Algebraic Complexity Reduction and Cryptanalysis of GOST

Nicolas T. Courtois

University College London, Gower Street, London, UK

Abstract. GOST 28147-89 is a well-known Russian government encryption standard. Its large key size of 256 bits at a particularly low implementation cost [83] make that it is widely implemented and used [70, 105, 66, 83, 88]. In 2010 GOST was submitted to ISO to become an international standard. 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. [63, 101, 100, 8] and by the additional non-linearity and diffusion provided by modular additions [63, 84]. At Crypto 2008 the hash function based on this cipher was broken. Yet as far as traditional encryption applications with keys generated at random are concerned, until 2011 no cryptographically significant attack on GOST was found.

In this paper we present several new attacks on full 32-rounds GOST. Our methodology is derived from the idea of conditional algebraic attacks on block ciphers [25, 20] 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/MITM attack on 8 or less rounds. We obtain some 50 distinct attacks faster than brute force on the full 32-round GOST and we provide five nearly **practical attacks** on two major 128-bit variants of GOST (cf. Table 6).

Recent updates: Our latest attacks combine all of [higher-order] truncated differentials, complexity reduction, [approximate] fixed points, reflections, MITM and software/algebraic attacks. Single key attacks are summarized in Table 3 p. 53 and Table 7 p. 153 and the fastest of these is a differential attack in 2^{179} by Courtois [40, 39]. In the multiple random key scenario the cost of recovering one full 256-bit GOST key decreases in a spectacular way at the expense of further growing data requirements. In Table 4 page 128 we summarize all our attacks in this space. Our fastest attack achieves **a nearly feasible T=** $\mathbf{2^{101}}$ (cf. Section 28.6 and [34]).

Key Words: Block ciphers, Feistel schemes, GOST, ISO 18033, key scheduling, self-similarity, differential cryptanalysis, advanced slide attacks, fixed points, reflection attacks, black-box reductions, low-data complexity, MITM attacks, algebraic attacks, SAT solvers, singel-key attacks, multiple-key attacks.

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 some 50 different attacks on GOST obtained with this methodology. Most of these attacks were developed in 2010/2011 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 have made public his first attack on full GOST [74] with time complexity of 2^{225}. Each single attack in our submission was already faster than this attack, and our fastest attack had time complexity of 2^{216}. It also contained several weak-key attacks and a practical attack on one 128-bit key variant. 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. In May 2011 an abridged version of this paper was submitted to Asiacrypt 2011 and it contained **two** distinct attacks with time complexity of 2^{216}. An alternative final step for one of these attacks was presented by Shamir *et al* at FSE 2012 with complexity of about 2^{192}, see [50]. In the meantime a better single-key attack on GOST was found by Courtois with $T = 2^{179}$, cf. [39, 40]. Principal single key attacks on GOST are summarized in Table 3 page 53 and Table 7 page 153.

In 2012 there were many major updates of this paper which greatly **reduced** the time or/and data complexity. We have also seen a major paradigm shift in understanding the security of GOST. Ciphers are NOT used with single keys. On the contrary. We have discovered that the multiple key scenario is stronger more practical and much more versatile than the single key scenario. Many of our earlier "weak" key attacks can now be transformed into "real" attacks which are able to recover ordinary GOST encryption keys generated at random. These attacks are very competitive and faster than any other known attack from the pragmatic perspective of the total price paid per each 256-bit key recovered. Our recent attacks combine all of differential, complexity reduction, reflections, fixed points, MITM and software/algebraic approaches. In Section 26 we introduce a new concept of approximate fixed point biclique which is a higher-order differential property and allows to propose new attacks with only about 2^{101} per key for breaking some keys in a population of 256-bit keys generated at random, this at the expense of further growing data requirements (over many keys). The data quantity per key is frequently just 2^{32}. All these are not single key attacks yet many are vastly more realistic attacks on GOST than 2^{192} from [50] and 2^{179} of [39, 40]. Our best attacks are summarized in Table 4 page 128.

This is our **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 various self-similarity properties of GOST can be exploited in a variety of distinct ways leading to some 50 distinct non-trivial attacks on full GOST faster than brute force.

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. [83]. The fact that GOST was believed to be secure until 2011, and now can be broken in so many different ways including several arguably nearly practical attacks 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 [12, 19] 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 [19, 15, 48] 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 15.1 and [50]). 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. [65], unlike United States DES which could only be used to protect unclassified information. Though some of our attacks are becoming remotely feasible, it is still too expensive in order to actually be able to decrypt GOST communications in practice. It is however wrong to believe that this is only academic research. It is believed that actual versions of GOST used in practice may have additional properties which could make them much weaker than most versions studied in this paper. Some weaker versions o GOST we can already break and the attacks tend to be practical: 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 101 on page 145.

The most likely impact of this research (and also other recent works on GOST [74, 50, 27, 36–38, 76]) 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 other upgrades in financial systems and critical data storage infrastructure.

3 Dark Side of Research: Peer Review

A first version of this paper was submitted to Crypto 2011. It was already an original 28-pages pages with five distinct single-key attacks which break GOST faster than brute force and much more.

Being a referee in a scientific conference is a difficult and challenging task. One referee of this paper have written: *"The paper is decently written, and the attack is easy to follow. However, there is a big, big, big problem : this attack, which is the main contribution of the paper has already been published at FSE'11 (it was even the best paper), just a few days before the CRYPTO submission deadline".*

There are two major problems with this statement.

First the referee claims that this attack is THE SAME as the Isobe attack from FSE 2011. This is **totally incorrect**. Moreover the precise attack which referee wrote about was very clearly presented as an example of "bad" attack, something which is maybe not even an attack given its very large memory requirements. Interestingly it is not obvious at all to see why an attack such as Isobe attack is interesting and should be published cf. [74]. Arguably only improved versions of Isobe attacks with less memory are really interesting, cf. [50].

In comparison our "bad" attack was just a much simpler and faster alternative in the same category, even though we never tried to promote it, as it is clearly not excessively good compared to just any other of our attacks. Interestingly absolutely all the attacks in our submission without exception (including the "bad" attack) were actually both simpler and strictly faster (!) than the Isobe attack published at FSE 2011. Our paper already contained in early 2011 several other non-trivial single-key attacks with various parameters, three weak-key attacks the best of which had time complexity of 2^{120} and a practical attack on one 128-bit key variant with complexity below 2^{70}. It has also introduced a highly innovative and powerful methodology for block cipher cryptanalysis which have later produced some 50 distinct attacks on GOST. How can all this be the same as Isobe attack?

Moreover and secondly, how can the referee claim that *"the main contribution"* of this paper is this attack? We are talking about a totally inessential MITM attack in the paper, strictly the worst attack in the paper, which takes a few lines to describe and which does not illustrate the methodology of the paper, on the contrary. Our paper has specifically explained that this attack needs to be discarded due to excessively large memory requirements. In the submitted version we read: *"Important: If we consider 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 [...] 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"* . To summarize the referee did not do very good work.

The best attack in this paper already had time complexity of 2^{216} which is much faster than Isobe attack in spite of larger data complexity. In May 2011 an abridged version of this paper was submitted to Asiacrypt 2011 and now it contained **two** distinct attacks with time complexity of 2^{216} which again were excessively good for the attacks known at this moment. In this submission the "inessential" attack we moved to the appendix. It was again refereed with extremely little care and one referee have again complained about why the explanation of this same "bad" attack was moved Appendix. Moreover he claimed that this attack (which was clearly an accessory result or not a result at all) was "the core of this paper".

The same referee at Asiacrypt 2011 wrote: *"I think that the audiences of Asiacrypt will not feel it is interesting".* This is the biggest insult to cryptography research we have ever seen. The fact is that half of papers accepted at this Asiacrypt are about things about which nobody ever heard about, not even professional cryptologists (say JH42, Armadillo,theory, incremental research, etc.), not to say it would interest anybody in the industry or government circles.

In contrast, we can wonder how many times it ever happened at Asiacrypt that a military-grade cipher, and an official government standard of a major country, used by large banks, implemented in SSL, was broken, while being in the process of being standardized by ISO to become a global industrial standard? This because – on the face value – it clearly appeared to be better and more competitive than any "Western" cipher (on the development of which tens millions of euros were spent?) Impacting potentially all of: national critical infrastructures, key financial systems and even ordinary computer software. Not many times: it seems that an event as described has very rarely happened in some 20 years of Asiacrypt. Interestingly AES got broken at the same Asiacrypt by an academic attack only very slightly faster than brute force. However this has happened after it was standardized by ISO. GOST was broken at the right moment. It could be worth billions of dollars to fix problems due to GOST.

Moreover while most modern cryptography research is about fiction, GOST has some serious historical significance. It is maybe THE ONLY cipher which would be worth cryptanalysing. Maybe the only cipher on the cryptanalysis of which billions of dollars could be and should be spent. One might suspect that GOST has been used for decades by some if not all of the most dangerous regimes on this planet (Soviet Union, Russia and regimes traditionally supplied by Russia such as Iran, North Korea, Syria, etc). Arguably it would be in the broad public interest to be able to uncover the content of such communications even if this is possible 30 years after the facts.

Important subsets of this paper were also refereed at other major conferences and have received some very useful feedback, especially at specialist conferences such as FSE and Indocrypt, and we need to thank one referee to make us improve the data complexity in several attacks in this paper, which also inspired us to realize that one can better. As a consequence we have invented a dozen of more advanced, better and faster attacks very recently. At occasions we have received a few more really silly comments and remarks... Some of these are answered in detail in Section 22.2.

4 Structure and Organization of This Paper

In Section 5 we describe what was known about GOST until now. In Section 6 we explain brute force attacks on a block cipher with small blocks such as GOST. In later Section 20.3 we comment on the data complexity of the attacks.

In Section 7 we work basic low-level algebraic and logical optimizations in view of developing software low-data complexity attacks.

In Section 8 we explain the concept of conditional algebraic attacks on ciphers with guess-then-determine and amplification, and in Section 8.1 we introduce the general concept of Algebraic Complexity Reduction which will be further developed starting just before and in Section 14.

In Section 9 through Section 12 we study various Guess-Then-Algebraic/Software attacks on up to 8 rounds of GOST. which are summarized in Table 1 on page 25. Additional variants are studied in Appendix J.

In Section 13 we go back to study of high-level properties of GOST and describe GOST cipher algebraically as a composition of permutations implied by the key scheduling. At his moment we view GOST as a composition of black boxed with for example 8 rounds each. We discuss the crucial Reflection and Involution properties inside GOST. In Sections 13.8 through Section 13.7 we relate these properties to well-known historical results on composition of permutations known since the 1930s. We emphasize the fact that these properties allow one to deduce values inside a composition of permutations, which is one of the key guiding principles in this paper. This is what allows black-box reductions where the attacker can obtain a number of Plaintext/Ciphertext pairs for a reduced round cipher. An early historical example of such a reduction from the 1930s is shown in Section 13.8.

Approximate reflections will be later studied in Appendix F and approximate fixed points in Section 26.

In Section 14 we study the concept of Black Box Algebraic Complexity Reductions in detail and summarize all single key reductions and attacks in this paper.

Most reduction results in this paper are reductions from 32 to 8 rounds of GOST. However in Appendix D we study reductions for 32 to 16 rounds of GOST, in Appendices C, G and H.2 we study self-similarity attacks focusing on a subset of 16 consecutive rounds inside GOST, in Appendix E we study chosen-data reductions, and in Section 30 we study reductions from 32 to 4 rounds for 128-bit GOST variants.

In Sections 15-16 we study single key attacks with 2^{32} of data and in Sections 17-18 we study single key attacks with 2^{64} of data. In Appendix A and B we provide additional non-trivial single key attacks. In Section 19 we discuss what makes GOST vulnerable against our attacks and explain what are the ingredients for even more powerful attacks.

In Section 21 we study basic weak key attacks on GOST. In Section 22 we study the conversion methodology of converting single key attacks into “real” attacks with multiple keys generated at random. In Sections 23-24 we study further weak key attacks which are particularly efficient or where the weak keys are somewhat "visible" to the attacker from the data. The attacker can detect which keys are weak and break GOST efficiently.

In Section 25 we introduce attacks which mix complexity reduction and differential cryptanalysis approaches. In Section 26 we introduce a major new concept of an approximate fixed point biclique which is a multiple approximate fixed point concept provoked by higher-order differential properties and in Section 26.3 and in Appendix I we apply it to GOST. In Sections 27-28 we develop a large number of new attacks. In Section 29 we summarize all our attacks on GOST single key and multiple key attacks.

An alternative point of view on cryptanalysis of GOST which focuses on obtaining values inside the cipher and encompasses all of our self-similarity attacks such as black-box reductions but also differential cryptanalysis and many other attacks is discussed in Appendix K.

In Section 30 we provide nearly-practical attacks on two well-known 128-bit variants of GOST which results are summarized in Table 6 page 147.

In Section 32 we look at ISO standardization of GOST. In Section 33 we summarize our results and provide some concluding remarks. Single key attacks are summarized in Table 3 page 53 and Table 7 page 153. Selected best attacks with single and multiple keys are summarized in Table 4 page 128.

In Appendix L we study some simple self-similarity attacks on basic components of the GOST hash function.

5 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 [67, 68, 101]. and according to Schneier [101] this set is used by the Central Bank of the Russian Federation. They also appear in more recent RFC4357 [69] as being part of the so called "id-GostR3411-94-TestParamSet". A source code was included in [101] however this code specifies apparently a wrong (reversed) ordering of the S-boxes compared to later code contained in Crypto++ library [105] which is also faster and contains a set of test vectors. 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.

Schneier reports that some Russian system manufacturers would use S-boxes generated at random [101]. Variable S-boxes could however be more costly to implement and also require more effort in key distribution and management. Consequently in practice S-boxes are likely to be fixed. There exists a Russian reference implementation of GOST which is a part of OpenSSL library and contains eight sets of S-boxes [70] 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 [16, 19, 25, 12, 13]) that the cipher is vulnerable to "software algebraic attacks" [12, 16, 19, 25, 97, 98, 20, 107, 109]. Therefore our attacks are expected to work with a similar complexity for any choice of S-boxes, see also [42, 44]. In all our attacks we assume however that the S-boxes are known. In 1998 Saarinen proposed a method which allows to efficiently recover secret S-boxes from an encryption chip through a chosen-key attack [99, 57].

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 [101]. 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 [75, 8], attacks for some well-chosen number of rounds [75, 6, 100], attacks with modular additions removed [8], related-key attacks [78, 56, 94], reverse engineering attacks on S-boxes [99, 57], and attacks on the hash function based on this cipher [72, 73, 47, 81]. GOST is also, likely to be very badly broken by side channel attacks [80]. 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 keys chosen at random and we look mostly at high-level structural attacks on GOST when it is used in encryption.

Until 2011 no key recovery attack on full-round GOST was ever proposed. Several basic attacks on GOST were developed roughly at the same time: [74] and the most basic MITM attack of this paper of Fact 24, which is much simpler and slightly faster. 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 [58, 7, 8]. 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 [73, 75]. 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 [62, 63]. The results were 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 [10] of GOST seems comparatively easier and have attracted more attention. In [63] the authors also estimate that, 7 rounds should be sufficient to protect GOST against differential cryptanalysis and also that “breaking the GOST with five or more rounds is very hard”. In addition, two Japanese researchers [100], explain that the straightforward classical differential attack with one single differential characteristic is unlikely to work **at all** for a large number of rounds. In the same paper [100], more advanced differential attacks on GOST are described. They exploit sets of differentials which follow certain patterns, for example certain S-boxes have zero differentials, other bits have non-zero differentials. These are essentially distinguisher attacks on the weak diffusion of GOST and they differ considerably from the classical differential cryptanalysis [10]: sets of differentials occur naturally with higher probability, and when they occur they give significantly less exploitable information about the secret keys. First advanced multiple differential attack was proposed in [100] allows to break about 13 rounds of GOST. Numerous recent works have tried to understand, evaluate and improve the resistance of GOST against differential and linear cryptanalysis, in view of the standardization of GOST [59–61, 95]. At the same time, in 2011-2012 many improved differential attacks on GOST have been found, allowing finally to break full 32 round GOST faster than by brute force, see [36–40]. In this paper we also exploit these properties and they are essential in our 3 fastest attacks on GOST, though many similar but slower attacks in this paper do not use any of these properties.

6 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.

Important Remark. A recent paper shows that there exists a "generic MITM" attack on full GOST with time complexity $2^{254.8}$, see [76] with 2^{256} of memory. This is a very important general attack. However the difference is only $2^{0.2}$ and moreover traditional brute force does not require any memory. Therefore it remains still valid to compare all our attacks on GOST to approximately 2^{255} GOST encryptions.

Part I

Low-Level Study Of Individual Components, Local Optimization and Early Software/Algebraic Attacks

7 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 [12, 13, 16, 19, 107, 109]. The main idea was explicitly proposed by Shannon in 1949, see [102]. For DES the idea was articulated as a method of Formal Coding [71]. The best currently known attack on DES can be found in [19]: 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 [53] and ElimLin cf. [25, 26]) or constraint satisfaction software such as SAT solvers which solve algebraic problems after conversion [17]. Many other methods have been proposed recently [97, 98, 106, 107, 109] and for one problem instance many different attack techniques do usually work to some extent, see [19] and though SAT solvers do frequently solve many practical problems where Gröbner bases run out of memory, see [17], it was also shown in [17] 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 (however they have been invented as a result of attempts to break GOST by an algebraic attack and they use algebraic equations in the form of sparse multivariate polynomial equations over small finite fields.)

7.1 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 [19] and a lot of complex encoding, conversion and solver software for algebraic cryptanalysis.

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: The method is in spirit very similar to the attacks on DES described in [19]. On method studied in [19] is to encode the S-boxes as an algebraic system of I/O relations (equations which relate Inputs and Outputs of these S-boxes). However, already for DES, the fastest attacks of this type described in [19] use a different method, with some 20 additional variables per S-box, which allow equations to be more compact and more sparse. For GOST S-boxes, important specific optimizations based on the idea of reducing their multiplicative complexity and are described in [46, 81]. In order to encode the addition modulo 2^{32} we follow the first method described in [25]. The concatenation of all these equations describing the whole cipher or a large chunk of it can solved by various conversion and solver software [53, 17, 103, 104].

