeated we have $\mathcal{E} = \mathcal{F} \circ \mathcal{F}$.

Then we define the Family 0' of weak keys as follows:

Definition M.0.1 (Weak Keys Family 0'). We define the Weak Keys Family 0' as 128-bit keys with repetition AND such that $\mathcal{F}$ has a fixed point A which is symmetric, i.e. $\overline{A} = A$.
This occurs with probability of $d = 2^{-32}$ over all GOST keys.

Then we have the following:

Reduction 14 (Family 0', $d = 2^{-32}$, Reduction to 1 KP for 4R). For every key in Weak Keys Family 0', given 2^{32} chosen plaintexts for GOST, we can compute A and obtain 1 P/C pair for 4 rounds of GOST correct with very high probability of about 2^{-1}.

Justification: It is the same as for Fact 22: If A is symmetric and $\mathcal{F}(A) = A$ then $Enc_k(A) = A$. However there are also, on average some other values for which $Enc_k(A) = A$, as every permutation of 64 bits has about one fixed point which occurs by accident, not due to the internal structure. Thus we obtain 1 P/C pair for 4 rounds of GOST $\mathcal{F}(A) = A$, which is correct only with high probability of about 1/2.

Fact 32 (Key Recovery for Family 0' With 128-bit Keys). We assume that the GOST key is 128-bit with repetition and belongs to family 0'. Then given 2^{32} CP one can recover the 128-bit key from Family 0' in average time of 2^{65} GOST encryptions and with negligible memory.

Justification: We obtain A with 2^{32} CP, as a symmetric fixed point of $\mathcal{F}(\cdot)$. This is done once at the beginning. The time to do this is less than 2^{32} GOST encryptions and can be neglected. We have

$$\mathcal{F}(A) = A$$

where $\mathcal{F}$ is the first 4 rounds with 128-bit key.
Our A is correct with high probability of about 2^{-1}.

Now we can obtain a uniform enumeration of exactly 2^{64} keys on 128-bits which satisfy this equation as follows: we fix the 64-bit key for the first 64 rounds, and because the GOST S-boxes are bijective, this gives us the knowledge of inputs and outputs of both rounds 3 and 4, and allows us to uniquely determine the second 64-bit of the key in time of encrypting with GOST for 2 rounds, which is 2/32 GOST encryptions. Overall, we get a uniform enumeration of exactly 2^{64} keys on 128-bits in time of 2^{60} GOST encryptions. Each of these keys needs to be checked with another P/C pair for the full 32-round GOST. The total time is $2^1(2^{60} + 2^{64}) \approx 2^{65}$ GOST encryptions. and half this time on average.

M.1 Attacks on A Diverse Population of 128-bit Keys

Now we need to translate this to a more realistic scenario where there is a population of different GOST keys, but we don't know which ones are weak.

Fact 33 (Attack with Diverse 128-bit Keys, 2^{32} CP Per Key, $d = 2^{-32}$). We assume that there is a population of 2^{32} devices with 128-bit GOST keys repeated twice to form a 256-bit key (and in principle not being weak keys in most cases). Then one of these keys on average is a weak key from Family 0'. Given 2^{32} CP per device, the device having the weak key can be identified and the key recovered in total time of 2^{66} GOST encryptions on average and with negligible memory.

Justification: Let j be a key number. For each of 2^{32} keys j, given the possibility to obtain 2^{32} CP per key, for all possible symmetric plaintexts, we check if there are any symmetric fixed points A for $Enc_{k_j}(\cdot)$. This first step takes about 2^{64} steps, but in practice this is really substantially less than 2^{64} GOST encryptions, and can be neglected.

Only for the weak keys, and in about on average one another case, any of the fixed points is symmetric. Most devices are rejected immediately except a few. We obtain a list of about 2^1 pairs j, A. In each of these 2^1 cases we apply Fact 32. Thus total time is about 2^{66} GOST encryptions and the memory remains negligible.

Is this attack practical? Given that the population of our planet is about 2^{33}, and one person can use during their life many cryptographic keys, this attack should be considered as **semi-realistic**. In a hypothetic future, for example if GOST becomes an ISO standard, given the fact that it has larger keys than triple DES, and is cheaper to implement than triple DES and any other comparable cipher [37], it is possible that GOST becomes quite widely used, also in a 128-bit version, which would be judged secure enough for practical purposes. Then assume that these keys are embedded in some secure hardware (common practice in the industry) which can be freely accessed by the attacker and he can dispose of 2^{32} CP per key. Then our attack will allow to recover some of these 128-bit keys in practice.

M.2 Attacks With 1 CP Per Key

Now we are going to develop an even more realistic scenario where there is a population of different GOST keys, and we are given only 1 CP per key. We can break GOST also in this scenario.

Fact 34 (Attack with Diverse 128-bit Keys, 1 CP Per Key, $d = 2^{-64}$). We assume that there is a population of 2^{64} devices with, possibly different, but can also be repeated, 128-bit GOST keys in repeated twice to form a 256-bit key (and in principle not being weak keys in most cases).

And that the attacker is given just the encryption of some symmetric plaintext, such as $A = 0$ (for full 32 round GOST) for each of these keys, plus any additional data for confirmation of the right keys, such as a ciphertext-only attack, in the form of some longer message encrypted with the same key in a given cipher mode such as CBC.