Remark. As recently as in July 2012, Russian researchers have written that this attack cannot work and is a fiction. This was presented at a major Russian cryptography conference [95] held in English with post-proceedings in preparation. This is a very curious statement. Some or our attacks are highly technical and it is not obvious that they will work. However the attack of Fact 3 above is particularly easy, we have only 4 rounds!

Everybody in cryptography knows that every cipher which is "not too complex" will be broken by a software algebraic attack, see for example [19, 20, 15, 1, 109]. In fact we have never yet met a researcher in cryptography who wouldn't be aware of these. We have early XSL-like methods [12], Gröbner bases algorithms [53], basic linear algebra methods such as ElimLin [25, 26], Semaev methods [97], SAT solvers [103, 104] and dedicated conversion methods [17], and other logical methods [106, 107]. Now with 4 rounds we would expect not only that some sophisticated attacks would work. We would rather expect more or less all these methods to work. Even though our methods contain some highly non-trivial optimizations cf. [46, 35, 32, 81] and some additional non-trivial tricks, the impact of optimizations in this sort of pathologically weak case for 4 rounds is small. We can hope only for a small constant factor improvement. We are confident that any computer science student at any university could write his own formal encoding of GOST, download and run the CryptoMiniSat software [104] or some other similar software, and obtain results not much worse than our Fact 3 in few hours of work. In contrast the authors of [95] call this "Fiction 3".

7.2 Attacks on 6 Rounds of GOST

In the similar way, we obtain the following result for 6 rounds which is not needed in our later work but may give the reader an idea of what kind of complexity one may expect for 8 rounds, which will our main concern in Section 9 through Section 11 and at other places.

Fact 4 (Key Recovery for 6 Rounds). Given 3 P/C pairs for 6 rounds of GOST the 192-bit key can be recovered in time equivalent to 2^{56} GOST encryptions on the same software platform.
This attack also works for arbitrary GOST S-boxes.

Justification: This is another experimental attack with guessing of a few bits and the timing is obtained by a computer simulations.

Initially we have just used software solvers to solve these problems. This naive methodology is however is not enough to obtain really very good attacks on GOST. With time we have realized that the crucial question in this type of attacks is a combinatorial optimization question which is also an essential structural cryptographic question. It is about which bits can be guessed and which bits should be determined, or in other words what is the optimal guess-then-determine strategy, and how to implement such a strategy. This topic will be studied in further sections.

Part II

Conditional Algebraic Attacks And Amplification

8 On Conditional Algebraic Attacks on Ciphers

Algebraic attacks allow to break many stream ciphers [16, 13, 14] but for block ciphers they only work for a limited number of rounds [16, 12, 19, 20]. 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, see [25, 20] for some examples.

In many cases, for example for DES [19], 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 [25] 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. The key question is the question of "gain" or **amplification**. Which set of k linear equations can be added to a system of equations in order to generate a extra αk linear equations by some given algebraic elimination algorithm such as ElimLin [25, 26] or a Gröbner bases algorithm [53] at a given degree. The question is highly non-trivial and extends to larger subcomponents of the cipher. For example in [25], the authors show a clever method for achieving as much $4n$ linear equations from n well-chosen linear assumptions done on the internal state variables for 9 consecutive steps of Snow 2.0. which gives $\alpha = 4$. This results in an algebraic attack on the keystream generation component of Snow 2.0. and the value of $\alpha = 4$ seems very hard to improve.

In this paper we want to achieve this type of simplification, and related amplification, at a higher level. We are going to exploit 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 [6, 8, 57, 20] 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 [20], 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 [19], a particular method for algebraic description of addition modulo 2^n [25], and some clever tricks at the high-level description of the cipher as in [58, 7, 8, 6, 57, 20]. 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. [7, 8, 6, 20]) 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.

8.1 Reductions and Black-Box Reductions

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

This process is called **Algebraic Complexity Reduction**. In most cases what we get is to compute (guess or determine) many internal values inside one or several decryptions, and literally break the cipher apart into smaller pieces. The notion of Algebraic Complexity Reduction creates new important **optimisation problems** in symmetric cryptanalysis: which deals with the fundamental question of how we can reduce the complexity of a cipher in cryptanalysis to a simpler problem, with a limited quantity of data, and with greatly reduced complexity, and this in the best possible (optimal) way while many interesting and non-trivial solutions will exist.

One example of a Black-Box Algebraic Complexity Reduction from 2^{64} KP for 32 rounds of GOST, to 4 KP for 8 rounds of GOST, can be found in [27] and a lot more such examples are found in the present paper. An early historical example of an attack from the 1930s which falls exactly within this Black-Box Algebraic Complexity Reduction methodology is discussed in Section 13.8.

8.2 Optimizing The Reductions: Amplification

Reductions can be compared in terms of the number of pairs obtained, the resulting reduced number of rounds, success probability, and in terms of plaintext complexity, see for example Table 3.

A key property of these reductions is the process of so called **Amplification** which is inspired by earlier work such as Section 6.3 of [25].

Definition 8.2.1 (Amplification, Informal). The goal of the attacker is to find a reduction where he makes some assumptions at a certain initial cost, for example they are true with probability 2^{-X} or work for certain proportion 2^{-Z}

of keys. Then the attacker can in constant time determine many other internal bits inside the cipher to the total of Y bits.

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

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

The amplification is an important question in algebraic cryptanalysis which was previously discussed in [25]. We should note that there are some difficulties in defining this ratio formally:

1. We claim that we need specific definitions for each individual cipher and for each specific attack method. For example Y can be the total number of linear equations obtained with the ElimLin algorithm [25, 19, 26] after adding a well-chosen set of X linear equations on the internal bits inside the cipher.
2. Intuitively, the higher, this amplification coefficient A is, while X and Z remain below a certain threshold, the stronger and more surprising is the attack obtained.
3. With higher values of X, the amplification can also be higher, however the attacker must limit the size of X for the whole the attack to remain fast enough overall.
4. It is very difficult to know if an attack with given parameters may exist.

In Section 13 we are going to start our study of **high-level** amplification: how some relations on larger blocks of GOST can induce some other very interesting relations. Before that, on Section 11.1 and in [35, 32] we study how the best possible amplification attacks can be achieved at the **low and medium-level** level, for up to 8 rounds of GOST. Other attacks of this type which are not in this paper can be found in [35].

Part III

Low-Level Attacks On Up To 8 rounds Of GOST With Well-Chosen Subsets of Key Bits, Guessing And SAT Solvers

9 Best Known Key Recovery Attacks on 8 Rounds of GOST with I/O Pairs

The key question in this section is given a very small number of I/O pairs for he 8 rounds of GOST, what is the best attack? We (and other researchers) have developed a number of such attacks with 2,3,4,6 and more pairs. We start by describing basic key results which are useful in this paper and only later and in Appendix J we will describe many other attacks of this type with variants and necessary background facts and methodology.

9.1 Key Results on 8 Rounds

In this Section and in Table 1 below we summarize the most important known low-data complexity attacks on 8 rounds of GOST.

Rounds	4	8		8		8		8	8
Key size	128	256							
Data	2 KP	2 KP		3 KP		4 KP		6 KP	≈ 600 KP
See	Fact 3	[50]	Fact 5	[35]	Fact 6	[35]	Fact 7	Fact 127	Fact 10
cf.	page 13	Fact 15 page 40			page 26		page 26	page 205	page 35
cf. also	[95]				Fact 16		Fact 13		
Memory bytes	small	$\mathbf{2^{43}}$	2^{46}	2^{68}	small	2^{69}	small		small
Time	$\mathbf{2^{24}}$	2^{128}	$\mathbf{2^{127}}$	$\mathbf{2^{107}}$	$\mathbf{2^{110}}$	2^{94}	$\mathbf{2^{94}}$	$\mathbf{2^{83}}$	$\mathbf{2^{50}}$

Table 1. Principal attacks on 8 rounds of GOST with 2,3,4 and more KP

We start with 2 pairs.

Fact 5 (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^{127} GOST encryptions on the same software platform and about 2^{46} bytes of memory.

Justification: Our initial result in 2011 was 2^{152}. Must better result was then obtained by Dinur Dunkelman and Shamir through a multi-dimensional advanced MITM attack, which however gets very complicated and technical in order to reduce the memory requirements, see [50] for details. In Section 12.4 we sketch a simpler (?) attack of similar type with a SAT solver and achieve a slightly faster attack than in [50] at the expense of slightly more memory.

With a similar methodology we will also obtain in Section 12.5 some very good results for 3 KP which case is very important in this paper. We have:

Fact 6 (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-Inversion attack, described in Section 12.1. Another attack with 2^{107} GOST encryptions but at the expense of MUCH larger memory of 2^{68} bytes is given in [35]. Finally, we also established that:

Fact 7 (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 [35] 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. This attack and additional useful variants of this attack are subsequently studied in Section 12.1 and in Appendix J.

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 [16, 19, 35, 32, 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 7. However in Section 12 and in [35, 50] the best known attacks on 2,3, 4 and 6 KP are fully described. We summarize key results in Table 1, however many more attacks exist, and several additional attacks not in this table is given in Section 12 in Appendix J and in [35, 50].

10 Low-Level Structural Properties of GOST

In order to study guess-then-determine attacks, we need to analyse the low-level structural properties of GOST such as how S-boxes in one round influence the S-boxes in another round, and how to predict bits inside the cipher with incomplete information about the key.

10.1 Connections Inside One Round Of GOST

GOST has 32 identical rounds such as the one described in Fig. 1 below. They differ only by the subsets of 32 key bits which they use. GOST is somewhat a structurally weak cipher: only 32 bits, a fairly small proportion, are used in each round.

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

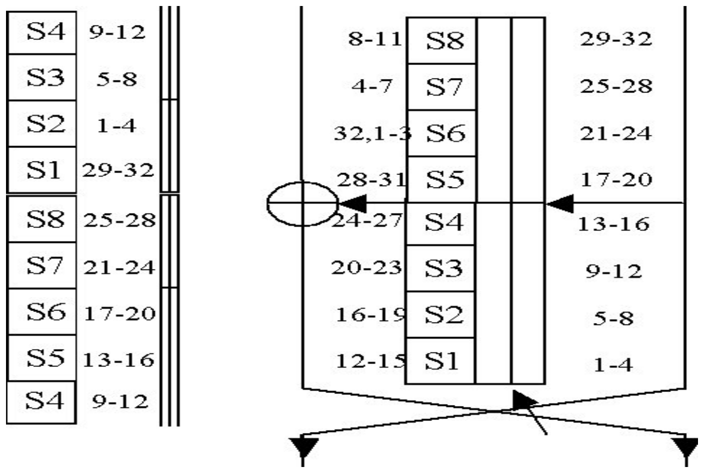


Fig. 1. One Round of GOST And Connections in The Following Round

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

10.2 Predicting Carry Bits in GOST

In this subsection we show certain basic facts studied in more details in [40] concerning the prediction of carry bits inside GOST which we need in this paper.

The key remark is that in many cases the carry bits and output bits in subsequent rounds of GOST can be computed with incomplete knowledge of all the key bits and data bits, on which this bit depends, in this and previous round. In other cases we will simply consider both the case when the carry bit is 0 and when the carry bit is 1.

We have the following basic fact:

Fact 8 (Following [40]). The input a on 4 bits of any particular S-box in GOST (for example the input of S6 in Fig. 2 below) can be computed as: $a = x + k + c \bmod 16$ where k are the 4 key bits at this S-box, c is a single carry bit with $c = 1 \Leftrightarrow x' + k' + c' \geq 16$ where x' and k' are the data and the key at the previous S-box, and c' is the previous carry bit.

Assume that the attacker knows only the outputs of only the two appropriate S-boxes at the previous round r, and x' (4 bits of the state 2 rounds earlier are also needed to obtain x'). Let d, e be respectively the most significant bits of k' and x'. Then we have:

If $d = e = 1$, we have $c = 1$ with probability 1 and we can compute a.

If $d = e = 0$, we have $c = 0$ with probability 1 and we can compute a.

If $d + e = 1$, we have $c = 0$ or $c = 1$ and we get two possibilities for a.

On average we obtain $2 \times 1/4 \times 1 + 1/2 \times 2 = 1.5 = 2^{0.6}$ possibilities for a. These possibilities for a are computed using only 5 bits of the key, the state of only 2 S-boxes in the previous round, and only 5 bits coming from 2 rounds earlier. Similarly, if we know the whole x' there are only $1.25 = 2^{0.3}$ possibilities.

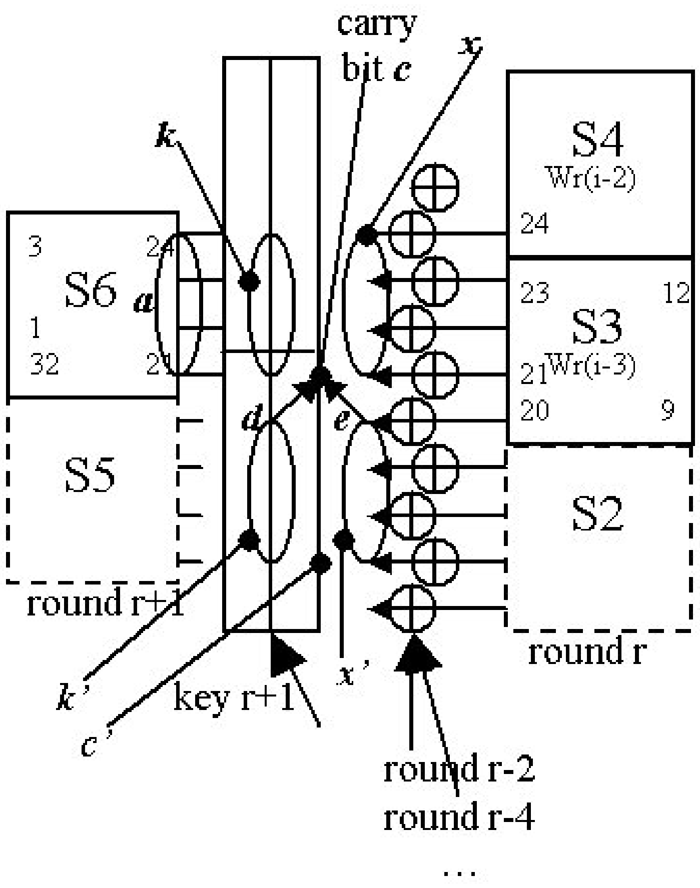


Fig. 2. Computation of the Input Of One S-box With A Carry Bit

11 Guess-Then-Determine And Software Attacks With SAT Solvers

Initially we have just used various algebraic and logical software solvers to find low data complexity attacks we needed in order to break full 32-round GOST. However very quickly we realized that the crucial question in this type of attacks is a combinatorial optimization question which is also an essential structural cryptographic question. It is about which bits can be guessed and which bits should be determined, or in other words what is the optimal guess-then-determine strategy, and how to implement such a strategy. Our key results are summarized in Table 1 on page 25 and many other results are found in the following sections, in Appendix J and in [35].

11.1 Amplification At The Low Level

Amplification is a key concept in cryptanalysis. The amplification is maybe easier to define when as in many initial steps in many of the attacks on GOST in this paper, we deal with black boxes, cf. Definition 8.2.1 page 20 and we obtain I/O relations for 8 rounds of GOST. In this section we work at the low level, when we look at a complete functional description of a cipher. The concept of amplification is equally important at this scale, though it seems actually trickier to define formally as we will see in this section.

The question is what is the best possible software attack with tools such as the ElimLin algorithm [25, 19, 26] SAT solvers [46, 17], Gröbner bases [53] and other [97]. In all these algorithms we observe the phenomenon of Amplification in various forms. For example we can study and count linearly independent linear equations and try to amplify their number by the ElimLin algorithm, see [25, 19, 26]. When the ElimLin algorithm is itself the last step of the attack, or if the SAT solver is the last step of the attack, this amplification phenomenon becomes very important. We observe an avalanche-like phenomenon where more and more new linear equations are generated in the ElimLin algorithm, until the system is solved. Similarly, with SAT solver there is a point of phase transition where the problem becomes really easy to solve.

If we want to understand algebraic cryptanalysis we need precisely to work on this face transition phenomenon itself. What happens after this threshold when the problem is just very easy to solve is less important. In this paper we focus more specifically on cryptographic attacks with SAT solvers, and on GOST, which is a nice example of a weak government standard cipher with relatively poor diffusion.

11.2 SAT Solvers in Cryptanalysis

There are two main approaches in SAT cryptanalysis or two main algorithms to break a cipher with a SAT solver:

1. **The SAT Method:** Guess X bits and run a SAT solver which, if the assumption on X bits is correct takes time T. Abort all the other computations at time T. The total time complexity is about $2^X \cdot T$.
2. **The UNSAT Method:** Guess X bits and run a SAT solver which, if the assumption on X bits is incorrect finds a contradiction in time T with large probability $1 - P$ say 99 %.
   With a small probability of $P > 0$, we can guess more key bits and either find additional contradictions or find the solution.
   The idea is that if P is small enough the complexity of these additional steps can be less then the $2^X \cdot T$ spent in the initial UNSAT step.
3. **A Mixed UNSAT/SAT Attack:** In practice maybe P is not as small as we wish, and therefore we may have a mix of SAT and UNSAT method: where the final complexity will be a sum of two terms none of which can be neglected. We will see a very nice example how a combined attack can be better than any of SAT and UNSAT methods in isolation in Section 12.1 and in [35].

11.3 Contradiction Immunity and SAT Immunity

If we want to qualify the resistance of a cipher against the two attacks described above, it is natural to define the two following quantities which are introduced in [32]:

Definition 11.3.1 (Contradiction Immunity or UNSAT Immunity cf. [32]). We define the Contradiction Immunity of a given cipher and for $M = 1$ plaintext/ciphertext pairs of the cipher as being the smallest possible number of key bits which can be fixed so that given $M = 1$ KP we can obtain a contradiction with probability at least 50 % by just examining the logical consequences of these key bits. We require this contradiction to be found in a very short time, less than 1 second for the best SAT solver available.

Similarly we define:

Definition 11.3.2 (SAT Immunity or Satisfaction Immunity, cf. [32]). We define the SAT Immunity of a given cipher and for M plaintext/ciphertext pairs of the cipher as being the smallest possible number of key bits which can be fixed so that given M KP we can compute the secret key by the best available SAT solver in a relatively short time, say less than 1000 seconds.

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

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

11.4 Main Applications of UNSAT/SAT Immunities

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

For a particularly simple and elegant example of simultaneous application of SAT and UNSAT Immunities in cryptanalysis we refer to Section 11.3.

Applications to Design of Block Ciphers. It is easy to see that the designer of a cipher can very effectively lower-bound these quantities. This will be achieved by making sure that each S-box in each round influences as many S-boxes as possible in the next round. This is not all very different than designing a cipher which is provably resistant to linear and differential cryptanalysis. Interestingly, Schneier once claimed that "Against differential and linear cryptanalysis, GOST is probably stronger than DES" [101]. Therefore we should also expect that Contradiction Immunity of DES and GOST are comparable. Happily, similar attacks with SAT solvers have been developed for both DES [19] and GOST [46]. In fact, it is obvious that the diffusion in DES is much better than in GOST and so is the Contradiction Immunity in DES. However we need to be careful about drawing any conclusions and direct comparisons do not mean much. If the contradiction immunity is 78 for 8 out of 32 rounds of GOST with 3 KP and 256-bit keys, is it better or less good, than contradiction immunity being 20 for 6 rounds out of 16 of DES with 1 KP and 56-bit keys? It is very hard to say.

11.5 Application to DES

In [32], some basic results for DES obtained by the methods of [19] are given.

Currently there is no attack on 8 rounds of DES and 1 KP which would be faster than brute force. For ultra low-data complexity attacks, 8 rounds of DES seem already secure or secure enough. It will be different for GOST, mostly because GOST has a much longer key and therefore many attacks are comparatively easier.

11.6 Application to GOST

Some basic results on the Contradiction Immunity and SAT Immunity of GOST are given in [32]. These results are constructive upper bounds. We reproduce the basic facts here and include additional facts and analysis.

For a long time we thought that the Contradiction Immunity of 8 rounds of GOST was about 128. The reason for that can be found in Fig. 4 in [40]: it is possible to see that GOST splits very neatly into two nearly independent ciphers with 128-bit key each, which are only loosely connected. With this idea it is easy to understand why a software/algebraic attack on 8 rounds of GOST with complexity of 2^{120} seems plausible and natural. However recently we found that it is much lower than 128, much closer to 64, see [32].

The goal of the attack is to find a set of bits such that if these bits are know, and given 1 pair for 8 rounds of GOST, it is possible to find a contradiction on this set of bits.

Notation, cf. Fig. 3 and Fig. 4 below: We denote by S^13 just 1 higher ranking bit at S-box 1 in a given round. Similarly we denote by S_33 the 3 lower ranking bits of S3.

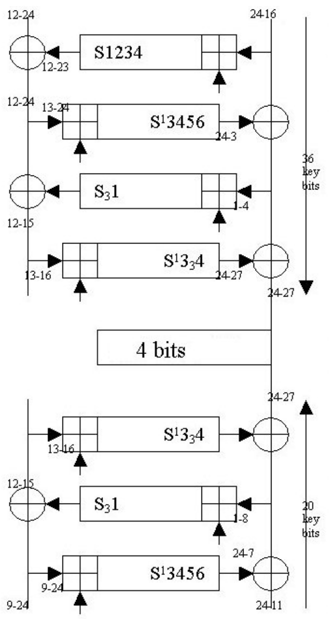


Fig. 3. A set of 56 bits which can potentially lead to a contradiction

With this set we see that there are many incertitudes in computing the middle 4 bits, in many rounds we have at least 1 bit missing to determine the result. Therefore frequently we do not obtain a contradiction. However we are close and we obtain that:

Fact 9. For the set of 56 bits depicted in Fig. 3 a contradiction with 1 KP can be found with probability about $2^{-4.1}$ over GOST keys chosen at random.

Justification: This was obtained experimentally with CryptoMiniSat. We can also observe that in many cases the contradictions are obtained for specific weaker keys, those which contain some bits at zeros or some bit at one which make the carries generated in modular additions easier to predict and decrease the number of cases in the middle. It is not correct to believe that for any key, by looking at many different encryptions we are bound to find a contradiction with certainty. This would lead to a single key attack on 8 round of GOST with time of essentially 2^{56}, as every wrong key could be rejected. Unhappily it is not as simple as that. We can however claim that:

Fact 10. For a proportion of at least about $2^{-4.0}$ GOST keys chosen at random given about 600 KP for 8 rounds of GOST the full 256-bit key can be determined in time of very roughly about 2^{50} GOST encryptions.

Justification: We sketch the attack. The data complexity is obtain by a calculation of type $(1-2^{-4.1})^{600} \approx 2^{-51}$. Thus we can eliminate most keys on 56 bits in the first step of the attack. Then we need to work with larger sets, and there is little doubt that with such a large quantity of data, most extensions of keys on 56 bits can also be eliminated. This attack requires further research.

We see that even if the contradictions occur with fairly small probability, if we have a sufficient quantity of data, the cipher will be broken in time of essentially 2^X. Unhappily this does not mean that if we dispose of 2,3,4 KP the cipher can be broken. In this case we really need to look at the Contradiction Immunity, with probability of obtaining UNSAT being around 50 %. For this we need to construct a better set with leads to less incertitude about the middle 4 bits and therefore contradictions are more likely to occur, for all keys, and not only for weaker keys.

11.7 Contradiction Immunity of GOST

We obtain the following result which also appears in [32]:

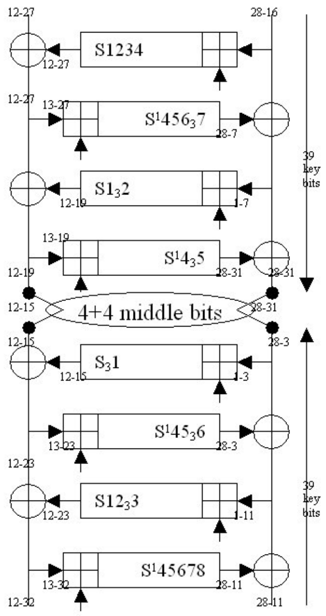


Fig. 4. Our best set of 78 bits for UNSAT

Fact 11 (Following [32]). The Contradiction Immunity for 8 rounds of GOST is at most 78.

Justification: We have constructed and tried many different sets aiming at a contradiction in the middle. Our best set is as follows (cf. Fig. 4): 0-15,47-58,64-70,111-114,128-130,175-182,192-202,239-255. The contradictions can be found in time of 0.06 s with CryptoMiniSat 2.92 software [104] with probability of about 50 %. It is easy to see that they could be obtained in essentially constant time by a dedicated algorithm with some small precomputed tables.

Remark. In fact we come remarkably close to 68 bits. If we consider the set of 68 bits later shown in Fig 5 and also used in [35], we we achieve about 39 % UNSAT with CryptoMiniSat 2.92. It is remarkably close but it does not achieve 50 % required. This may seem strange, but in order to achieve 50 %, many more key bits are needed, cf. Fact 11 above. This is because in GOST it makes a lot of sense to guess key bits for full S-boxes, and S-boxes active in one round call for other S-boxes being also active.

11.8 SAT Immunity of GOST

Unhappily, it turns out that a set which is good for UNSAT is typically NOT good at SAT. No SAT solver software we dispose of is able to find the missing bits if the 78 bits of Fig. 4 are fixed. Happily we have found sets which are very good at SAT and they are in fact smaller than 78. Our best result is as follows:

Fact 12 (Following [32]). The SAT Immunity for 8 rounds of GOST and 4 KP is at most 68.

Justification: We use the following set of bits depicted in Fig 5 0-15,51-55,64-66,128-130,179-183,192-207,224-231,244-255 which is also used in [35]. All the remaining 256-68 bits can be determined in time of about 400 seconds using GOST encodings described in [46] and with CryptoMiniSat 2.92 [104].

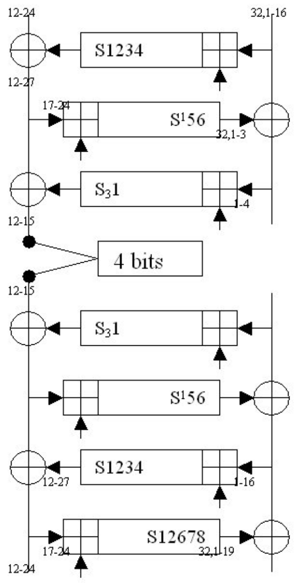


Fig. 5. Our best set of 68 bits for SAT

12 New Low-Level MITM-Inversion Attacks on 8 Rounds

In this paper we use many low-data complexity key recovery results for 8 rounds of GOST such as Fact 5. Without such attacks it is not clear at all if GOST can be broken faster than brute force and an overwhelming majority of attacks on GOST relies on these attacks. Interestingly even today some researchers still claim that they do not exist, see [95].

Early attacks we have discovered were pure software attacks, and then Dinur Dunkelman and Shamir have proposed an advanced MITM attack at FSE 2012, for the case of 2 KP and 8 rounds [50]. Subsequently we have developed many attacks of this type with 2,3,4 and more KP and for 4,6,8 rounds including some faster alternatives to Dinur-Dunkelman-Shamir attack [50]. Three attacks for 3 and 4 KP are described in [35]. In this paper we describe many more such attacks. Many of these results rely on experimental results with software and best results have been obtained with SAT solver software used at several places inside the attack.

Some of these attacks are quite complex and technical and they mix the idea of [multi-dimensional] Meet-In-The-Middle and software algebraic inversion attacks. Some are particularly simple and easy to understand. In Section 12.1 we are going to present one such result on 4 KP to start with, which illustrates the methodology perfectly take only a few lines to describe, is easy to understand, and can also be seen as a very nice example of application of the notion of SAT and UNSAT Immunities of Section 11.3.

Unhappily all known very efficient attacks on 2 and 3 KP are much more technical. They will be described below. Key results are shown in Table 1, and many additional results are given this section and in Appendix J.

12.1 A Mixed MITM-Algebraic Inversion Attack For 4 KP

For 4 KP our best attack is MUCH simpler and it was already published in [35]. Later in Section J.2 we are going to develop additional important variants of this attack.

12.2 A Mixed Attack with 4 KP from [35]

Fact 13. Given 4 KP for 8 rounds of GOST the full 256-bit key can be found in time of about 2^{94} GOST computations and negligible memory.

Justification: As in [35] we proceed as follows.

1. We use our set of 68 bits as in Fig 5.
2. We run the software 2^{68} times for all possible assignments of the 68 bits.
3. Computer simulations with the timeout of 7 seconds, a proportion of $1-2^{-5}$ of cases on 68 bits terminates with UNSAT within 2 s on average.
4. Overall, we only need to run a proportion of 2^{-5} of all the 2^{68} cases for as many as 400 seconds, in other cases it simply terminates automatically within 2 s which is 2^{23} GOST encryptions on the same CPU.
5. Assuming that all the other cases run for 400 s (some still terminate earlier) our conservative estimate of the attack time is $2^{68+23}+2^{68+31-5} \approx 2^{94}$ GOST computations.

12.3 Mixed MITM-Algebraic Inversion Attacks - Basic Fact

These attack was inspired by our previous more or less successful attempts to design an "algebraic" key recovery attack on 8 rounds of GOST [27, 46] which is needed as a last step for some 40 attacks described in this paper, and by other attacks on GOST in particular various MITM attacks [35, 32, 50]. In [32, 35, 50] we see that there exist interesting sets of bits which allow one to design a "good" attack of type guess-then-determine: guess certain bits and determine the remaining bits by software. We would like to be able to do something different that a classical MITM attack. We would still guess some bits in the middle, but then instead of using the classical approach with large memory, use a SAT solver to determine the same key bits with negligible memory. What we will show here is in fact a very general approach which can be used to transform many MITM attacks into low-memory versions with a software solver. It is a mixed MITM-Algebraic/Inversion attack in which we will guess data bits and determine key bits, and not vice-versa, like it was in traditional MITM attacks.

First we establish again a basic preliminary fact about the middle bits. We assume that we know the 71/87/103 key bits in the first 4 rounds (or in the last 4 rounds) as shown in one of the three cases depicted in Fig. 6. We want to be able to compute these 24/32/40 bits or produce a short list of possible values on 24/32/40 bits. Then it is easy to see that:

Fact 14. Given the 71/87/103 key bits in one half of and in any of the three cases depicted in Fig. 6 and the plaintext or ciphertext for one sigle encryption, there are on average about $2^{3.6}$ possibilities to determine the 24/32/40 bitsmiddle bits U, V, except for the last lower 103 bits where we obtain only $2^{2.3}$ possibilities.

Justification: The justification is nearly exactly the same in all the six cases. In roder to understand this attack better we may to look at Fig. 1 on page 27. We will first look at the first case with 71 bits in the upper 4 rounds.

1. In R2 we know all data bits 13-3 but not the bit 4 for S1. Following Fact 8 we will have about $2^{1.3}$ possibilities for the joint state of S56781 in R2.
2. In R3 we know 15 out of 16 bits for S1234 and there is no carry entering. This gives only 2^1 possibilities for state of S1234 in R3.
3. In R4 we know the lower 15 out of 16 input bits of S4567 with the higher bit of S7 missing, and want only to determine the state of S567 not of S4, and we know one data bit 12 at S4. We have $2^{1.3}$ possibilities.
4. Similarly in the backwards direction and knowing 71 key bits in the lower 4 rounds we get $2^{0.3+1+1.3+1}$ possibilities for the same middle U, V.
5. By extending our argument by exactly 1 S-box in each round we obtain the same result for 87 bits.
6. We obtain the same result for 103 bits in the upper half.
7. One case is different: 103 bits in the lower half. Here we have only $2^{1.3+1}$ possibilities.

12.4 A Mixed MITM-Algebraic Inversion Attack For 2 KP

Now we present a hybrid MITM-Algebraic attack which can be applied potentially to 8 rounds of GOST and 2,3,4 and more KP. However for 4 and 5 KP our preliminary results show it is not as good as other attacks known to us. Therefore we will just apply it to 2 and 3 KP. We proceed as follows.

Fact 15. Given 2 KP for 8 rounds of GOST we can enumerate 2^{128} possible keys on 256 bits in total time of about 2^{124} GOST computations and with 2^{46} bytes of memory.

In order to check these 2^{128} possible solutions with some data for 32 rounds, and assuming that only one key is correct, we need 2^{127} full GOST encryptions on average.

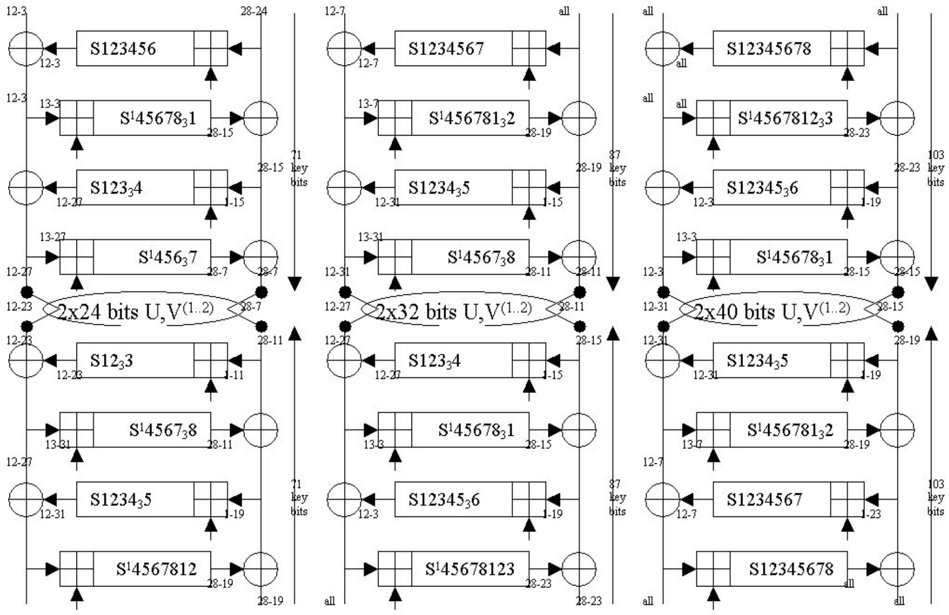


Fig. 6. Mixed MITM-Algebraic Attacks On 8 Rounds of GOST

Justification:

We proceed as follows:

1. We work with the right pane of Fig. 6 with 103+103 key bits.
2. We have $2 \cdot 40 = 80$ middle bits.
   All the steps which follow are executed 2^{80} times which will be taken into account directly in all the figures below.
3. Following Fact 14, given a fixed set of 80 middle bits there are $2^{103+2\cdot3.6-80} = 2^{30.2}$ possibilities for the 103 key bits in the upper 4 rounds.
4. By using the special case in Fact 14 we see that for the same 80 middle bits, there are $2^{103+2\cdot2.3-80} = 2^{27.6}$ possibilities for the lower 103 key bits.