Then for 2^{32} cases, the key will be a weak key from Family 0', and if one case on average A, being symmetric, will be a fixed point of $\mathcal{F}$, and also of $Enc_k()$ and will be also equal to A.

Then the case out of 2^{64} with the weak key, and with A being a fixed point can be identified and the key recovered in total time of 2^{66} GOST encryptions on average and with negligible memory.

Justification: Let j be a key number. For each of 2^{64} keys j, we filter out the keys for which A is a fixed point. We expect to obtain one right case, in which the key will be a weak key from Family 0', and A being a fixed point of $\mathcal{F}$, and one another case where the fixed point A occurs by accident. This first step takes about 2^{64} steps, but in practice this is really substantially less than 2^{64} GOST encryptions, and can be neglected.

Most devices are rejected immediately except a few. We obtain a list of about 2^1 pairs j, A. In each of these 2^1 cases we apply Fact 32. Each of the 2^{64} keys found in this process needs to be checked with another few P/C pairs for the full 32-round GOST or with the data provided for the ciphertext-only attack The total time is again about 2^{66} GOST encryptions and the memory remains negligible.

Is this attack practical? Given that the population of our planet is about 2^{33}, and one person can use a standardized cipher such as GOST 10 times per day over one year to encrypt a message of 5 Megabytes containing only zeros, then one of the keys used over that period can be identified and recovered in total time of about 2^{66} GOST encryptions.

Remark It is easy to see that we also can have an attack with 2^X CP per key and $d = 2^{-64+X}$ and total time of still 2^{66} GOST encryptions for any $X = 0 \ldots 32$.

N One Particularly "Bad" Family B of 128-bit Keys

There is another natural method to use GOST with 128-bit keys. We assume that the second part of the key is not the repetition but an inverted repetition of the first part. By definition we call GOST keys of this form $k = (k_0, k_1, k_2, k_3, k_3, k_2, k_1, k_0)$ the Family B of keys (B stands for "Bad"). We don't know if this method is ever used in practice to encrypt data, but this method is is also one of those weak variants explicitly discussed on page 603 of [3]. This makes the key schedule perfectly periodic in spite of the inversion of keys in the last 8 rounds of GOST which is a protection against known slide and fixed point attacks. Thus one should not be surprised that this will make a "pathologically" bad block cipher with many interesting attacks. We would like to stress the fact that this key schedule is fully compliant with the GOST encryption standard, yet very weak. We have:

Fact 35 (GOST with Family B Keys). We assume that the GOST key is in Family B, in other words, let $k = (k_0, k_1, k_2, k_3, k_3, k_2, k_1, k_0)$. Again let $\mathcal{F}$ be the first 4 rounds with 128-bit key. Then we have the following immediate and easy to prove consequences of the structure of the cipher:

1. The sequence of round keys becomes perfectly periodic and symmetric:

rounds	1 8	9 16	17 24	25 32
keys	$k_0k_1k_2k_3k_3k_2k_1k_0$	$k_0k_1k_2k_3k_3k_2k_1k_0$	$k_0k_1k_2k_3k_3k_2k_1k_0$	$k_0k_1k_2k_3k_3k_2k_1k_0$

Table 4. The effect of key scheduling on Family B keys

2. The second 4 encryption rounds can be written as follows:
$$\mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S}$$
3. The first 8 encryption rounds $\mathcal{E}$ can be written as follows:
$$\mathcal{E} = \mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \tag{10}$$
$$\mathcal{E}^{-1} = \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{S} \tag{11}$$
4. The function $\mathcal{S} \circ \mathcal{E}$ is an involution and it is equal to its own inverse.
$$\mathcal{S} \circ \mathcal{E} = \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$$
$$\mathcal{E}^{-1} \circ \mathcal{S} = \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$$
5. It follows that for every X, Y:
$$\begin{gathered} Y = \mathcal{E}(X) \\ \Updownarrow \\ \overline{X} = \mathcal{E}(\overline{Y}). \end{gathered}$$
6. The function $\mathcal{S} \circ \mathcal{E}$ for the first 8 encryption rounds without the final twist, is a conjugated version $\mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$ of a function which has exactly 2^{32} fixed points. It follows that it has exactly 2^{32} fixed points which are exactly those and only those for which the state is symmetric after the first 4 rounds.