5. Now we present a method to recover these key bits without enumerating all the 2^{103} cases, saving both running time and memory. Each half separately. We use the same encoding of GOST as in [46] and a SAT solver [103, 104].
6. We consider fixing 24 key bits out of 103 which are used in the first two rounds (cf. right pane of Fig. 6). We use only 16 key bits 0-15 in round 1 and 8 key bits 47-54 in round 2. Then we run a SAT solver to determine the missing 103-24 bits with a time-out of 27 seconds. There may be one, or several solutions (see later), or a contradiction.
   This takes 0.05 s on average with CryptoMiniSat 2.92 [104]. This is about 2^{18} GOST encryptions on the same CPU.
7. If the system is UNSAT, it means that the 24 bits can be rejected.
8. It may seems strange that we enumerate $2^{30.2}$ solutions by examining 2^{24} cases. The explanation is that solutions are NOT random but correlated, they are organized in clusters with several solutions sharing identical 24 bits, and due to limited diffusion and that Fact 8 deal with averages however in practice we have 0, 1 or several cases. For the first 4 rounds of GOST fixing the 80 middle bits for two GOST encryptions will already lead to frequent contradictions on 24 key bits. This happens also at a later stages of this attack: larger subsets of key bits for GOST will be excluded, while the cases which are not excluded will lead to several solutions which need to be enumerated.
9. With 24 bits fixed, our system of equations is SAT with probability experimentally only about $2^{-2.35}$, which is much lower than expected due to the clustering of solutions highlighted above. If it is SAT we will complete these 24 bits by successively 2, 4, 6, 8, 10, 12, 14 and 16 more bits and run the SAT solver again and output one solution. This is needed to make sure to fragment the space sufficiently and to output all multiple solutions, because our current SAT solver software outputs only one solution.
   The time to do this additional enumeration of solutions can be neglected compared to the current step because only with frequency of $2^{-2.35}$ we have SAT and expect on average not 1 but $3^{30.2-24} \approx 2^{6.2}$ solutions. Then we guess additional 2,4,6 and more bits if it is SAT, which will fragment the space of solutions already. Solutions will be still be clustered and most choices examined will be shown contradictory in time in 0.2 s. The average proportion of choices which still survive is decreasing quickly. Other choices which give SAT again, will require further fragmentation of the space,
   We stop at an arbitrary threshold of 24+16 even if it is SAT. At this stage the probability that a given solution obtained with a SAT solver is still not unique is going to be very small, at most 2^{-10}, and we can afford to ignore some solutions (with probability $1-2^{-10}$ the whole attack might fail because of cases which were ignored and lost in this enumeration). , which loss can be afforded).
   Thus we can enumerate all the $2^{30.2}$ solutions in total time spent being roughly about $2^{80+24+18} = 2^{122}$ GOST encryptions.
10. At this stage we enumerate $2^{80+30.2} = 2^{110.2}$ cases with 103 key bits + 80 data bits for the upper half.

11. We can further extend each of these cases by further 12 key bits in rounds 234, by extending the set by 1 S-box in each round except in the first round which is completely known now.
    Then given 103+12 key bits it is very fast to compute the resulting $16=2 \cdot 8$ new middle bits, and we estimate it requires only about $1/32 = 2^{-5}$ GOST encryptions and doing it 2^{12} times requires about $2^{110.2+12-5} \approx 2^{117}$ GOST encryptions.
12. At this stage we enumerate $2^{110.2+12} = 2^{122.2}$ cases with 103+12 key bits in the upper half and 80+16 data bits.
13. In the same way, for the same 80 middle bits, we can enumerate all the $2^{27.6}$ possibilities for the 103 key bits in the lower 4 rounds which is expected to be even slightly easier, and take less than 2^{122} GOST encryptions.
14. At this stage we enumerate $2^{80+27.6} = 2^{107.6}$ cases with 103 key bits + 80 data bits for the lower half.
15. Again we can also further extend each of these cases by further 12 key bits in rounds 567 and efficiently compute the same 16 additional middle bits, which requires slightly less than 2^{117} GOST encryptions obtained above.
16. At this stage we enumerate $2^{107.6+12} = 2^{119.6}$ cases with 103+12 key bits in the lower half and same 80+16 data bits.
17. These two enumerations require only about 2^{46} bytes of memory to store.
    In all these cases, 80 middle bits are fixed, and 16 additional data bits are variable. We can assume that the data are sorted or hashed according to the 16 new middle bits.
18. Now for each current set of 80 middle bits, and the sets stored in memory with keys on 115 bits and 16 more middle bits, we are going to fix the 16 middle bits.
    This is done 2^{96} times and then due to sorting of our data by the new 16 middle bits, we read the section of $2^{30.6-16} = 2^{14.6}$ cases with keys on 115 bits from the upper list, and a section with $2^{27.6-16} = 2^{11.6}$ cases with keys on 115 bits from the lower list.
    Thus we can now enumerate $2^{80+16+14.6+11.6} = 2^{122.2}$ possible cases with keys on 230 bits and 96 middle data bits.
19. It remains to be seen how can we efficiently extend these to 2^{128} solutions to 256 bits. This should be very easy because very few bits are missing. We expect that on average there will be $2^{128-122.2} = 2^{5.8}$ ways of extending each solution with 26 additional bits.
    In order to see that this is going to be very easy we observe that the unknown 26 key bits are in the middle 4 rounds and only 10 are in rounds 3 and 6, and further 16 are in rounds 4,5. All the other bits inside GOST are known from 230 key bits known in each case. This is like breaking a 26-bit cipher with 3 rounds and can be implemented with precomputed optimizations.
    Overall we expect that this enumeration requires only about $2^{5.8} \cdot 2/32$ GOST computations. It needs to be done $2^{122.2}$ times with an overall cost of about $2^{122.2+5.8-4} = 2^{124}$ GOST computations.

Related Research and Discussion: Our total is about 2^{124} GOST encryptions. However if we need to check the 2^{128} possible solutions with some data for 32 rounds, and assuming that only one key is correct we will really need about 2^{127} full GOST encryptions on average, and we will be able to output the correct key half way on average.

A similar result with 2^4 times higher running time and 2^5 times less memory is obtained in [50]. In fact our result can very easily be transformed into a slower attack with less memory. For example just be fixing 4 key bits in the first 4 rounds all the way through we can reduce the size of the upper list 2^4 times and make some stages of the attack 2^4 times slower. Therefore we can also achieve a result very close to the result in [50] as well as many other variants.

12.5 A Mixed MITM-Algebraic Inversion Attack For 3 KP

Now for 3 KP we have the following attack:

Fact 16. Given 3 KP for 8 rounds of GOST we can enumerate 2^{64} possible keys on 256-bit in total time of about 2^{110} GOST computations and small memory.

Justification:

We proceed as follows:

1. We work with the left pane in Fig. 6 with 71+71 key bits.
2. We have $3 \cdot 24 = 72$ middle bits.
3. Following Fact 14, given a fixed set of 72 middle bits we expect to get $2^{71+3\cdot3.6-72} = 2^{9.8}$ possibilities for 71 bits of the key in the upper 4 rounds.
4. In the same way, for the same 72 middle bits, there is $2^{9.8}$ possibilities for the 71 key bits in the lower 4 rounds.
5. We fix 16 key bits out of 71: first 16 bits in the first round. Then we run a SAT solver to determine the missing 71-16 bits. This takes 0.5 s on average with CryptoMiniSat 2.92 [104]. This is about 2^{20} GOST encryptions on the same CPU.
6. If the system is UNSAT, it means that the 16 bits can be rejected.
7. Total time spent in this step is about $2^{72+16+20} = 2^{108}$ GOST encryptions.
8. If the system is SAT which happens with probability experimentally about 2^{-5} we extend these 16 bits by up to 12 more bits, one bit at a time, checking for UNSAT, which also takes less 0.5 s in each case, is done for one more bits only for 2^{-6} of cases, for two more bits only for 2^{-7} of cases, etc... Overall we expect that the number of cases is divided by two each time and all this enumerates all the solutions for these 16 bits (on average less than one but sometimes a few) in total time of about $2^{72+16-5+20+1} = 2^{104}$ GOST encryptions which overall is smaller than the initial step above.
9. In the same way, for the same 72 middle bits, we enumerate all the $2^{9.8}$ possibilities for the 71 key bits in the lower 4 rounds in the same total time spent of another 2^{108} GOST encryptions.
10. We need a negligible quantity of memory to store these two sets of $2^{9.8}$ half-keys on 71 bits.
11. In each case, given the 71+71 bits we run a SAT solver to determine the remaining 256-142 bits. This is done $2^{72+9.8+9.8} = 2^{91.6}$ times.

12. The solution is expected to be unique, this is because there are only 2^{64} solutions on 142 bits.
13. If the set of 72 middle bits + 142 key bits is correct, this takes 17 seconds or 2^{26} GOST encryptions with CryptoMiniSat 2.92 [104].
14. If we run this in each case, the total time spent in this last step would be about $2^{72+9.8+9.8+25} = 2^{117}$ GOST encryptions. However we do NOT need to do it.
15. There are at least 72 middle bits which are fully random inputs, plus the 9.8 bits of entropy in the choice of the other bits in the above procedure.
    Therefore we estimate that a correct choice of 72+142 bits for which we have to recover the key in 17 seconds as above happens only with probability about $2^{64-72-9.8} \approx 2^{-18}$. Most of the time we expect that the same SAT solver execution as above, will in fact terminate much earlier, within about 0.1 s with UNSAT, which will be an early abort in all but 2^{-18} of cases.
16. Thus the time spent in the last step of our attack is only about $2^{72+9.8+9.8+18} + 2^{72+9.8+9.8+25-18} = 2^{110}$ GOST encryptions which dominates the whole attack. The storage required is negligible.

Related Research: Another faster attack with only 2^{107} GOST encryptions BUT at the expense of MUCH larger memory of 2^{68} bytes is given in [35].

Research Challenges: One might think that the result could be less than 2^{110}. For example consider the following well chosen set of just 39+39 bits which is obtained using the same pattern as in Fig. 6. The exact bits used are: 0-15,47-58,64-70,111-114,128-130,175-182,192-202,239-255. We can run a SAT solver 2^{78} times for all possible assignments of the 78 bits.

Even for 78 bits, a very substantial proportion of $1 - 2^{-3.5}$ of cases on 78 bits already terminates with UNSAT within 0.3 s on average. Time spent in this step is ONLY about $2^{78+19} \approx 2^{97}$ GOST computations. Then we can fix more bits, a superset of 78 bits, one new bit at a time and continue only if the result is not already shown UNSAT which is expected to happen with a non-negligible probability. At the end we will obtain a system which will be solved with a SAT solver which however will need to run only for a very small proportion of cases not yet filtered out. The only question is purely combinatorial one: what is the best trajectory extending the 78 bits with further bits? We leave it for future research. It is an optimisation problem for our already known attack in 2^{110} GOST encryptions which could potentially be improved down to maybe even 2^{98} GOST encryptions.

This would be very good and would also mean that the current best attack for 4 KP which is 2^{94} according to [35], can also be further improved. With more data even faster attacks exist and in Section J.4 we are now going to show that with 6 KP one can go down to about 2^{83}.

Part IV

High-Level Complexity Reduction And Single Key Attacks on GOST

13 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	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$\mathbf{k_0} k_1 k_2 k_3 k_4 k_5 k_6 k_7$	$k_7 k_6 k_5 k_4 k_3 k_2 k_1 \mathbf{k_0}$

Table 2. 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 [75]:

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

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.

13.1 The Internal Reflection Property of GOST

We start with the following observation which is also used in weak attacks on GOST from [75] and to cryptanalyse the GOST hash function at Crypto 2008 [73]. Both attacks also exploit fixed points, and can only work for some special (weak) keys. This property is exploited in many (but not all) of our attacks, and in a more fundamental way: without fixed points and for arbitrary keys.

Fact 17 (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: Our permutation $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ has a well-known "conjugated" structure of type $Q^{-1} \circ P \circ Q$ cf. [55, 82]. Consequently it has the same cycle structure as the swap function S and exactly 2^{32} fixed points. 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: A number of 2^{32} fixed points is a quite large number which fact plays an important role in cryptanalysis, see Section 13.3 below. For example with 2^{32} fixed points, one can expect that one of them will be symmetric, which fact is also exploited in several of our attacks see for example Fact 28.

13.2 The Involution Property in GOST

In addition we also have a strictly stronger and more powerful property:

Fact 18 (Involution Property). The last 16 rounds of GOST $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ are an involution.

In general, this may appear as very surprising, the Involution Property implies the Internal Reflection Property, see Section 13.4 below and [51, 54].

Applications. For examples of cryptanalytic applications of this Involution Property of Fact 18 we refer to Section G, Appendix A and [27], Appendix C, and Appendix H.3.

13.3 Discussion: Attacks With Large Number of Fixed Points

Having many fixed points is like having **a semi-transparent cylinder**. Sometimes, with high probability we are able to see through a whole large number of rounds of a block cipher.

This really is a crucial and powerful property in cryptanalysis. This is basically what allows to reduce the number of rounds in various attacks. It is of the ways to achieve "Complexity Reduction" through the reduction in the number of rounds in this paper and we use it countless times in this paper. Therefore in a certain way the most significant single basic remarkable property which we exploit countless times in this paper is Fact 17 above which is also known as Reflection Property by Kara [75].

13.4 Reflection Property and Involutions

Fact 17 basically says that the last 16 rounds of GOST $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ have **"an exceptionally large number of fixed points"**. Now we also have Fact 18 above which basically says that the last 16 rounds of GOST are an involution. Is there a relation between these two facts?

Yes, there is. Actually the second property implies the first property. In a very recent paper [51] from October 2014, Dinur, Dunkelman, Keller and Shamir recall a known result in combinatorics which can be found on page 596 of [54] and remark that the property of having a very large number of fixed points holds for more or less **any involution** chosen uniformly at random. Therefore it should hold for most involutions, not only for those in which we have been able to count the number of fixed points exactly as we do in Fact 17. This is exploited to develop some new attacks in [51], for example an improved attack on 18 rounds of GOST.

13.5 Some Earlier Work on Block Ciphers - Large Number of Fixed Points

Cryptanalytic attacks which exploit the fact that some permutations have a very large fixed points in modern block ciphers are not new either. For example in Biryukov and Wagner [8] Section 6 page 605 we read:

> "For a DES weak key, all round subkeys are constant, and so encryption is self-inverse and fixed points are relatively common: there are precisely 2^{32} fixed points. Note that this property will also be found in any Feistel cipher with palindromic round key sequences, so the slide attack is not the only weakness of ciphers with self-similar round subkey sequences".

13.6 Some Even Earlier Historical Work - Conjugated Permutations

Both our Reflection Property of Fact 17 and Involution Property of Fact 18 above, can be derived by an application of the well-known theorem which says that P and $Q^{-1} \circ P \circ Q$ have the same cycle structure. This theorem is due to Rejewski and has played a very important role in the historical government and military cryptography. This result is sometimes called "The Theorem Which Won World War 2", see [55, 82, 89–91, 18].

13.7 Further Research - Composition of Involutions

There is further ample cryptographic literature which exploits the cycle structure not only for involutions but also for compostions of two involutions which was also already studied and exploited by Rejewski in the cryptanalysis of Enigma as early as in the 1930s, and which still has many applications in recent research, see for example [90, 91, 9, 51, 49, 82].

The key theorem used in these works is as follows:

Fact 19 (Rejewski Theorem). Let $\mathcal{Q} \circ \mathcal{P}$ be a composition of two involutions without fixed points. The number of cycles of each length k for $\mathcal{Q} \circ \mathcal{P}$ is an even number.

Moreover these cycles are in a one-to-one correspondence induced by $\mathcal{P}$, the inverse of which is a one-to-one correspondence induced by $\mathcal{Q}$.

Justification: This theorem is due to Rejewski and appears in [93, 90–92]. This theorem was used in the 1930s in the cryptanalysis of Enigma[1]. It is also used and cited in a number of more recent papers including [9] and [49]. One proof by induction due to Rejewski himself was given on page 140 of [92]. Here is another simple proof. Let X be a point which lies on a cycle of length k, and does not lie on a shorter cycle. Then X is a fixed point of $(\mathcal{Q} \circ \mathcal{P})^k$. However because both are involutions, the same X is also a fixed point for it's inverse permutation which is simply $(\mathcal{P} \circ \mathcal{Q})^k$. Then $\mathcal{Q}(X)$ is also a fixed point for $(\mathcal{Q} \circ \mathcal{P})^k$.

We see that each time X lies on a cycle of lengthly exactly k and not on a shorter one, also $\mathcal{Q}(X)$ lies on a cycle of the same exact length, which cannot be shorter because this property holds for every point on this cycle and $\mathcal{Q}$ is bijective. Now can X and $\mathcal{Q}(X)$ ever lie on the same cycle (and the two cycles would merge)? This means that either we have $X = \mathcal{Q}(X)$ which is excluded because we assumed that Q had no fixed points, or that $X = (\mathcal{Q} \circ \mathcal{P})^k (X)$, for some smaller k, however we assumed there was no shorter cycle for X. Therefore the bijection $X \mapsto \mathcal{Q}(X)$ maps whole cycles to whole cycles which are distinct from the original cycle. Now this bijection $\mathcal{Q}$, since $\mathcal{Q}$ is an involution, is clearly one-to-one when acting on cycles and no cycle is transformed onto itself. Thus we get an even number of cycles of each length k. We also remark that the inverse mapping acting on whole cycles will be the one be induced by $\mathcal{P}$.

Applications. Composition of two involutions occurs frequently in cryptanalysis. The condition of having no fixed points can be typically relaxed, at the price of having results which are not always true but which are true with some probability, typically quite large, which is sufficient in key recovery attacks, we restart the whole attack several times.

One type of situation is that each time when we have a composition of two involutions, they are likely to have 2^{32} fixed points each [cf. Section 13.4 above] and we expect that both are likely to have a shared fixed point. This allows the

[1] If two Enigma messages (or message keys) have been encrypted with the same initial setting had the same letter in the plaintext at two positions say 1 and 4, then the relationship between the two encrypted letters was a product of two involutions which correspond to the two full encryptions at these moments, for example $\mathcal{A}_1 \cdot \mathcal{A}_4$.

attacker to infer values inside some block of encryption rounds and this fact is used many times in this paper (cf. also Appendix K.4).

In general a decomposition of a permutation as a product of two involutions is not very easy to obtain for the full 32-round GOST. Instead, in this paper the attacker needs to work harder. He can typically only infer a fairly small number of relations on concrete points, and he does get oracle access to anything which would be a composition of two involutions. Nevertheless fixed points in a composition of two involutions do occur many times in our attacks. Most applications of Fact 19 in our work are very specific degenerated special cases and we do not explicitly cite Fact 19, we work on these special cases directly. For example each time we are composing some involution $\mathcal{P}$ with $\mathcal{S}$ we get a shared fixed point with high probability. This is however a degenerated special case, and it is really and simply equivalent to looking for fixed points for $\mathcal{P}$ which are symmetric. Such situations happen many many times in this paper, cf. for example in Fig. 9 page 62 and also in other papers on GOST, e.g. [75, 77].

A composition of two involutions also occurs more easily for some special classes of GOST keys, cf. for example in Section 31. One application of Fact 19 will be later obtained as sub-point 16 in Fact 93 on page 136.

13.8 Factoring Permutations Which Involve Involutions

The Fact 19 above can be used to factor permutations. We have the following result, which is a modern version of a classical 1930s method adapted to the context of block ciphers.

Fact 20 (Rejewski Permutation Factoring Method). Let $\mathcal{Q} \circ \mathcal{P}$ be a composition of two involutions, and let $\mathcal{P}$ have p rounds and let $\mathcal{Q}$ have q rounds with $p \leq q$. We assume that the attacker has oracle access to $\mathcal{Q} \circ \mathcal{P}$. We assume that there is a key recovery attack on $\mathcal{P}$ given the fact that it has only p rounds, and that this attack requires only a limited number of P/C pairs. Then attacker can factor $\mathcal{Q} \circ \mathcal{P}$ and recover the key of $\mathcal{P}$.

Justification: We apply Fact 19 above and consider the smallest value k such that $\mathcal{Q} \circ \mathcal{P}$ has exactly 2 cycles of length k, which following Fact 19 must be related and one cycle is $X, \mathcal{Q}(\mathcal{P}(X)), \ldots$ the other is $\mathcal{P}(X), \mathcal{P}(\mathcal{Q}(\mathcal{P}(X))), \ldots$ possibly starting at some location inside the other cycle. We just need to guess which cycle is which, pick a random point on once cycle, and guess which point on the second cycle is the corresponding points. Overall with probability $\frac{1}{2k}$ we obtain as many as k correct P/C pairs for $\mathcal{P}$ which should be sufficient for key recovery.

Furthermore we have the following practical variant which is written in the sprit of the present paper which emphasizes **black box reductions** (cf. Section 14 below) from an attacker disposing of a number of P/C pairs for a larger number of rounds of a cipher to an attacker disposing of a number of P/C pairs for a smaller number of rounds.

Fact 21 (Rejewski Black Box Reduction Method). Let $\mathcal{Q} \circ \mathcal{P}$ be a composition of two involutions, and let $\mathcal{P}$ have p rounds and let $\mathcal{Q}$ have q rounds with $p \leq q$. We assume that the attacker disposes of two short cycles of length k for $\mathcal{Q} \circ \mathcal{P}$. Then the attacker can generate k pairs for p rounds which are correct with a large probability.

Justification: It is exactly the same as in previous result, except that we do no longer assume that there exists no other cycles of the same length. Yet with a large probability two cycles of the same length are mapped onto each other by $\mathcal{P}$. In a key recovery attack, we simply start again if our assumptions are wrong, and repeat until they are right and we recover the correct key.

Unhappily except in highly degenerated case, it is not easy to apply this result to full GOST.

We also have the following result:

Fact 22 (Generalized Rejewski Black Box Reduction Method). The same result applies if the attacker has access to a certain number of cycles not for $\mathcal{Q} \circ \mathcal{P}$ but for some fixed power $(\mathcal{Q} \circ \mathcal{P})^n$.

Justification: Here we can observe that $(\mathcal{Q} \circ \mathcal{P})^n$ can be decomposed itself a product of 2 involutions for example it can be seen as $\mathcal{Q} \circ \mathcal{P}^{-1} \circ (\mathcal{Q} \circ \mathcal{P})^{n-1}$ and we can apply Fact 21 directly.

Remark. This sort of decompositions appear in Fact 93 on page 136.

14 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 and call this process **Black Box Algebraic Complexity Reduction**.

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, 4 or 6 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 3) 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.

14.1 Synthetic Summary of Our Reductions For Our Principal Single Key Attacks on GOST

The following Table 3 on page 53 gives a summary of all key steps in the main single key attacks described in this paper, some other single key attacks are summarized in Table 7 page 153. Attack with multiple keys and weak keys are summarized in Table 4 page 128.

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.

Reduction Summary					
Reduction cf.	Red. 1 §15.1	Red. 2 §16	Red. 3 §17	Red. 4 §17.1	Red 5 §18
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. 9		
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	$A \mapsto B$ $B \mapsto C$	$A \mapsto B$ $B \mapsto C$ $C \mapsto D$	$Z \mapsto A$ $A \mapsto B$ $B \mapsto C$	$Z \mapsto A$ $A \mapsto B$ $B \mapsto C$ $C \mapsto D$	$A \mapsto A$ $E \mapsto \mathcal{S}(A)$
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 24	Fact 5	Fact 6		Fact 7	Fact 5
Time to break 8R	2^{128}	$2^{127}/2^{128}$	2^{110}		2^{94}	$2^{127}/2^{128}$
Storage bytes	2^{132}	$2^{39}/2^{46}$	-	2^{67}		$2^{39}/2^{46}$
♯ false positives	2^{224}		2^{192}	2^{128}		2^{192}
Attack time 32 R	2^{224}	$\mathbf{2^{223}}/2^{224}$	2^{238}	$\mathbf{2^{206}}$	2^{222}	$\mathbf{2^{191}}/2^{192}$

Table 3. Summary of our single-key attacks with a black box reduction from full 32-round GOST to a low-data key recovery attack on 8 rounds of GOST. In practice ciphers are NOT used with single keys but with multiple keys, and there is a continuous space of further attacks with growing data requirements, see Table 4 page 128.

15 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 17) 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.

Our first attack will require a lot of memory. The second version significantly less.

15.1 Self-Similarity Attacks on GOST With One Single Reflection

We have:

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 obtain two P/C pairs for 8 rounds of GOST (having full 256-bit key) and our two pairs will be correct with probability 2^{-96}.

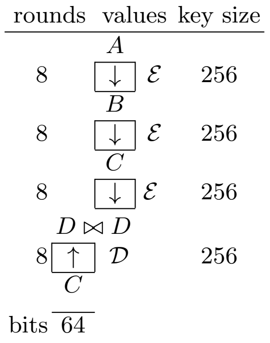


Fig. 7. The simple reflection attack with 2 or 3 KP for 8 R obtained

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}.
4. Let C be the encryption of A, $C = Enc_k(A)$. Then, since $\mathcal{E}^3(A)$ is symmetric:
$$C = Enc_k(A) = \mathcal{D}(\mathcal{S}(\mathcal{E}^3(A))) = \mathcal{D}(\mathcal{E}^3(A)) = \mathcal{E}^2(A). \qquad (2)$$
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. We do not seek to guess nor to determine D, this will be done later in Reduction 2.
7. 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 $C = \mathcal{E}(B)$.

Related Research. This method described in Reduction 1 and shown in Fig. 7 is an application of the Reflection Property initially proposed by Kara [75] and used in a weak key attack. In October 2010 we have been able to find this reduction and an appropriate second step in order to break GOST encryption faster than brute force for arbitrary GOST keys. In 2012 the same property was reused in [50] and combined with an improved final step which we revisit in this paper and propose several additional variants, see Fact 15. Actually both reductions from 32 to 8 rounds described in [50] have been initially described in earlier versions of this paper: these are Reduction 1 of Fig. 7 and Reduction 5 of Fig. 10. Consequently both attacks from [50] can be seen as improvements of two last step of two attacks of the present paper, which initially very substantially improved our timings. However later we have been able to catch up, and currently in this paper we achieve the same and even slightly lower complexities for all the attacks described in [50].

Special Cases. Unlike the great majority of attacks on GOST which work only for some keys, this method is of the very few known which leads to attacks which work for all GOST keys, On the contrary. this is also the only method known which still leaves the attacker a considerable degree or freedom: the attacks based on this method work for more or less arbitrary set of 2^{32} plaintext/ciphertext pairs out of 2^{64}. This suggests that there will be many interesting special cases of this attack. For example such that A and C are somewhat related in order the make it easier for the attacker to determine A, C with faster running times at the expense of a higher data complexity. Unhappily this is not easy because if A, C are related then for the two pairs obtained in the attack there could be more than 2^{128} solutions making the attack slower. Therefore the starting point needs to be rather the extended attack with 3KP in Section 16 in which we also guess D which is then extended in Section 25.2.

Applications of Reduction 1. Below we will present two basic methods to recover the key using this Reduction 1. We start with a very simple attack with a lot of memory which is our first and least efficient attack on GOST, which will be later compared to Isobe attack from 2011 [74]. These two attacks have as far as we know been found roughly at the same time with a precision of a few months and our attack is simpler, faster and overall better than the Isobe attack of [74] which seems to require a lot of unnecessary work.

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

Fact 23. 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.

$$\begin{array}{rccl}
\text{rounds} & \multicolumn{2}{c}{\text{values}} & \text{key size} \\ \hline
 & A & B & \\
4 & \boxed{\downarrow} & \boxed{\downarrow} & 128 \\
 & P & Q & \\
4 & \boxed{\downarrow} & \boxed{\downarrow} & 128 \\
 & B & C & \\
\text{bits} & \overline{64} & \overline{64} &
\end{array}$$

Fig. 8. Simple 2-dimensional Meet-In-The-Middle attack on 4+4 rounds of GOST

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 as shown in Fig. 7 and we get immediately an attack faster than brute force:

Fact 24. 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 23. 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 [74]. However both attacks are **not the same but different**. Our attack is much simpler and slightly faster. 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. It is also true that this and other of our attacks have been found even before their attack was submitted to FSE 2011 in November 2010 (at the time we decided not to submit our paper to FSE).

More recently, there is much more work on MITM attacks on various number of GOST rounds of GOST [35, 32] and on reducing the memory requirements of these attacks [50, 35]. Some important results which combine the MITM and software/inversion approaches are now included in Appendix 12, earlier such results can be found in [35, 32].

15.3 A Reflection-MITM-Algebraic Attack

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

Fact 25. 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 $2^{96+128} = 2^{223}$ GOST encryptions, which is 2^{32} times faster than brute force. The memory required is about 2^{46} bytes.

Justification: This is again straightforward:

1. Again given 2^{32} KP X_i, Y_i, we use the Reduction 1, as shown in Fig. 7. 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 5 which allows to enumerate 2^{128} keys in total expected time of 2^{127} GOST computations each. The total time spent in this step is $2^{96+127} = 2^{223}$ 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. Our initial attack was slower and was greatly improved due to the work of [50]. Our current version is event slightly faster though it requires slightly more memory. It requires 2^{32} known plaintexts, and the running time is 2^{223} GOST encryptions. The storage requirements are about 2^{46} bytes which point requires a lot of effort in [50]. It is also clear in Fact 15 and in [50] that there are many variants of this attack which will use less memory at the expense of slightly slower running time.

In what follows we are going to describe better attacks, with lower complexity or lower storage, or both.

16 Extended Single Reflection 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 3). 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, as shown in Fig. 7, 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, (cf. the same Fig. 7). 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 26. 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 $2^{128+110}$ 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 6 (cf. page 26) in each of 2^{128} cases tried and on average, and in total time equivalent to 2^{110} 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 6 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 6).
   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^{17} 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.

Special Cases. Again this method is of the very few known which still leaves the attacker a considerable degree or freedom: we expect that there will be many interesting special cases of this attack. For example such that A and C are somewhat related in order the make it easier for the attacker to determine A, C with faster running times at the expense of a higher data complexity. In Section 25.2 we give an example of such attack and we expect there is many more such special cases.

17 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 17) 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.

Alternative Formulation. An alternative way of viewing this assumption is as follows: there is a symmetric fixed point for the last 16 rounds of GOST. This equivalent to saying that both $\mathcal{E}^2(A)$ and $\mathcal{E}^3(A)$ are symmetric. It happens that symmetric fixed points for the last 16 rounds of GOST happen with very high probability, because the last 16 has many fixed points in the first place, see Section 31.2, while for other permutations the whole GOST it is very rare to have symmetric fixed points, see Section 21.

Fact 27 (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}.

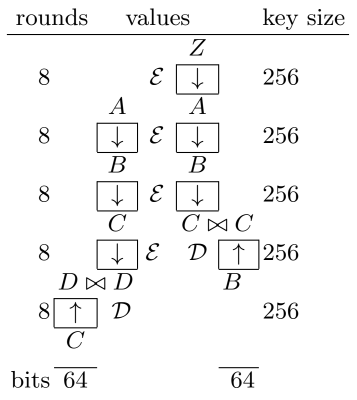


Fig. 9. An attack on GOST with double reflection

Thus we obtain the following result:

Fact 28. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{206} 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 6 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 6). 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.

17.1 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 7 to 6 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 3 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.

17.2 An Important Variant In The Multiple Key Scenario

Total cost of our attack of Fact 28 is about 2^{206} GOST encryptions and it requires 2^{64} KP. We have designed a much faster and more powerful version of this attack in the multiple key scenario. All we need to do is to assume that $Z = D$ which leads to the situation depicted in Fig. 31 which makes that the attack requires only $2^{32}$of data and occurs only for some proportion of $d = 2^{-64}$ of GOST keys. So far we have black box reduction attack which ignores the internal structure of GOST. Unexpected help comes from additional strong vulnerability of GOST w.r.t. advanced truncated differential attacks such as described in [36–40], which are no longer simple black-box attacks. It is possible to show that, within these 2^{-64} keys they exist keys with much stronger properties. We obtain an attack with complexity of 2^{117} total to find some but not all full 256-bit keys generated at random see Fact 69 page 114 or even an attack with about 2^{110} total see Fact 81.

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

So far all single-key attacks on GOST ever found exploited the internal reflections [74] and in our best single-key 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 [74–76].

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 can be seen in Fig. 10 and is also summarized in the last column of Table 3.

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.

Fact 29. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{191} GOST encryptions. The storage required is 2^{64} times 8 bytes.

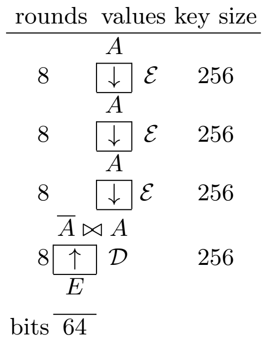


Fig. 10. The fixed point reduction with 2 KP 8 R and $P = 2^{-64}$

Justification: We combine Reduction 5 and Fact 5. The time to check all the false positive keys is included in the 2^{191} GOST encryptions (see Fact 15 or [50] for more details). The overall attack is summarized in the last column of Table 3.

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 17 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 [19], 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.

18.1 Related Research and History

In an the older version of this paper submitted to Asiacrypt 2011 the complexity of this attack was 2^{216}. A major breakthrough was an improved alternative final step for this attack which was first proposed in [50] and reduces the overall running time to about 2^{192}. For an independent method to achieve the same result, including variants with even slightly less time and slightly more memory, see Fact 15 in Section 12.4.

An analogous attack but with two fixed points for 8 rounds is the Family 5.2. attack given in Section 25.4. In Section 25.4 and I.4 we are going to re-visit fixed point attacks on GOST: we will discover the existence of double and multiple fixed points with special properties which do not exist for random permutations but do exist for 8 rounds of GOST and occur with surprisingly high probabilities. These and similar events lead to some of the best attacks on GOST ever found and many such attacks are described in this paper. They allow to decrease the total cost of recovering certain GOST keys generated at random in a truly spectacular way, down to 2^{130}, 2^{110}, and even 2^{101}, see Table 4 page 128. These attacks work in the most realistic scenario with a population of distinct random GOST keys. It works for some of these keys, while the keys for which the attack does not work can be discarded at a surprisingly low cost.

19 Discussion and Towards Better Attacks

19.1 Strong Self-Similarity, Low Complexity and Low Diffusion

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 17) 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 16 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 [58, 7, 8, 6] neither it exploits fixed points like in [20, 75]. In addition in Appendix A and Appendix B.1 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 [19, 48] 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^{110} GOST encryptions and only 3 KP, cf. Fact 6. 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 [16, 19, 25, 12, 13], significantly 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 [83, 46], and this also makes it vulnerable to Algebraic Cryptanalysis [12, 16, 25, 97, 98, 20, 46] and also to Meet-In-the Middle attacks [50], as well as to various combinations of the two, see Appendix 12.

It is worth noticing that until now we do not exploit any other property or weakness of GOST other than A) and B) above, and using only these two properties we are already able to construct some 50 very diverse and rather non-trivial attacks on GOST faster than brute force.

Remark. The power of the attacker will be substantially enhanced by considering multiple keys cf. Section 20.1 and internal self-similarities, the third interesting ingredient C) which will be introduced in Section 20.2 below.

19.2 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 [20], here the attacker ends up with two instances of a slightly different cipher with a shared key.

19.3 Beyond GOST

This paper is motivated by the idea of software/algebraic cryptanalysis (AC) and of self-similarity attacks which are disruptive techniques 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 [19] or GOST (this paper). Then, if the cipher has special properties at the high level, like in KeeLoq [20] 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, cf. Section 26, this type of attack could also work 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 [20] and elsewhere [58, 7, 8, 6], 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 there exists a final step for such an attack which is powerful enough to work below a certain threshold. We contend that the situation created by many self-similarity attacks with complexity reduction is quite unique. In the most basic form of slide attacks [58, 7, 8] (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 significantly 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 [20]. However in advanced self-similarity attacks like in this paper and in [20], 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 17.1, but probably by no means we could have 5, which would already require a number of assumptions which cannot be afforded by the attacker. In weak key attacks we can with a lot more effort obtain maybe 6 pairs cf. Section 27.3 but again not more.

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 various forms of software algebraic attacks (cf. also [48] for other low-data attacks). Then if the components of the cipher have low complexity, low diffusion and additional properties a software algebraic attack, could potentially become the best attack known on the given cipher. Recent tendency however is that such attacks may be improved or/and combined with MITM (Meet-In-The-Middle) attacks cf. Appendix 12, or/and conditioned by some differential properties of GOST cf. Section 25.

20 New Directions and New Methods

We could stop our paper here with a single key attack on GOST with complexity 2^{191} which is 2^{64} times faster than brute force. However we can achieve much more than this. We need to prepare to a major paradigm shift in understanding the security of ciphers such as GOST with low circuit complexity and a lot of self-similarity properties. Ciphers are NOT used with single keys. On the contrary. Almost every cipher is used with many different keys generated at random.

20.1 The Multiple Key Scenario

In the following sections we intend to demonstrate that this multiple key scenario is stronger more practical and much more versatile than the single key scenario. We will initially study several weak key classes. However we do **not** care about weak key attacks unless they can be transformed into "real" attacks which are able to recover ordinary GOST encryption keys generated at random.