7. X is a fixed point of $\mathcal{E}$ if and only if $\overline{X}$ is a fixed point for the same $\mathcal{E}$.
8. For every $k \geq 1$ we have
$$\mathcal{S} \circ \mathcal{E}^k = \mathcal{G} \circ (\mathcal{S} \circ \mathcal{G})^{k-1} = (\mathcal{S} \circ \mathcal{G})^{k-1} \circ \mathcal{G}$$
$$\mathcal{E}^{-k} \circ \mathcal{S} = \mathcal{G} \circ (\mathcal{S} \circ \mathcal{G})^{k-1} = (\mathcal{S} \circ \mathcal{G})^{k-1} \circ \mathcal{G}$$
where we define $\mathcal{G} \stackrel{def}{=} \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$ which is an involution.
For every $k \geq 0$ the function $\mathcal{S} \circ \mathcal{E}^k$ is an involution and it can be written as
$$\mathcal{G} \circ \mathcal{S} \circ \mathcal{G} \circ \mathcal{S} \circ \cdots \circ \mathcal{G},$$
where $\mathcal{G}$ appears $k-1$ times and swap $\mathcal{S}$ appears k times.
9. For every $k \geq 0$ this function $\mathcal{S} \circ \mathcal{E}^k$ can be written in the form $\mathcal{H}^{-1} \circ \mathcal{S} \circ \mathcal{H}$ as follows:
$$\begin{cases} \mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-l} \circ \mathcal{S} \circ \mathcal{E}^l & \text{when } \ k = 2l \\ \mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-l} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{E}^l & \text{when } \ k = 2l+1 \end{cases}$$
Consequently for every k it has exactly 2^{32} fixed points which are exactly those for which the state is symmetric after the first $4k$ rounds of GOST.
10. The whole encryption process is perfectly periodic provided that we "undo" the final "irregular swap" and we have:
$$Enc_k = \mathcal{S} \circ \mathcal{E}^4 \tag{12}$$
11. In particular, the encryption function Enc_k is an involution.
12. If the attacker has access to the encryption oracle, he can use it to decrypt any message.
$$Y = Enc_k(X)$$
$$\Updownarrow$$
$$X = Enc_k(Y).$$
13. Consequently the encryption function Enc_k can be distinguished from a random permutation in constant time.
14. X is a fixed point of $\mathcal{S} \circ Enc_k$ if and only if $\overline{X}$ is also a fixed point for $\mathcal{S} \circ Enc_k$.
15. The whole encryption function Enc_k has exactly 2^{32} fixed points which are exactly those for which the state is symmetric after the first 16 rounds.

The next question is what is the best key recovery attack on this version of GOST. As we will see below, the most obvious (classical) slide and fixed point attacks, provide an immediate reduction in the number of rounds. However key recovery for 8 rounds is still far from being easy, even with the symmetry in the key schedule and the particular "involution with a twist" structure $\mathcal{S} \circ \mathcal{E} = \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$ implied by the Family B keys, and key recovery remains difficult. Especially in cases where the number of P/C pairs which can be obtained remains very small. Better attacks will be obtained, because we will be able to obtain pairs for 4 rounds, and when we will study cyclic properties of $\mathcal{E}$, and important involution and reflection properties of $\mathcal{E}$ and discover many 128-bit keys are 'weak' w.r.t. some previously studied weak key classes. All these properties provide multiple very useful degrees of freedom for the attacker which we will exploit.

N.1 Basic Fixed Point Attacks on Family B Keys

First we present one simple attack on $\mathcal{E}$ which requires 2^{64} KP. Later we will discover that $\mathcal{E}$ has some "very special" fixed points which allows attacks requiring much less data.

Reduction 15 (Fixed Point Reduction for Family B). Given 2^{64} known plaintexts for GOST with keys being in Family B, it is possible to obtain two P/C pairs for 4 rounds of GOST correct with probability of about 2^{-66} on average.
Justification: Let A be a fixed point for 8 rounds. By definition we have:

$$\mathcal{E} = \mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}.$$

This function is expected to have only a few fixed points, and we recall that X is a fixed point of $\mathcal{E}$ if and only if $\overline{X}$ is also a fixed point for $\mathcal{E}$, cf. Fact 35. In contrast $\mathcal{S} \circ \mathcal{E}$ has exactly 2^{32} fixed points. We expect that for 63 % of keys in Family B there exists a fixed point A with $A = \mathcal{E}(A)$ and very few other fixed points. Then we can observe also that it is a fixed point for $\mathcal{S} \circ Enc_k$, which is also expected to have only a very few fixed points, (as opposed to Enc_k which has exactly 2^{32} fixed points).

Thus a A can be easily guessed by the attacker given 2^{64} KP. Unhappily we also have fixed points of $\mathcal{S} \circ Enc_k = \mathcal{E}^4$ which are not fixed points of $\mathcal{E}$, but occur naturally. We consider that our fixed point for $\mathcal{S} \circ Enc_k$ will correctly be also a fixed point for $\mathcal{E}$ with probability of roughly about $2^{-1.5}$.

Then there exists $B = \mathcal{F}(A)$ such that we get two pairs for 4 rounds: $B = \mathcal{F}(A)$ and $\overline{B} = \mathcal{F}(\overline{A})$. These two pairs are distinct if neither B not A are symmetric, which happens with high probability. Additionally, it is easy to see that the overall event that there exist A, B where none of the two values is symmetric, AND $B = \mathcal{F}(A)$ AND $\overline{B} = \mathcal{F}(\overline{A})$ is likely to occur with probability at least about 63 % over Family B keys (for other keys this attack fails). This can be justified as follows There are still $(2^{64} - 2^{32})^2 \approx 2^{128}$ couples A, B where neither A nor B are symmetric, and the equations $B = \mathcal{F}(A)$ AND $\overline{B} = \mathcal{F}(\overline{A})$ will be satisfied with probability about 2^{-128} in each case. And $1 - (1 - 1/N)^N \approx 63\%$, where $N = 2^{128}$.

Finally we need also to guess $B = \mathcal{F}(A)$ and our guess will be correct with probability 2^{-64}. Overall we get two pairs for 4 rounds: $B = \mathcal{F}(A)$ and $\overline{B} = \mathcal{F}(\overline{A})$ which are correct and distinct with probability of about 2^{-66}.