20.2 New Dimension In Self Similarity Attacks

We also would like to produce families of weak key attacks which give the attacker an additional degree of freedom, so that we can in the future find many more such families which eventually could cover a much higher proportion of the key space than currently expected in this paper. This will be achieved in Section 25 and Section 26 where we will combine all of differential, complexity reduction, reflection, fixed point, MITM and software/algebraic approaches to GOST to find many new very efficient attacks with 256-bit keys generated at random. The third crucial ingredient in our work is:

C) the possibility of approximate self-similarity of several encryptions achieved through advanced [truncated] Differential Cryptanalysis (DC) attacks.
   In fact **DC is also an approximate self-similarity attack** in which we have identical state for most but not all bits inside different encryptions. In Section 26 we will study such properties also for multiple encryptions which quite surprisingly can happen for real-life GOST keys and leads to many interesting attacks.

These new attacks however require much more work. They require a modified and dedicated final step for each class of differential properties, see Section J.2 for an example of such dedicated attack. Moreover many of these attacks have been only optimized for one specific set of GOST S-boxes. We refer to [42, 44, 81] for a discussion of alternative S-boxes, while more than half of attacks described in this paper are expected to work more or less equally well for arbitrary S-boxes, it is much less trivial for differential attacks, see [39, 40, 95, 42, 44, 81]. On the other side, it is possible to see that these attacks allow the attacker considerable freedom in the choice of sets of active bits in advanced truncated differential properties. We can potentially have an exponential number of such differential sets, see Section for one simple example 25.6, leading possibly to even better attacks or to more complex attacks which combine many such properties.

20.3 Is It Interesting to Break GOST With The Full Code-Book?

A common misconception is to believe that attacks on a block cipher can only use less than 2^n of data, where n is the block size. Many people criticize attacks with 2^n of data as being very academic and not very practical. In fact such attacks are as realistic and important as many other cryptographic attacks. They are relevant in many realistic scenarios. The knowledge of the whole code-book does NOT replace the knowledge of the key. Some of the reasons for that are:

1. The key may be used to recover a master key.
2. In many cryptographic attacks, the exact spec of the cipher which is being broken or the S-boxes are not exactly known to the attacker.
3. To recover the key could be the only way to “prove” the authenticity of decryptions to a third party. An electronic proof of authenticity can be used to sell such data for a higher price.
4. A code book may be fake, contain errors or maliciously altered.
5. If just a few entries are missing, it will not affect our attacks. However these few entries may be much more valuable than the majority of entries.
6. In many cases where symmetric keys are used in the real life, they are used in order to somewhat to avoid having a large database (large code-book) to store or transmit. An interesting example are the scratch cards used by hundreds on millions of people every day. For any cipher with short blocks, plaintext / ciphertext pairs can be used as a form of electronic currency. Even if the attacker obtains nearly all the pairs already spent, only new “fresh” cryptograms can still be spent, and these can only be obtained through key recovery.
7. In Pay TV all pairs already sent to all the decoders can be considered as being public, only key recovery can allow to decrypt future content.
8. The paper [21] enumerates many more interesting scenarios where recovering the key given [nearly] the whole codebook is very valuable.

20.4 Data Complexity In Realistic Scenarios With Multiple Keys

In fact many attacks on GOST require more than 2^{64} of data obtained from encryptions with multiple keys. Such attacks are typically stronger more practical and much more versatile than the attacks with a single key, see numerous examples in Section 22.1, 23.2, 24.1, 24.6 and many more in Section 25 and 30.1. In these attacks the goal of the attacker is to recover some cryptographic keys given some limited computing power which is much higher than 2^{64} and therefore attacks with much more than 2^{64} of data make sense. The question of what is the best possible attack with $2^{1.5n} = 2^{96}$ of data makes as much sense as looking for the best attack with $2^{0.5n} = 2^{32}$ KP cf. Table 4 page 128.

In general for any cipher the security level decreases when the data complexity grows, and attacks with $D = 2^a$ of data which work for a proportion $d = 2^{-b}$ of keys and $T = 2^b$ computing power are hard to compare for different values of D, d, T. However there is one possible unified comparison metric.

A simple method to compare all such attacks is to consider the total overall computational cost defined as the total effort **per one key** recovered, even if we have to examine data from multiple encryptions which includes checking all the keys and breaking one of the weaker keys. The time complexity for one key recovered in the multiple random key scenario is a simple and yet very realistic method to compare all these attacks one a simple linear scale, and our key results are compared in Table 4 page 128.

Interestingly, even though the total quantity of data available in such attacks will be typically higher than 2^{64}, at the same time the data per device (for one distinct key) can be very low, frequently it is just 2^{32}. We are in the most general distributed attack scenario.

Part V

Security of GOST in the Multiple Random Key Scenario

21 Some Interesting Weak Key Attacks on GOST

In this paper we do NOT study weak keys in the sense in which a majority of papers in cryptography do. This would arguably be a waste of time. We study weak keys as a prelude to developing "regular" attacks in which keys are generated at random and weak keys can only appear with their natural probability. We are interested exclusively in the most realistic attacks scenario with multiple random keys, which is how encryption algorithms are used in practice.

21.1 Introduction To 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 [75]. 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 [75]. This type of weak-keys which are frequent enough to occur in the real life are worth studying. We can also add all weak key are worth studying if they allow us to decrease a total cost of finding one key, see Section 22.2.

Given the fact that even without weak keys, we have been able to find a dozen of 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 start by recalling the method of [75].

21.2 Weak Key Family 0

Fact 30 (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. Such additional fixed points are unlikely to be symmetric. Thus we obtain 1 P/C pair for 8 rounds of GOST $\mathcal{E}(A) = A$, which is correct with very high probability (we expect about $1 - 2^{-32}$).

21.3 Key Recovery With Family 0

Now in [75], this method of Fact 30 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 30). 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.

21.4 Weak Key Family 1 (Different Than Earlier Family 1)

Previous Versions: 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 31 (Weak Keys Family 1, $d = 2^{-32}$, Reduction to 1,2,3,4 KP for 8R). We define the Weak Keys Family 1 by keys such that 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 $d = 2^{-32}$ over the keys.

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 guess the symmetric value F and get 2 P/C pairs for 8 rounds correct with probability of about 2^{-34}.

We can also can guess C and with A, B, C we get 3 P/C pairs for 8 rounds correct with probability of about 2^{-66}.

Finally if we guess both B and F we get 4 P/C pairs for 8 rounds correct with probability of about 2^{-98}.

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

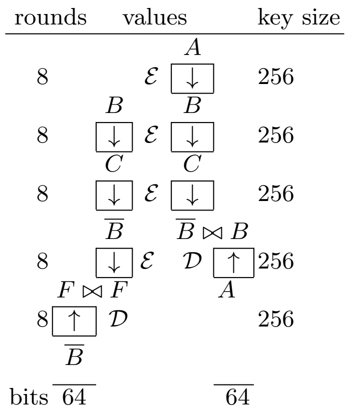


Fig. 11. Our Family 1 of Weak Keys

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 can then also guess F, which is not very hard because it is a symmetric value, and get 2 P/C pairs for 8 rounds $B = \mathcal{E}(A)$, $\mathcal{E}(\overline{B}) = F$ correct with probability of about 2^{-34}.
7. Going one step back, if we guess not F but C, we get 3 pairs $B = \mathcal{E}(A)$, $C = \mathcal{E}(B)$, $\overline{B} = \mathcal{E}(C)$ correct with higher probability about 2^{-66}.
8. We can also guess both C and the symmetric value F, which leads to 4KP for the price of being correct with probability of about 2^{-98}.

Now we need to examine the consequences of this reduction with 1,2,3,4 P/C pairs obtained. First we present one attack with 3 KP and then another with 2 KP which is better.

Fact 32 (One 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^{176} GOST encryptions and with negligible memory.

Justification: We use Fact 6 with the 3 P/C pairs obtained, and in total time equivalent to 2^{110} 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^{176} 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 cracking one weak key. However it is very important to examine what happens if some keys are weak and some are strong. For keys which are not weak, we need to run the attack and see that it fails most of the time. Overall for arbitrary 256-bit keys generated at random we would obtain one key in total time of about 2^{176+32} GOST encryptions per key effectively found. In order to obtain an arguably better attack than [50] 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 exhibit several such attacks (cf. later Table 4).

Now we describe a faster attack, with reduction to 2 KP.

Fact 33 (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^{161} GOST encryptions and with negligible memory.

Justification: Following Fact 31 we need about only 2^{34} attempts to guess A, F correctly and obtain 2 pairs for 8 rounds. We use Fact 5 with the 2 P/C pairs obtained, and in total time equivalent to 2^{127+34} GOST encryptions due to [50], we enumerate 2^{127+34} candidates for the full GOST key to be checked with additional pairs.

Now we can again look at a population of randomly generated keys. We have the following fact, anticipating what we later call "conversion" of a weak key attack into a "regular" attack.

Fact 34 (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^{193} GOST encryptions.

Justification: We apply the Fact 33 to 2^{32} random devices. For each device if the key is weak, we run the attack which takes 2^{161} GOST encryptions. Otherwise we still run it but it is going to fail. In one case on average, the attack will work and output a valid key which can be checked with additional pairs for that device. The total time is about 2^{161+32} GOST encryptions.

21.5 Weak Key Family 2.1

Recent changes 09/2012: We thank anonymous referee for pointing out that none of our Family 2 attacks requires 2^{64} of data, they all require only 2^{32} of data now.

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 17, and Section 17.1, 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 35 (Weak Keys Family 2.1, $d = 2^{-32}$, Getting 3,4 and 5 KP for 8R). We define the Weak Keys Family 2.1 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 2.1, we have the following reductions:
-with 2^{32} CC we obtain 3 P/C pairs for 8 rounds of GOST correct with $P = 2^{-64}$, (where CC means Chosen Ciphertexts)
-with 2^{32} ACC (Adaptive Chosen Ciphertexts) 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 $P = 2^{-96}$,
-in fact in a sense we get also 4.5 pairs, we have one additional pair where one value is not known but it is symmetric, also correct with $P = 2^{-96}$ and with 2^{32} CC,
-with 2^{32} ACC (Adaptive Chosen Ciphertexts) we obtain 5 P/C pairs for 8 rounds of GOST correct with $P = 2^{-96}$.

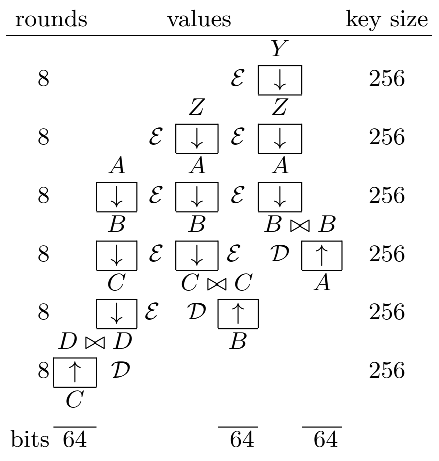


Fig. 12. Family 2.1: a triple reflection attack with up to 5 pairs for 8 rounds

Justification: It is easy to see that the event $\mathcal{E}(A)$ AND $\mathcal{E}^2(A)$ AND $\mathcal{E}^3(A)$ being symmetric leads to key density $d = 2^{64-32-32-32} = 2^{-32}$. The attack is shown in Fig. 12 where $\bowtie$ denotes $\mathcal{S}$ (swapping the two 32-bit halves). This can be seen as an extension of Fig. 9.

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 $A = \mathcal{E}^2(Y)$.

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 will need 2^{32} ACC: ciphertexts need to be adaptively decrypted. We do **not** need 2^{64} KP as claimed in earlier versions of this paper.

We proceed as follows:

1. We guess two symmetric values B, C. They are both correct with $P = 2^{-64}$.
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}. So far need 2^{32} Chosen Ciphertext (CC).
4. Furthermore, if we also decrypt A we get also one additional pair $\mathcal{E}(Y) = Z$, at the price of 2^{32} ACC (Adaptive Chosen Ciphertexts). Here though A is not symmetric we do NOT need 2^{64} KP. We can decrypt all possible A obtained from decryption of all possible symmetric values C. This requires 2^{32} ACC.
5. Going one step backwards, we don't decrypt A but also guess D which is symmetric, We get also 4 pairs $\mathcal{E}(Z) = A$, $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$, $\mathcal{E}(C) = D$ and our guess is correct with a degraded probability 2^{-96} however we need only 2^{32} CC.
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}. We need 2^{32} ACC (Adaptive Chosen Ciphertexts).

Out of four possibilities given in Fact 35, we will use two in order to obtain two new weak key attacks:

Fact 36 (Key Recovery for Family 2.1, 2^{32} CC, $d = 2^{-32}$).
One can recover the keys for the Weak Keys Family 2.1 with 2^{32} CC, running time of 2^{174} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the first reduction of Fact 35 and of Fact 6 which allows to enumerate a set of solutions and the time is 2^{64+110} 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^{174} 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 in the ACC scenario, we obtain:

Fact 37 (Faster Key Recovery for Family 2.1, 2^{32} ACC, $d = 2^{-32}$).
One can recover the keys for the Weak Keys Family 2.1 with 2^{32} ACC, running time of 2^{158} GOST encryptions and with negligible memory.

Justification: Here we replace 3 KP by 4 KP and Fact 6 with 2^{110} by Fact 7 with 2^{94}. We need a total time of $2^{64+94} = 2^{158}$ GOST encryptions.

22 Conversion of Weak Key Attacks Into Regular Attacks on Random 256-bit Keys

If we are dealing with the problem of key recovery of a **single** fixed 256-bit GOST key, then our best attack in Table 3 on page 53. requires 2^{64} known plaintexts has the time complexity of 2^{191} This can be further reduced to 2^{179} by a very different attack, see [39, 40].

However, in this paper we have many arguably much better attacks than (2^{191} time, 2^{64} data). In particular attacks with 2^{32} of data. This can be seen in our later Table 4 on page 128 which summarizes all our weak key and some other attacks. For many attacks we have that the ratio Time / d is less than 2^{191}. This actually **does mean** that GOST key can be recovered in overall total time of less than 2^{191}. which is substantially less than our best attack 2^{191} of Table 3 which can be achieved either by the method of [50] or using Fact 15.

One of these current attacks in less than 2^{190} will even be better than 2^{179} from [39, 40] on cost per key, another will require significantly less data, only 2^{32} instead of 2^{64} of data required in 2^{192} of [50] and 2^{179} from [39, 40]. Here is how this can be done, first we will look at Family 2.1 and obtain an attack in 2^{190}. Later in Section 23.2 we will re-examine Family 3 and get an attack in 2^{159}.

22.1 Conversion of Family 2.1 Into A "Regular" Attack In The Multiple Key Scenario

The following attack scenario is quite realistic: even today we can imagine 2^{32} devices with different GOST keys. We are going to convert the Weak Key Family 2.1 attack into a "regular" attack on multiple random 256-bit keys in the multiple key scenario.

Fact 38 (Conversion of Family 2.1 For Multiple Keys, $d = 2^{-32}$).
If we have a diverse population of at least 2^{32} different keys, with access to 2^{32} ACC 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 37 to each of the 2^{32} devices with random keys. We recover one key out of 2^{32} in total time of 2^{32+158} including the time to check all the other devices. In one case on average, the attack will work and output a valid key which can be checked with additional pairs for that device. For the other devices the attack fails and we abort it after some 2^{158} GOST encryptions.

Quite remarkably the data complexity is only 2^{32} per device.

Remark 1: In this "conversion" process, 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. It can be seen as a fair method of comparing various attacks on ciphers which are used by many different devices, with random diversified cryptographic keys.

Single Key Attacks: Though the attack above is not a single attack it is more realistic than the best known single key attacks on GOST [50, 39, 40], because it will actually recover one full 256-bit key with a total cost of 2^{190} GOST encryptions. Single key attacks are NOT at all realistic attacks on ciphers.

It almost never happens that a cipher is used with just one single key, there is usually some diversity. Our attack scenario is quite realistic: even today we can imagine 2^{32} devices with different GOST keys. Here we have weak keys which ARE frequent enough to occur in the real life and the are also individually CHEAP enough to break to worry about this attack overall, which is reflected by the total cost per key being 2^{190} GOST encryptions.

Related work on GOST with 2^{32} of data per device: Until now ALL known attacks on GOST with 2^{32} of data had always a cost of 2^{224} per key found, see Section 15.2 and [74] and [50]. In this paper for the first time ever we achieve 2^{190} GOST encryptions per key recovered, in a scenario with a realistic number of devices with different random 256-bit keys and with reasonable memory, better than any other known result.

In what follows we are going to develop numerous additional attacks with conversion. For example in Section 23.2 we will convert the Weak Key Family 3 and get the same sort of attack in total time of 2^{159} per key (!) for multiple regular keys generated at random.

22.2 Conversion of Weak Key Attacks Into Regular Attacks - FAQ

All this is quite interesting and requires some discussion which takes the form of Frequently Asked Questions.

1. *It is obvious that all short-cut attacks on weak keys can be transformed into a "regular" attack (i.e. with typical random keys).* – Correct.
   This is achieved by running the attack for all keys, weak and regular ones, and assuming that for regular keys the attack somewhat fails to recover the key, and if it doesn't, we simply time out at roughly the same cost.
   This is correct BUT most of the time this is a terribly bad attack with higher cost per key than other competing attacks. In this paper we see the opposite. It is a rare thing to see a weak key attack being strictly THE BEST known attack on the cipher in question simultaneously in two data complexity categories. This happens for example in Section 22.1 and 23.2 and at many other places.
2. *Let k be the key size. In my understanding, the brute force attack for weak keys with density 2^{-d} is 2^{k-d}.*
   No, this is wrong, or this may be wrong, for a typical cipher and for some family of weak keys, the brute force can still be still 2^k. The error of the reviewer is to consider that if we have 2^{k-d} keys in the given weak key class, than we can sample 2^{k-d} keys and check them. Unhappily weak keys are frequently defined by properties such that this space cannot be sampled efficiently.
   For example in our Family 2.1, cf. Fact 35, it is not easy to see if one can enumerate these 2^{256-32} keys in time of 2^{256-32}. This is probably impossible. Therefore by default, for individual weak keys, the brute force is still 2^k. Moreover to enumerate them all of them correctly (and be sure to distinguish the weak keys from those which are not weak) it could probably be even slower than 2^k.

This unless there exists a special shortcut attack in time T. This complexity T can be bigger or equal to 2^{k-d}, or smaller than 2^{k-d}:

(a) In the first case, transforming this key attack into a "regular" attack is slower than brute force. Still the attack is not uninteresting.
If the attacker does not know that a key used by somebody is a weak key, the attack is not very interesting because $2^d T \geq 2^k$.
However if the attacker knows from some source that this key is weak, then he will have an attack faster than brute force with time $T < 2^k$.

(b) In the second case $T < 2^{k-d}$ we obtain an undeniable advantage compare to brute force without any extra assumption or condition. Again we transform our attack into a "regular" attack: we run it also for stronger keys and abort at 2^{k-d} steps if needed. Overall total time is $2^d T < 2^k$.

3. Can a multiple key attack in 2^{159} be still slower than a single key attack in 2^{179}? *Yes if key diversity is limited.* Here is one example.
If we want just to recover one key with many different devices around, the attacks described in this paper have no match. However let's assume that we have the means to do 2^{188} computations. For this price we can recover 2^{29} keys just by the method of Section 23.2 or 2^{10} keys by the method of [39, 40]. However in the first case I need more devices, about 2^{29+32} devices with different random keys, while in the second case I need only 2^{10} devices to break. Therefore if the population of devices is limited and IF we want to recover many keys, a slower single key attack can be better than a faster multiple-key attack.

23 More Weak Key Attacks: Below 2^{128} Per Key

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}.

23.1 Weak Key Family 3

The following attack is very interesting because even though the weak key assumption is quite costly, it allows to obtain a lot of data inside the cipher at a very low price, essentially for free. This sort of event is quite unprecedented.

Fact 39 (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.

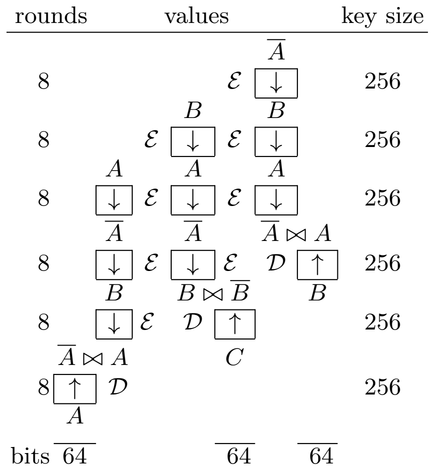


Fig. 13. Weak Key Family 3 which gives 4 pairs for 8 rounds

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 40 (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 39 and Fact 7 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}.

23.2 Conversion of Family 3 Into A "Regular" Attack with Random Multiple Keys With Cost of 2^{159} Per Key Found

Again we have the following conversion:

Fact 41 (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 40 to 2^{64} random devices. For each device with probability about half, if the key is weak, and Following Fact 39, 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: Though it is not a single attack it is more realistic than the best single key attacks on GOST [50, 39, 40], because it will actually recover one full 256-bit key with a total cost of 2^{159} per key.

24 Attacks With Symmetric Fixed Points

In 2012 another weak key attack was proposed by Kara and Karakoç, cf. [77]. This attack is quite interesting because it requires only 2^{32} CP which relatively few attacks achieve. It is even more remarkable when converted to a multiple key scenario. It is possible to see that the conversion can be done in a particularly efficient way leading to an attack with expected full cost of recovering one 256-bit key generated at random being as low as about 2^{130} [77]. In this paper we revisit this attack from [77], show that in some cases it does NOT work as predicted, and propose new variants of it.

24.1 New Weak Key Attacks with 2^{32} of Data: Family 4.1 and 4.2

These new attacks are based on the assumption that there are two symmetric fixed points in the first 16 rounds of GOST. Fixed points for the first 16 rounds are studied in Reduction 13 but here we assume in addition that both fixed points are symmetric.

Such symmetric fixed points happen only for weak keys and can happen for two distinct reasons. It is possible to see that the attack from [77] can be split into two independent weak key attacks which we will call Family 4.1 and Family 4.2 (the authors describe it as one single attack with a distinction of two distinct events Event1 and Event2, see [77]).

Fact 42 (Weak Keys Family 4.1 and 4.2 from [77] with $\mathbf{d \geq 2^{-64}}$). We define the Weak Keys Family 4.1 by keys such that there exists two **symmetric** $A \neq B$ such that $\mathcal{E}(A) = B$ and $\mathcal{E}(B) = A$. This occurs with probability at least $d \geq 2^{-65}$ over the GOST keys.
Likewise we define Family 4.2 by keys such that there exists **two symmetric** fixed points $A \neq B$ such that $\mathcal{E}(A) = A$ and $\mathcal{E}(B) = B$. This occurs also with key density of at least $d \geq 2^{-65}$.

Important Note: Very surprisingly it is possible to see that for 8 rounds of GOST and at least for the default set of GOST S-boxes (otherwise cf. [42]) both Family 4.1 and 4.2 events occur for a substantially larger proportion of keys (!). These additional events which are due to the internal structure of GOST and were not anticipated by the authors, the present attack of [77] fails. In order to clarify the situation we are going to introduce an additional assumption which will allow us to fix the attack of [77] and clarify the sitation. For this purpose we define the following notion:

Definition 24.1.1 (Unrelated Texts). We say that two points A, B on 64 bits are *unrelated* plaintexts for GOST if they lead to distinct values for all the eight S-boxes in the first round. Random pairs are *unrelated* with high probability.

Now we describe and revisit the attack of [77] which in this paper is divided in two distinct attacks each working for a different proportion of $d = 2^{-65}$ keys.

Fact 43 (Attack on Family 4.1 $d = 2^{-65}$, 2 CP for 8R). For every key in Family 4, and given 2^{32} CP we obtain 2 P/C pairs for 8 rounds of GOST correct with probability close to 1.

Then if the texts A, B in these pairs are *unrelated* with Fact 5 given 2^{32} CP for the full GOST we can break these weak keys in time of 2^{127}.

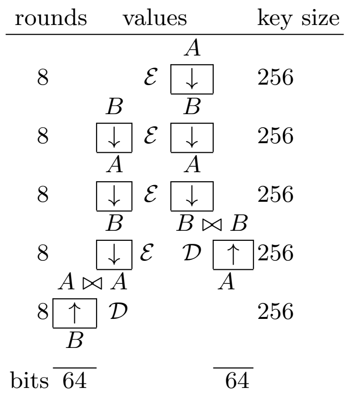


Fig. 14. Family 4.1 of Weak Keys by Kara and Karakoç, $d = 2^{-65}$ [77].

Justification: We have $\mathcal{E}^3(A) = B$ and $\mathcal{E}^3(B) = A$. Both values are symmetric therefore $Enc_k(A) = \mathcal{E}^2(A) = A$ and the same for B. We have two distinct symmetric fixed points of $Enc_k()$. Other symmetric fixed points of $Enc_k()$ are not very likely to exist. The attacker only needs encryptions of all symmetric plaintexts which gives 2^{32} CP. We obtain 2 KP for 8 rounds which are $\mathcal{E}(A) = B$ and $\mathcal{E}(B) = A$. If the texts A, B in these pairs are *unrelated* we apply Fact 5 which gives the time complexity of 2^{127}.

Important Note: Unhappily it is possible to show that this attack of [77] can **NOT** work when A, B share many bits (they are related). This at least for the default set of GOST S-boxes (otherwise cf. [42]). This is due to the fact that the last step in this attack must either be the attack of [50] or our alternative to it: Fact 5 based on Fact 15 page 40. It is easy to see that in both cases when A, B share many bits the number of solutions to our system with 2 KP for 8 R which need to be checked with additional data for the full 32-round GOST, is going to increase substantially, making it impossible to enumerate these solutions in time 2^{128} GOST computations as it is necessary to do in [77].

There are two ways to solve this problem. First we will restrict to cases with *unrelated* A, B. Later in Section 27.1 we will design a different attack which will benefit from the fact that these pairs tend to be related.

24.2 Second Attack Variant From [77]

Let us see the second method from [77] which is even simpler and it is about the event studied the earlier Kara attack from [75] (a.k.a. Family 0 in this paper) occurring twice leading to 2 pairs for 8 R instead of 1.

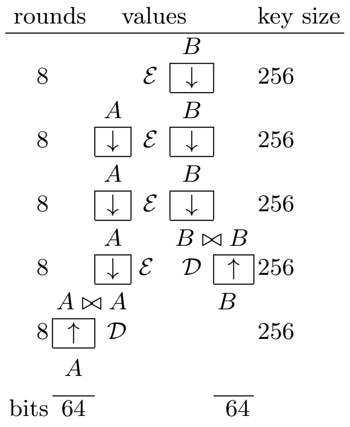


Fig. 15. Family 4.2 of Weak Keys = Family 0 Happens Twice

Fact 44 (Attack on Family 4.2 $d = 2^{-65}$, 2 CP for 8R). For every key in Family 4.2, and given 2^{32} CP we obtain 2 P/C pairs for 8 rounds of GOST correct with probability close to 1.
Then again only if the texts A, B in these pairs are *unrelated* with Fact 5 given 2^{32} CP for the full GOST we can break these weak keys in time of 2^{127}.

Justification: Again we have $\mathcal{E}^3(A) = A$ and $\mathcal{E}^3(B) = A$ and the symmetric fixed points also work for the whole $Enc_k()$. Other symmetric fixed points of $Enc_k()$ are not very likely to exist. The attacker only needs encryptions of all symmetric plaintexts which gives 2^{32} CP. We obtain 2 KP for 8 rounds which are $\mathcal{E}(A) = A$ and $\mathcal{E}(B) = B$. Again if A, B *unrelated* we can apply Fact 5 and we obtain 2^{127} GOST encryptions.

Further work on this attack is not possible before we clarify the probability that events such as defining Family 4.1 and 4.2 occur for the real life GOST cipher.

24.3 On Frequency of Symmetric Fixed Points in GOST

Fact 45 (Symmetric Fixed Points in GOST). For 8 rounds of GOST and a random key, the probability that there is a symmetric fixed point is about 2^{-33} and is expected to be the same for a random permutation.
However for the default set of GOST S-boxes the probability that there are two symmetric fixed points sharing as many as 50 bits which will be the inactive bits

in mask $0x8070070080700700$ as illustrated on later Fig 20 page 98, is at least 2^{-60} instead of about $2^{-2-64-50}$ for a random permutation.

Fact 46 (Frequency of Weak Keys Family 4.1 and 4.2). Consequently at least for the default set of GOST S-boxes, weak keys in Family 4.2 occur with key density of at least $d \geq 2^{-60}$ over GOST keys. Likewise weak keys in Family 4.1 occur for these S-boxes with key density of at least $d \geq 2^{-60}$ over GOST keys.

Justification: Let $A' = \mathcal{E}(A)$ and $B' = \mathcal{E}(B)$. Following [38] 8 rounds of GOST there are 2^{77} pairs with the input difference of type $0x8070070080700700$, and for a proportion of 2^{-25} of them, which is 2^{52} pairs A, B on average, the output difference is also in $0x8070070080700700$. This for the default set of GOST S-boxes. Now our simulations show that when the propagation occurs the entropy of $A \oplus B$ is low and the probability that $A \oplus B = A' \oplus B'$ is only about 2^{-9}. Furthermore $A = A'$ with probability 2^{-64} which also implies $B = B'$. Furthermore A is symmetric with probability 2^{-32}. Then since their difference lies within $0x8070070080700700$ which is symmetric, B is symmetric with probability at most 2^{-7}. Overall for a proportion of $d = 2^{52-9-64-32-7} = 2^{-60}$ of GOST keys we have two symmetric fixed points A, B sharing the same 50 bits. For Family 4.1, in the middle of our argument instead of assuming that $A = A'$ which holds with probability 2^{-64} and also implies $B = B'$, we need to assume $A = B'$ which holds with the same probability and implies $A' = B$.

Remark 1. This is higher than what we would get for having just two symmetric fixed points without any extra condition which is roughly about 2^{-2-64} for a random permutation.

Remark 2. So far this reduction in probability was only demonstrated for the default set of GOST S-boxes. However, this is a conservative estimate with just one mask $0x8070070080700700$. These probabilities need to be added for different masks and in the real life this probability should be even higher. We also expect similar results for other sets of S-boxes. We refer to [42, 44] for more research on alternative S-boxes. It is not correct to claim that this sort of property will not exist for other S-boxes [95], see [42, 44]. It may be weaker for other S-boxes.

Again symmetric fixed points for 8 rounds are also symmetric fixed points for 32 rounds and we obtain a distinguisher attack on full GOST:

24.4 Distinguisher Attacks on Full GOST

Fact 47 (Two Distinguisher Attacks on Full GOST). This distinguisher improves on the result of [77] and works is we have access to many instances of GOST with different keys. For the default set of GOST S-boxes the probability that there are two symmetric fixed points is at least 2^{-60} instead of about 2^{-65} for a random permutation. Moreover the probability that there are two symmetric fixed points sharing as many as 50 bits in $0x8070070080700700$ is at least 2^{-60} instead of about 2^{-65-50} for a random permutation.

24.5 Family 4.1 and 4.2 in the Multiple Key Scenario

As in Section 22.1 and elsewhere, we extend these two attacks to the multiple key scenario.

Fact 48 (Family 4.1 and 4.2 in the Multiple Key Scenario). If we have a diverse population of at least 2^{64} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in total overall time of at least 2^{133} GOST encryptions for the default set of GOST S-boxes. For other S-boxes the complexity is expected to be also about 2^{129} GOST encryptions.

Justification: This attack requires that some 2^{64} devices with random keys exist. It is possible to see that for a proportion of about 2^{-65} keys, A, B are expected to be *unrelated* and our Family 4.1 attack works in time 2^{127}. For another proportion of about 2^{-65} keys, A, B are also *unrelated* and our Family 4.2 attack works in time 2^{127}. Overall for a proportion of about 2^{-64} keys, one of these attacks will work. In each case we need to try both the attack of Fact 43 and the attack of Fact 44 with $2^{127} + 2^{127} = 2^{128}$ GOST computations total.

As in many other attacks such as Fact 48 a remarkable thing happens with keys which are not weak. They can be rejected at a low cost and we do not have to run the whole attack for a majority of the keys. This is true because for a random permutation, the probability that $Enc_k()$ has two symmetric fixed points A, B is low, however it is NOT as low as we initially thought. There is no need to run the attack 2^{64} times. However we need to run the attack at least 2^{64-59} times because Family 4.1 or 4.2 implies that the whole 32-round GOST has two symmetric fixed points and this happens with probability about $2^{-60} + 2^{-60}$ following Fact 46. This at least for the default set of GOST S-boxes. Therefore we need to run the whole attack at least 2^{64-59} times, **and potentially more**.

For all except a proportion of about 2^{64-59} GOST keys, we can reject them right away. We have 2^{64} different keys and we check them for two symmetric fixed points in time of about 2^{32} each, which will be very small compared to the dominant term in the attack. With 2^{64} different keys, two symmetric fixed points will occur only a few times and it is possible to see that, Overall the complexity of this attack is at least $2^{128+64-59} = 2^{133}$ GOST computations to recover one key, and possibly more, this is if the probabilities in Fact **??** are higher. In Section 27.1 we are going to show how to benefit from the fact that GOST has more symmetric fixed points than expected, which is a problem in this attack.

For other sets than the default set of GOST S-boxes we are not aware of a similar reduction in probability of having two symmetric fixed points, it is expected to be the same as for a random permutation. It is in general not correct to claim that this fails for other S-boxes [95], see [42, 44, 81].

If this works as claimed for a given set of S-boxes, then the analysis of [77] applies. We expect that $Enc_k()$ has two symmetric fixed points with frequency of about 5/2 the frequency that either Family 4.1 or Family 4.2 event occurs. We expect to run our attack maybe twice and get the complexity of at least $2^{128+1} = 2^{129}$ GOST computations.

24.6 Another Weak Key Attack With More Data

Let us see yet another reduction with fixed points in the first 16 rounds. In Family 4.1 and 4.2 we have two different reasons why the first 16 rounds of GOST are going to have two fixed points and these points are symmetric. An anonymous reviewer has once suggested yet another method to obtain two fixed points for the first 16 rounds as follows. In this method fixed points are not symmetric but they are related to each other and their difference is still symmetric. Attacks based on this method require more data than attacks based on Family 4.1/4.2 keys but will work for keys for which other attacks fail.

Fact 49 (Weak Keys Family 4.3, $d = 2^{-64}$, 2 KP for 8R).
We define the Weak Keys Family 4.3 by keys such that there exists a point A such that $\mathcal{E}(A) = \overline{A}$ and $\mathcal{E}(\overline{A}) = A$. This occurs with density $d = 2^{-64}$.

For every key in Family 4.3, and given 2^{64} CP we obtain 2 P/C pairs for 8 rounds of GOST correct with probability close to 1.

Thus given 2^{32} CP for the full GOST we recover A and $\overline{A}$ for free (because they satisfy both $Enc_k(\overline{A}) = A$ and $Enc_k(A) = \overline{A}$ which is unlikely to happen by accident, cf. also Family 4.4. described later).

Then we can produce 2^{128} candidates for a weak key in Family 4.3 in time of about 2^{127} GOST encryptions with Fact 5. These keys are checked with additional data for full GOST and we find the right key in time of roughly about 2^{127} GOST encryptions overall.

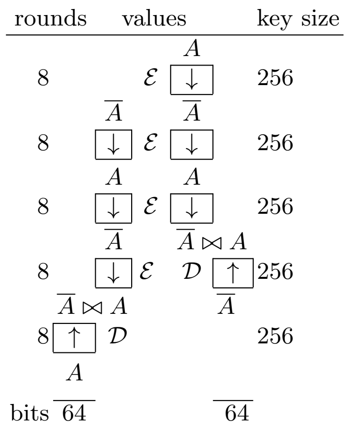

rounds
values
key size
A
8
$\mathcal{E}$ $\boxed{\downarrow}$
256
$\overline{A}$ $\overline{A}$
8
$\boxed{\downarrow}$ $\mathcal{E}$ $\boxed{\downarrow}$
256
A A
8
$\boxed{\downarrow}$ $\mathcal{E}$ $\boxed{\downarrow}$
256
$\overline{A}$ $\overline{A} \bowtie A$
8
$\boxed{\downarrow}$ $\mathcal{E}$ $\mathcal{D}$ $\boxed{\uparrow}$
256
$\overline{A} \bowtie A$ $\overline{A}$
8 $\boxed{\uparrow}$ $\mathcal{D}$
256
A
bits 64
64

Fig. 16. Family 4.3 of Weak Keys

Now we are going to extend the Family 4.3 attacks to the multiple key scenario.