This Fact 15, will be used to recover keys for Family B.

Fact 36 (Fixed Point Attack for Family B). Given 2^{64} known plaintexts for GOST with keys being in Family B, the key can be computed in time equivalent to 2^{90} GOST encryptions. Memory is required only to store the 2^{64} KP.
Justification: We use our Reduction 15 above and apply Fact 3: in each case the 128-bit key can be found in time of 2^{24} GOST computations and with negligible memory. Overall the key can be computed in time equivalent to 2^{90} GOST encryptions which is obtained as 2^{24+66}.

In what follows we are going to show an attack which is slightly slower but requires much less data. This type of improved attacks are possible, because $\mathcal{E}$ has another particularly interesting property.

N.2 On The Existence of Very Special Fixed Points for Family B

We have the following non-trivial fact:

Fact 37 (Special Symmetric Fixed Points for Family B). Given a GOST key in Family B chosen at random with probability of about $2^{-0.7}$ the first 8 rounds $\mathcal{F}$ have a symmetric fixed point (instead of, it would happen with probability about 2^{-32} for a random permutation).

Justification: We have

$$\mathcal{E} = \mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}.$$

We consider all the 2^{64} possible pairs A, B such that both A and B are symmetric. The probability that for a fixed pair A, B we have $\mathcal{F}(A) = B$ is 2^{-64}. The probability that there exists a pair A, B such that $\mathcal{F}(A) = B$ is $1-(1-1/N)^N \approx 63\%$, where $N = 2^{64}$. Then, we exploit the fact A and B are both symmetric and obtain:

$$\mathcal{E}(A) = \mathcal{S}(\mathcal{F}^{-1}(\mathcal{S}(\mathcal{F}(A)))) = \mathcal{S}(\mathcal{F}^{-1}(B)) = \mathcal{S}(A) = A.$$

Remark: Symmetric fixed points occur with high probability for any function which has about 2^{32} or more fixed points, for example if it is of the form $\mathcal{G} \circ \mathcal{S} \circ \mathcal{G}^{-1}$ or $\mathcal{S} \circ \mathcal{G} \circ \mathcal{S} \circ \mathcal{G}^{-1}$ or $\mathcal{G} \circ \mathcal{S} \circ \mathcal{G}^{-1} \circ \mathcal{S}$ etc. Consequently they also occur for the function $\mathcal{E} \circ \mathcal{S} \circ \mathcal{D}$ for normal 256-bit keys, i.e. the last 16 rounds of GOST. These points are precisely those which allowed to obtain and exploit a double reflection in our best attack on 256-bit GOST in Section 11.

N.3 Fixed Point and Multiple Reflection Attacks

The main reason why keys in Family B are particularly weak is that, following Fact 37, $\mathcal{E}$ has symmetric fixed points, which leads to fixed points for bigger components such as $\mathcal{E}^4$, thus becoming detectable for the attacker. It is also a multiple reflection attack: we are to create a double reflection in $\mathcal{E}$ which leads to fixed points and further reflections inside $\mathcal{E}^4$.

Fact 38 (Family B vs. Family 0). A GOST key in Family B chosen at random belongs to Weak Keys Family 0 with probability of about $2^{-0.7}$, instead of about 2^{-32} for a normal 256-bit GOST key chosen at random.

Justification: We recall that, by definition, the Weak Keys Family 0 are keys such that $\mathcal{E}$ has a fixed point A which is symmetric, i.e. $\mathcal{E}(A) = A$ and $\overline{A} = A$. and we have already established, cf. Fact 37, that such symmetric fixed points exist for $\mathcal{E}$ with very high probability $\approx 63\%$ over all keys in Family B.

Fact 38 is a very interesting observation. In full 256-bit GOST, and in the first "direct" method suggested by Biryukov and Wagner of using GOST with 128-bit keys, one could identify and break some weak keys which occurred with probability 2^{-32}. Here weak keys of Family 0, (also known from [35]) occur with a very high probability, while the overall secret key is also shorter, of 128-bit only. This will lead to a very good attack on GOST Family B of "inversed" keys, which will work for 63 % of all such keys. For the remaining 37 % of Family B keys this attack fails (but other attacks on Family B should still work).

How do we proceed to recover GOST keys? Here we could use the Fact 22: for all Family 0 keys, given 2^{32} CP, one can compute one P/C pair for 8 rounds of GOST nearly for free, i.e. one which will be correct with very high probability of about 2^{-1}. Moreover, it is also possible to see that several such pairs could be obtained with a non-negligible probability, this is because several pairs A, B such that $\mathcal{F}(A) = B$ and A, B are symmetrical can exist (for a lower proportion of Family B keys though). This leads to an attack just very slightly faster than 2^{128} GOST encryptions by direct application of Fact 5. We don't study these attacks because they are not very fast and slower than our previous fixed point attack above (cf. Fact 36).

Instead, we are going to directly look at the question of getting P/C pairs for 4 rounds of GOST, which requires one to guess the (symmetric) value B. The following result follows immediately:

Reduction 16 (Family B Reduction to 1 KP for 4R). Given a GOST key in Family B chosen at random with probability of about $2^{-0.7}$ over the key, and given 2^{32} CP, one can obtain a P/C pair A, B for 4 rounds, where both A and B are symmetric, and our guess will be correct with probability of 2^{-32}

Justification: For every key in Family B, with probability of $0.63 \approx 2^{-0.7}$ the function $\mathcal{E}$ has at least one symmetric fixed point A, and it can be found given on average only 2^{31} CP and in the worst case twice that number. The value A can be found by the attacker because it is also a fixed point for $\mathcal{S} \circ Enc_k$, and the probability that $\mathcal{S} \circ Enc_k$ has other fixed points which are symmetric, is negligible.

Once the right A is identified with almost-certainty, we need also to guess $B = \mathcal{F}(A)$ and our guess will be correct with probability 2^{-32}. Overall we get one pair for 4 rounds: $B = \mathcal{F}(A)$ where both A, B are symmetric, and our guess is correct with probability 2^{-32}.

Furthermore, with a non-negligible probability such an event can happen twice:

Reduction 17 (Family B Reduction to 2 KP 4R). For a random key in Family B, and given 2^{32} CP, one can compute two distinct random couples A, B and A', B' of four symmetric texts which satisfy $\mathcal{F}(A) = B$ and $\mathcal{F}(A') = B'$ for 4 rounds of GOST with overall probability of at least 2^{-66} over the choice of the key and the choice of B and B'.

Justification: Only for some keys this can happen. We need to compute the probability that at least two distinct random couples A, B of symmetric texts satisfy $\mathcal{F}(A) = B$ for 4 rounds of GOST. This is 1, minus the probability that none of the $N = 2^{64}$ possible couples A, B satisfies $\mathcal{F}(A) = B$, minus the probability that exactly one out of N couples satisfies $\mathcal{F}(A) = B$. This is equal to:

$$1 - (1 - 1/N)^N - \binom{N}{1} (1 - 1/N)^{N-1} (1/N)^1 \approx 1 - 2/e \approx 26\% \approx 2^{-2},$$

then we guess B and B' and obtain an overall probability of

$$(1 - 2/e)2^{-32-32} \approx 2^{-66}.$$

For these 26% of keys in Family B where this can happen, the success probability is 2^{-64} and the key recovery is particularly easy. Here is how we proceed.

Fact 39 (Attack on Family B Keys). For a fraction of at least $0.26 \approx 2^{-2}$ keys in Family B, given 2^{32} CP, the attacker can break GOST in total time of about 2^{91} GOST computations. The memory required is to store the 2^{32} texts.

Justification: First we need to find all the points for 32 rounds such that $Enc_k(A) = \overline{A}$. The time to do it is only about 2^{32} steps, which in practice is much less than 2^{32} GOST encryptions.

We expect that on average about 5 such points will be found, 2 arising due to our attack, and three more totally unrelated fixed points are expected an average for any 4-fold iterated permutation such as $\mathcal{E}^4$. We refer to see [11, 39] for detailed work and explanations on fixed point statistics in iterated permutations: one fixed point on average is expected for any permutation, and two more on average will be inherited, as being fixed points for $\mathcal{E}$ and for $\mathcal{E}^2$.

In order to filter out the fixed points which are useful for the attack we need to check typically about $\binom{5}{2} \approx 2^3$ cases. This allows us to identify the right subset of points $\{A, A'\}$. For each case (A, A') out of about 2^3 cases which we need to check,we guess B and B'. Then we apply Fact 3 to these two pairs for 4 rounds $\mathcal{F}(A) = B$ and $\mathcal{F}(A') = B'$ and recover the key in time of 2^{24} GOST encryptions. Each key candidate is then checked against additional P/C pairs for 32 rounds, and the number of false positives which need to rejected is about 2^{66}, and the time needed to reject them is negligible compared to 2^{24}. Thus the total complexity of our attack is about $2^{32+32+3+24} \approx 2^{91}$ GOST computations.

N.4 Cycling Attacks vs. Slide Attacks on Family B Keys

The simplest (classical) form of slide attacks applies for ciphers with perfect periodicity, where the whole encryption process is a k-th iteration of a smaller component $\mathcal{E}$. They work by guessing certain P/C pairs for a reduced-round cipher, and getting additional pairs through sliding, see [26, 2, 3].

However when the block size is smaller that the key size, the sliding attack are not very good, because it is possible to obtain P/C pairs for the smaller component $\mathcal{E}$ **without guessing** any initial relations on $\mathcal{E}$. This can be done directly by exploiting the cycles for the permutation $\mathcal{E}$ which can be easily computed and analysed. We are going to describe and apply this method here, and we will discover that in the case of this particular $\mathcal{E}$, it is much easier than for other permutations with similar structure and key size. This is because particular permutation has an anomalous cycle structure, where all cycles have lengths much shorter that expected. This in turn being due to the internal structure of $\mathcal{E}$. We have the following result:

Fact 40 (Cycle Structure of $\mathcal{E}$ for Family B). Let $\mathcal{E} = \mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$ where $\mathcal{F}$ is an arbitrary keyed permutation. The typical cycle for $\mathcal{E}$, by which we define the cycle on which we are likely to be if start from a random point X has about 2^{32} points, instead of about 2^{63} for a random permutation. The chances that X is on a cycle with much higher size are very small sizes of at least 2^{32+t} occur with probability which decreases very quickly with t at a double exponential speed.

Justification: This fact is due to the fact that if we iterate $\mathcal{E}$, and if a reflection occurs inside one of $\mathcal{S}$ functions, by which we mean that we encounter a symmetric value, and this $\mathcal{S}$ has no effect, then further iteration is going to effectively undo, step by step, all the previous steps. More precisely, we recall from Fact 35.9. that for every $k \geq 0$ there exists $\mathcal{H}$ such that we have

$$\mathcal{S} \circ \mathcal{E}^k = \mathcal{H}^{-1} \circ \mathcal{S} \circ \mathcal{H}$$

and moreover we have the following precise decomposition:

$$\begin{cases} \mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-l} \circ \mathcal{S} \circ \mathcal{E}^l & \text{when } \ k = 2l \\ \mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-l} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{E}^l & \text{when } \ k = 2l+1 \end{cases}$$

Depending on which $\mathcal{S}$ the reflection occurs, we will be in the first and or the other case, and it is obvious that $\mathcal{E}$ will start going backwards revisit all points previously visited or their symmetric images, and return to the initial point or its symmetric image, after a reflection and the same number of steps. In other words, given any starting point X, and for a random k, we have $\mathcal{S} \circ \mathcal{E}^k = \mathcal{H}^{-1} \circ \mathcal{S} \circ \mathcal{H}$, and if we consider all possible $k = 1, 2, 3, \ldots 2^{31}$ with a large probability p a reflection will occur for some k and we will obtain that $\mathcal{E}^k(X) = \overline{X}$ for one $k \geq 2^{31}$. We can note that since $\mathcal{E}$ has two applications of S each, this probability p is already about 60 % for 2^{31} applications of $\mathcal{E}$. Then this process continues until another reflection occurs, and further applications of $\mathcal{E}$ will join the initial path and form a complete cycle. Thus we get cycles with two reflection points, and with overall expected cycle size being about 2^{32}. Moreover cycles much longer than 2^{32} are unlikely to happen: the chances that X is indeed on a cycle with size of at least $2^{32+t} = 2^{t+1} \cdot 2^{31}$ will decrease double exponentially fast with t, because a segment of size 2^{31+1+t} without any reflection occurs with probability $p^{2^{t+1}}$.

Remark 1: Reflections can occur on boundaries of $\mathcal{E}$, or inside some $\mathcal{E}$ with $\mathcal{E}(Z) = \overline{Z}$ for this particular application of $\mathcal{E}$. Generally we expect to have two reflections inside each cycle, this cannot however be guaranteed, there may be shorter cycles which contain one or zero reflections, which are natural cycles which occur by chance.

Remark 2: The cycle structure of $\mathcal{E}$ is rich and fascinating. From our proof it follows that we expect that very frequently but not always, the points X and $\overline{X}$ will lie on the same cycle. It also happens in Fact 37 which is a special case with a very short cycle, with one symmetric point. For example, this will happen each time a reflection occurs inside one of the applications of $\mathcal{E}$ with $\mathcal{E}(Z) = \overline{Z}$ at this point. Indeed if $\mathcal{E}(Z) = \overline{Z}$ for at least one point Z lying on a given cycle, then it is possible to see that for every point T lying on this cycle, $\overline{T}$ is also on the same cycle and moreover the points are visited in exactly the opposite direction when walking on the cycle. This comes from the fact that $\mathcal{S} \circ \mathcal{E}^k$ is an involution and $\mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-k} \circ \mathcal{S}$, cf. Fact 35.8. Thus we have if $\mathcal{E}(Z) = \overline{Z}$ then for any $k \in \mathbb{N}$ we have $\mathcal{S}(E^{k+1}(Z)) = \mathcal{E}^{-k}(\mathcal{S}(\mathcal{E}(Z))) = \mathcal{E}^{-k}(Z)$.
More generally, if just for one point on the cycle Z the point $\overline{Z}$ lies on the same cycle, for all points on this cycle, their symmetric image lies on the same cycle. The proof is the same as above: if $\mathcal{E}^k(Z) = \overline{Z}$ then $\mathcal{S}(E^{m+k}(Z)) = \mathcal{E}^{-m}(\mathcal{S}(\mathcal{E}^k(Z))) = \mathcal{E}^{-m}(Z)$. In particular if k is even $k = 2l$ this gives us a situation where $\mathcal{S} \circ \mathcal{E}^k = \mathcal{E}^{-l} \circ \mathcal{S} \circ \mathcal{E}^l$ and the reflection occurred at the border

of $\mathcal{E}^l(Z)$, while previously we have seen one example where a reflection occurred inside $\mathcal{E}$. Both cases are possible therefore.

Remark 3: There are also rare cases where the points X and $\overline{X}$ will lie on two distinct cycles. For example this happens in Reduction 15 where X and $\overline{X}$ are two distinct fixed points for $\mathcal{E}$. Moreover in the case when the points X and $\overline{X}$ lie on two distinct none of these cycles contains any symmetric point, and none of these cycles contains any couple of points $Z, \overline{Z}$ which would already force the two cycles to merge, as shown above. Then it is easy to see, that both these cycles are of the same size and contain exactly the symmetric images of all the points from the other cycles, visited in exactly the order but in the opposite direction. We call this situation 'twin cycles'. Moreover none of these cycles contains any reflection, because this will lead for the two 'twin cycles' to merge totally, as shown above. This means that this situation of disjoint 'twin cycles' is quite rare and occurs only for small cycles without any reflection whatsoever, which are a small minority of cycles.

Remark 4: Thus for a great majority of cycles, if the size of the cycle is odd, and if it contains all the symmetric images of all the points, it means that the cycle must contain an odd number of symmetric points, otherwise the size would be even. However because we expect that there are two reflection points we expect that one reflection occurs on a boundary of $\mathcal{E}$ and a second reflection occurs inside some $\mathcal{E}$, and there is no more reflections and no more symmetric points.

N.5 A Simple Cycling Attack on Family B Keys

In the real life the attacker does not have access to $\mathcal{E}$ but to $\mathcal{E}^4$. This however allows in many interesting cases to easily reconstruct whole cycles for $\mathcal{E}$ and thus get many P/C pairs for 8 rounds without any effort. More precisely:

Reduction 18 (Cycling Reduction for Family B). Given 2^{32} chosen plaintexts for GOST with keys being in Family B, it is possible in time of roughly 2^{32} operations, to obtain about 2^{32} P/C pairs for $\mathcal{E}$ which are simultaneously correct with overall probability of about 2^{-1}.

More precisely, for any point X chosen by the attacker, with probability at least $1/2$ over X, we can compute a cycle which contains X, and be able to compute $\mathcal{E}^k(X)$ for any $k \in \mathbb{Z}$.

Justification: Let $S \approx 2^{32}$ be the size of the cycle on which lies the point X for the permutation $\mathcal{E}$. Moreover, with probability $1/2$ over X this integer S is odd and $GCD(S, 4) = 1$. We have verified experimentally with random GOST keys from Family B and with many random starting points X that $S \approx 2^{32}$ is a reasonable assumption and that the probability that S is odd, is indeed large enough and close to $1/2$ so that our attack will work.

We recall that $\mathcal{E}^4 = \mathcal{S} \circ Enc_k$ therefore the attacker has access to $\mathcal{E}^4$. The attacker starts with $X_0 = X$ and computes:

$$X_{i+1} = \mathcal{S}(Enc_k(X_i)) = \mathcal{E}^4(X_i)$$

The attacker obtain thus a cycle the size of which divides S and if S is odd, we have $GCD(S, 4) = 1$ and the cycle size for $\mathcal{E}^4$ is equal to S. Moreover we are

in a cyclic group of a known size and can easily compute $\mathcal{E}$ for any point lying on our cycle:

$$\mathcal{E}(A) = (\mathcal{E}^4)^d(A)$$

where we define $d = 4^{-1} \bmod S$ in the same way as in the RSA cryptosystem. Thus in total time of essentially 2^{32} steps, the attacker can compute a table of 2^{32} P/C pairs for 8 rounds of GOST $\mathcal{E}$.

Remark 1: The attacker is **not** able to see if S is odd, but if S is odd, which happens with probability of about 2^{-1} then his resulting table of about 2^{32} values for $\mathcal{E}$ is going to be correct.

Remark 2: There will be cases when $S = 2T$ and T odd, and the attacker will see a sub-cycle of length T, and can be mistaken to believe that $S = T$,

Remark 3: It is possible to see that if at the start we choose X symmetric, it increases the probability that the cycle size S will be odd and thus results in a higher probability that our attack will work.

This Reduction 18 will be used to recover keys for GOST Family B given only 2^{32} chosen plaintexts. This is done as follows.

Fact 41 (Cycling Attack for Family B). For any GOST key in Family B and given about 2^{33} CP we can recover the key in time of 2^{89} GOST computations.

Justification: We apply Reduction 18, start from a symmetric value X, and obtain about 2^{32} P/C pairs for $\mathcal{E}$ which are simultaneously correct with overall probability of about 2^{-1}. This in time of roughly 2^{32} operations.

Then we need to guess the internal value for just one application of $\mathcal{E}$ such that $\mathcal{E}(Z) \neq \overline{Z}$ which guarantees that if we guess the internal value, we will obtain two distinct P/C pairs for 4 rounds. The we apply Fact 3: the 128-bit key can be found in time of 2^{24} GOST computations and with negligible memory.

On average we need to repeat the attack for 2^1 distinct cycles hoping that S is odd for one of them. Overall the key can be computed in time equivalent to about $2^{1+64+24} \approx 2^{89}$ GOST encryptions.

N.6 Fine Improvements On The Solver Side

Until now, in our attacks on Family B keys, we only used the very simple Fact 3 in the final stage of the attack. However, in many specific cases, finer and faster algebraic attacks with SAT solver software can be found, leading to an overall faster key recovery attack on this version of GOST with 128-bit keys. We have the following result:

Fact 42 (Key Recovery for 4 Rounds and 3 KP with symmetric ciphertexts). Given 3 plaintexts for 4 rounds of GOST for which we know that the corresponding ciphertext is symmetric, one can produce a list of 2^{32} candidates for the the full 128-bit key, one of which is the correct key, in time equivalent to 2^{78} GOST encryptions on the same software platform. The memory requirements are very small. The attack works with a similar complexity for any choice of GOST S-boxes.

Justification: This is an experimental result. In this attack we do not know the ciphertext for any of the 3 plaintexts, but we assume it is symmetric, which gives 32 bits of information about this ciphertext. Furthermore we guess 57 bits of information as follows: we guess 32 bits of information about the first symmetric ciphertext, 25 bits of information about the second symmetric ciphertext, and 0 bits of information about the third symmetric ciphertext. This method of guessing bits un-evenly is the one which experimentally works the best. Then the key can be recovered in 2 seconds, which is about 2^{21} GOST encryptions on the same PC. Overall the complexity including the guessing phase is $2^{21+57} = 2^{78}$ GOST encryptions on the same PC. Given that the symmetry of the 3 ciphertexts provides 96 bits of information about the 128-bit key, the attack will produce roughly about $2^{32} - 1$ false positives.

This will lead to an important improvement in the running time of our best attack so far (cf. Fact 41).

Fact 43 (Improved Cycling Attack for Family B). For any GOST key in Family B and given about 2^{35} CP we can recover the key in time of 2^{81} GOST computations.

Justification: As in the previous attack, each time we apply Reduction 18, we start from a symmetric value X, and obtain about 2^{32} P/C pairs for $\mathcal{E}$ which are simultaneously correct with overall probability of about 2^{-1} (only if S was odd). This in time of roughly 2^{32} operations, each time.

Here on the contrary to the last attack given in Fact 41, we will be interested in pairs with $\mathcal{E}(Z) = \overline{Z}$, while in addition Z being not symmetric. Each time we do the above steps, we expect to find one such value on average, for each set of 2^{32} CP used in cycling. In these cases, due to the structure of $\mathcal{E}$ we get one plaintext for 4 rounds for which the ciphertext after 4 rounds is symmetric (but unknown).

We need to repeat this 3 times, moreover we need that S is odd simultaneously in all the 3 cases. Therefore we need to about 6 sets of 2^{32} CP, out of which we need to select three for which all S are odd, which requires on average 2^3 trials, and for each trial we apply Fact 42 to get a list of 2^{32} possible keys in time equivalent to 2^{78} GOST encryptions. Overall we get a list of 2^{32+3} possible keys in time equivalent to 2^{78+3} GOST encryptions. In a further step of the attack we check all these keys against some pairs for 32 rounds, which (as usual) takes negligible time compared to 2^{78+3} GOST encryptions.

Overall our attack finds the right 128-bit key given less than 2^{35} CP and in time of 2^{81} GOST encryptions.

N.7 Summary of Results on Family B and Other 128-bit Keys

In the following table we compare various attacks with focus on both families of 128-bit keys studied and comparison to some other families of keys.

Key size/type	256		Direct128	Inversed128			
Key family	Normal	Family 3	Family 0'	Family B, Appendix N			
Reduction cf.	Red. 3	Fact 28	Red. 14	Red. 15	Red. 17	Reduction 18	
Attack	Sect. 11	Fact 29	Fact 32	Fact 36	Fact 39	Fact 41	Fact 43
The density d	0.63	2^{-64}	2^{-160}	2^{-129}	2^{-130}	2^{-129}	
From (data 32 R)	2^{64} KP	2^{64} KP	2^{32} CP	2^{64} KP	2^{32} CP	2^{33} CP	2^{35} CP
Obtained 8R	**3** KP	**4** KP	1	≥ 2	≥ 2	2^{32}	2^{33}
Selected 8R	**3** KP	**4** KP	-	1	2	1	3
Valid w. prob.	2^{-96}	2^{-1}	-	2^{-2}	2^{-2}	2^{-1}	2^{-3}
Obtained 4R	-		1	2	2	2	3/2
Valid w. prob.	-		2^{-1}	2^{-66}	2^{-66}	2^{-65}	2^{-3}
Storage bytes	2^{67}	2^{67}	-	2^{67}	2^{35}	2^{36}	2^{38}
♯ False positives	2^{128}	2^{64}	2^{64}	2^{64}	small	2^{64}	2^{32}
Attack time 32 R	2^{206}	2^{95}	2^{65}	2^{90}	2^{91}	2^{89}	$\mathbf{2^{81}}$
Cost of 1 key, if	2^{207}	2^{159}	2^{66}	2^{91}	2^{93}	2^{90}	$\mathbf{2^{83}}$
key diversity $\geq$	$2^{0.7}$	2^{64}	2^{32}	$2^{0.7}$	2^{2}	2^{1}	2^{2}

Table 5. Comparison of our attacks on 128-bit keys compared to some other attacks

Comparison of Different Attacks on Family B Keys: In this paper we presented 4 different attacks on GOST keys of Family B with similar time complexity. Our first attack (cf. Fact 36) required 2^{64} KP which is not very realistic. Our second, third and fourth attack require only about 2^{32} CP. Our third and fourth result (cf. Fact 41 and Fact 43) are better than the second result because they work for arbitrary Family B keys, not 26% of them as in the second result (cf. Fact 39). Then the time complexity gets smaller in each case with a moderate increase in data complexity. This last improved attack of Fact 43 is arguably now the best known attack on GOST Family B, and currently will also be the best known attack with key density $d = 2^{-128}$, cf. Table 3.

We omit possible improvements to the second result (cf. Fact 39) by using also Fact 42, which will not be very interesting because it would work even for a smaller fraction of keys than 26% while our best attack works for all keys.