Fact 50 (Family 4.3 in the Multiple Key Scenario). Given a population of 2^{64} different random keys, with access to 2^{64} KP per key, we one can recover **one** of these 256-bit keys in total overall time of about 2^{129} GOST encryptions.

Justification: With Fact 49 above, we can break weak keys of Family 4.3 in time of 2^{127} for each key. As in many other attacks such as Fact 48 it is easy to reject keys which are not weak. This is because for a random permutation, the probability that $Enc_k()$ has a point A such that $Enc_k(\overline{A}) = A$ and $Enc_k(A) = \overline{A}$ is about 2^{-64}. For all except a proportion of about 2^{-64} GOST keys, we can reject them right away in time of 2^{64} each. For the remaining 2^{-64} for roughly a proportion of 2^{-2} of them they come from Family 4.3 which is a conservative estimate (other such cases which are not like in Fig. 16 for example a later Family 4.4 can occur by accident with probability of about 2^{-64}).

Thus we only need to run our attack maybe twice in order to recover a correct key in one case. Overall we expect to recover the key in time of about 2^{129} GOST encryptions though it might still be a slightly optimistic estimation.

Discussion: In this case, do we do NOT expect problems which make the complexity in Fact 46 a bit higher than initially expected, some 2^{133} instead of 2^{129}, but we might be wrong.

24.7 Yet Another Similar Key Attack With More Data

Let us see yet another different yet similar reduction with fixed points in the first 16 rounds which points are going to be related.

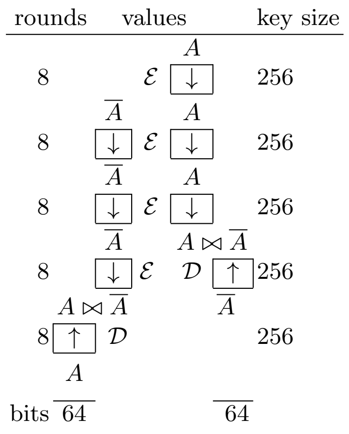


Fig. 17. Family 4.4 of Weak Keys

Fact 51 (Weak Keys Family 4.4, $d = 2^{-64}$, 2 KP for 8R).

We define the Weak Keys Family 4.4 by keys such that there exists a point A such that $\mathcal{E}(A) = A$ and $\mathcal{E}(\overline{A}) = \overline{A}$. This occurs with density $d = 2^{-64}$.

For every key in Family 4.4, and given 2^{64} CP we obtain 2 P/C pairs for 8 rounds of GOST correct with probability close to 1.

Thus given 2^{32} CP for the full GOST we recover A and $\overline{A}$ for free because they satisfy both $Enc_k(\overline{A}) = A$ and $Enc_k(A) = \overline{A}$. This is unlikely to happen by accident, though it could also come from earlier Family 4.3.

Then we can produce 2^{128} candidates for a weak key in Family 4.4 in time of about 2^{127} GOST encryptions with Fact 5. These keys are checked with additional data for full GOST and we find the right key in time of roughly about 2^{127} GOST encryptions overall.

Now we are going to extend the Family 4.4 attacks to the multiple key scenario.

Fact 52 (Family 4.4 in the Multiple Key Scenario). Given a population of 2^{64} different random keys, with access to 2^{64} KP per key, we one can recover **one** of these 256-bit keys in total overall time of about 2^{129} GOST encryptions.

Justification: With Fact 51 above, we can break weak keys of Family 4.4 in time of 2^{127} for each key. Again as in many other attacks such as in Family 4.3 and earlier it is easy to reject keys which are not weak. This is because for a random permutation, the probability that $Enc_k()$ has a point A such that $Enc_k(\overline{A}) = A$ and $Enc_k(A) = \overline{A}$ is about 2^{-64}. For all except a proportion of about 2^{-64} GOST keys, we can reject them right away in time of 2^{64} each. For the remaining 2^{-64} for roughly a proportion of 2^{-2} of them they come from Family 4.4. Other such cases which are not like in Fig. 17 for example coming from earlier Family 4.3 can occur by accident with probability of about 2^{-64}).

Thus we only need to run our attack maybe twice in order to recover a correct key in one case. Overall we expect to recover the key in time of about 2^{129} GOST encryptions though it might still be a slightly optimistic estimation.

Part VI

Advanced Combined Attacks

25 Complexity Reduction And Differential Cryptanalysis

Most of the work in this paper and all the work until now are about black box reductions: for a long time we ignore what is inside 8 round of the GOST cipher and develop a variety of attacks based on high-level self-similarity of GOST. In this section and in many other attacks which follow, we will exploit the low-level self-similarity of GOST in the form of sets of differentials such as studied in [100, 36–39]. This will be done with the same key objective which is the core of this paper: find interesting ways to obtain 2,3,4 pairs for 8 rounds of GOST with various parameters. Differential properties will be used NOT for solving of systems of equations which represent 8 rounds of GOST: not yet and not here (even though they do very seriously affect the solving process, see for example later Appendix J.1). Instead they will be used for our black-box reductions: they are going to modify in a very substantial way the probabilities that certain configurations can happen, and the probabilities that certain quantities can be guessed by the attacker.

We will be using massively the fact that 8 rounds of GOST are NOT a random permutation yet for a long time we will still be considering it as a block box and work at the high level, on complex I/O relations with inputs and outputs of several (related) instances of 8 rounds of GOST with the same key, and on how these blocks can be connected together to form interesting relations on full 32-round GOST leading to efficient attacks.

First we are going te establish some useful properties of GOST and in particular differential properties with well chosen additional constraints.

25.1 One Basic Differential Property For 8 Rounds of GOST

We consider the following standard situation which has been introduced [36] and further studied in [36–40]. Let $\varDelta = 0x80700700$ which mask has 7 active bits out of 32. We denote by $(\varDelta, \varDelta)$ a set of $2^{14} - 1$ non-zero differences on 64 bits with up to 14 active bits.

Fact 53. For 8 rounds of GOST we have about 2^{51} pairs which satisfy the differences $(\varDelta, \varDelta)$ at both ends AND in the middle (cf. Fig 18).

For a random permutation, there are 2^{77} pairs with the input difference $(\varDelta, \varDelta)$, and for a proportion of 2^{-50} or 2^{27} pairs on average, the output difference is also in $(\varDelta, \varDelta)$.

```
0x80700700 0x80700700
      (4 Rounds)
0x80700700 0x80700700
      (4 Rounds)
0x80700700 0x80700700
```

Fig. 18. Basic Event With 3x14 active Bits For 8 Rounds

Justification: We do computer simulations similar as in [38, 40]. For the middle pairs with difference $(\varDelta, \varDelta)$ there are still 2^{77} possible pairs. Following [38] this propagates backwards for 4 rounds and gives the input difference (Δ, Δ) with probability $2^{-13.6}$ and forwards with another $2^{-13.6}$. Overall we predict $2^{77-13.6-13.6} = 2^{49.8}$ pairs which survive if the 2 propagations are independent. We have also tested this propagation for 8 rounds and our simulations show that the 2 propagations are NOT independent and this is in fact in our advantage: there are not $2^{77-27.2} = 2^{49.8}$ but $2^{77-26} = 2^{51}$ pairs on average over the key, which satisfy ALL the differences in Fig 18.

25.2 First Example: Simple Attack With Single Reflection $d = 2^{-43}$

This attack is the straightforward and the simplest possible extension of the basic attack with single reflection of Section 15.1 which also studied in [75, 50]. This attack is not very good but it is one of the simplest attacks which illustrates very well how almost any of our attacks can be combined with additional assumptions about the data inside these attacks, and the difficulties which we may have in order to design such attacks: the necessity to be able to compute various probabilities exactly. It also shows the fact that some of these probabilities are going to be surprisingly low.

This example can be skipped by the reader and one can read directly the next attack which is designed from scratch.

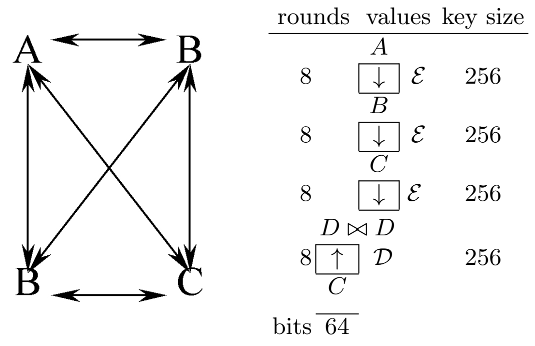


Fig. 19. Family 5.1 - a simple reflection attack with 2 or 3 KP for 8 R obtained and with shared 50 bits for each pair.

Fact 54 (Weak Keys Family 5.1, $d = 2^{-43}$). We define the Weak Keys Family 5.1 by keys such that there exists two points A, B such that $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$ and all these 3 points lie in the same space with 50 shared bits, with $A \oplus B \in 0x8070070080700700$ AND with $A \oplus C \in 0x8070070080700700$ AND such that $D = \mathcal{E}(C)$ is symmetric. cf. Fig 19. This occurs with probability $d = 2^{-43}$ over the key.

Justification: Given A, the points B, C, D are determined uniquely. The probability that A and B have the same 50 bits, i.e. $A \oplus B \in 0x8070070080700700$ is 2^{-50} and 2^{14} points A survive. Then the probability that after 8 rounds we have also $B \oplus C \in 0x8070070080700700$ is probably roughly about 2^{-25} following again [36–39] though it might be different due to the additional constraints.

Thus $2^{14-25} = 2^{-11}$ points A survive. Finally the probability that D is symmetric is 2^{-32}. Overall $2^{-11-32} = 2^{-43}$ points A survive on average for each key. These probability estimations are very rough and might be different due to various dependencies.

Now we are going to explain how to recover such keys in Family 5.1.

Fact 55 (Key Recovery For Family 5.1). Given 2^{64} KP we can obtain 3 KP for 8 rounds which are correct with probability 2^{-48} and accordingly we can recover a weak key in Family 5.1 in time of 2^{158} GOST encryptions and with small memory.

Justification: We guess A and check C for the condition $A \oplus B \in 0x8070070080700700$. Only 2^{14} pairs are consistent. Thus the pair A, C can be guessed with probability 2^{-14}. We claim that B has very low entropy, maybe it can be guessed with probability 2^{-2}. This is because both values A and C are known, and both differences $A \oplus B \in 0x8070070080700700$ and $B \oplus C \in 0x8070070080700700$ give low entropy for B, about 8 bits as shown by computer simulations, and B must lie in the intersection of these two highly biased distributions. This special distribution needs to be studied in more details.

Finally we guess D which is symmetric. Thus the quadruples A, B, C, D can be guessed with low probability maybe about $2^{-14-2-32} = 2^{-48}$. We have obtained 3 pairs for 8 rounds of GOST which are $A \mapsto B$, $B \mapsto C$, $C \mapsto D$. Then we need to apply a dedicated version of our attack of Fact 6 but with additional differences. We conjecture that it should take time of 2^{110} GOST encryptions in the same way as in Fact 6. This has not been demonstrated at this moment however in Fact 124 described in Appendix J.1 we present a different version of it in which we also obtain 2^{110}. Overall we need $2^{48+110} = 2^{158}$ GOST encryptions to break keys of Family 5.1.

Fact 56 (Family 5.1 in the Multiple Key Scenario). Given a population of 2^{48} different random 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^{201} GOST encryptions.

Justification: With Fact 55 above, we can break weak keys of Family 5.1 in time of 2^{158} for each key and 2^{64} KP per key. We need to repeat this attack 2^{48} times, for both weak and strong keys.

25.3 Double, Related And Approximate Fixed Points

We are also interested in some more peculiar differential sets which are designed for specific attacks described in this paper and have never been studied before.

The interesting question is can we exploit the periodicity of GOST key schedule in such a way that differential propagation will be achieved not through differentials but through periodicity. The question is if a pair A, B propagates as in Fig. 18, what is the probability that A', B' lie in the same affine space of dimension 50 as A, B. This opens interesting questions such as whether we could have $(A', B') = (A, B)$ or $(A', B') = (B, A)$ or similar and what is the probability these occur. One of these possibilities is illustrated in Fig. 20.

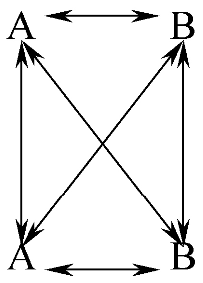


Fig. 20. Two related fixed points for 8 rounds

In general we do NOT always want to have fixed points, but only some relations between A, B and their encryptions after 8 rounds A', B'. Our basic differential propagation result of Fact 53 involves 2 points and with some additional effort the affine space at the input may be the same as at the output. We get a situation as in Fig. 21.

This is closely related and can be seen as a special nearly degenerated case of an interesting general cryptanalytic concept of a "approximate fixed point biclique" configuration which we will introduce later in all generality, see Definition 26.0.1 page 104.

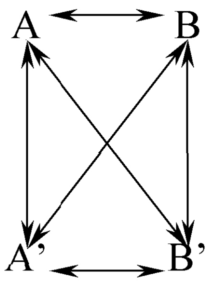


Fig. 21. An approximate fixed point biclique with $k = 2$, cf. Definition 26.0.1 page 104.

Here we have only 2 encryptions. Each pair on the picture shares the same 50 identical bits and all the 4 points lie in the same affine space with 2^{50} elements. We have "approximate" fixed points which differ by only up to 14 bits.

One of the very interesting properties of such configurations is the low entropy of A, B, A' etc. and even lower entropy for some other quantities such as $A \oplus B$ or for conditional entropies.

Fact 57 (Double Fixed Points and Swaps for 8 Rounds). For the default set of GOST S-boxes the probability that there exists two fixed points A, B for 8 rounds is about $(1 - 2/e) \approx 0.26$ and is expected to be the same for a random permutation.

However the probability that there exists two fixed points A, B sharing 50 bits in the precise sense of $A \oplus B \in 0x8070070080700700$ is about 2^{-20} instead of about 2^{-52} for a random permutation.

Moreover the probability that there exists two points A, B sharing 50 bits ac above which are either fixed two fixed points or a swap, i.e. $B = \mathcal{E}(A)$, $A = \mathcal{E}(B)$ OR $A = \mathcal{E}(A)$, $B = \mathcal{E}(B)$, is at least $d = 2^{-19}$ over the key.

Justification: We have 2^{77} with suitable input differences and they propagate with probability $2^{-25.0}$, cf. [37, 38]. this gives 2^{52} pairs which survive. Furthermore the probability that $A' = A$ OR $B' = A$ is 2^{-63}. Furthermore we have $A \oplus B = A' \oplus B'$ with probability 2^{-9} due to low entropy of the differences when the propagation occurs. Overall we obtain $2^{52-63-9} = 2^{-20}$.

25.4 Basic Attack With Periodic Differentials $d = 2^{-21}$

In this section and in later Family 5.3-5.4 attacks we are going to generalize the Fixed Point attack in 2^{191} from Section 18 (a variant with complexity 2^{192} is also described in [50]), which remains the fastest single key attack in this paper though it requires 2^{64} of data. We are going to show that there are attacks with multiple fixed points which are faster and more realistic than a single key attack with $T = 2^{191}$ and 2^{64} of data.

We observe that if a pair A, B propagates as in Fig. 18, in addition it will give exactly the same pair A, B with probability at most 2^{-77} and in fact about $2^{-64-9} = 2^{-73}$. This is due to the fact which we have observed experimentally that if the differential propagates the entropy of $A \oplus B$ is not 14 bits but about 9 bits, this independently of the key. We have 2^{51} pairs such as in Fig. 18 and for a random GOST key the probability that one is periodic with A, B being mapped to A, B respectively is about $2^{-73+51} = 2^{-22}$. This is a special case of related fixed points for 8 rounds of GOST, see Fact 123 in Appendix I.4 for the general case and variants which agree on a smaller set of bits. Here we limit to the case of pairs with 50 bits fixed. We have the following interesting family of keys:

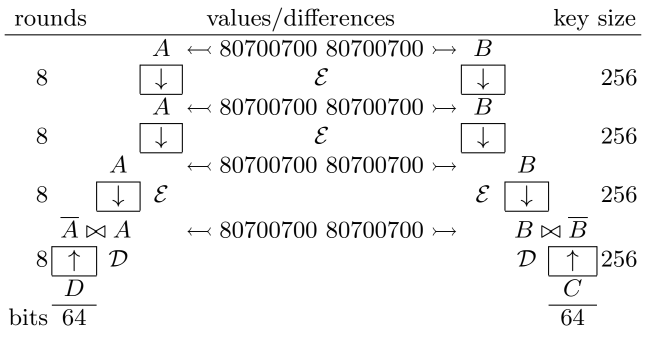

rounds
values/differences
key size
A ⟵ 80700700 80700700 ⟶ B
8
↓
ℰ
↓
256
A ⟵ 80700700 80700700 ⟶ B
8
↓
ℰ
↓
256
A ⟵ 80700700 80700700 ⟶ B
8
↓ ℰ
ℰ ↓
256
Ā ⋈ A ⟵ 80700700 80700700 ⟶ B ⋈ B̄
8
↑ 𝒟
𝒟 ↑
256
D
C
bits 64
64

Fig. 22. A Differential-Double-Fixed-Point Attack on GOST

Fact 58 (Weak Keys Family 5.2, $d = 2^{-21}$). We define the Weak Keys Family 5.2 (it was Family 9 in earlier versions of this paper) by keys such that there exists two points A, B such that $B = \mathcal{E}(A)$, $A = \mathcal{E}(B)$ OR $A = \mathcal{E}(A)$,

$B = \mathcal{E}(B)$, and such that the pair A, B and their encryptions after 8 rounds have the differences with 14 active bits depicted in Fig. 18. The first of the two cases is shown in Fig. 22. This occurs with probability $d = 2^{-21}$ over the key.

Justification: We have 2^{52} pairs such as in Fig. 18 and for a random GOST key the probability that one is periodic with A, B being mapped to A, B respectively is about $2^{-72+51} = 2^{-22}$ as above. Furthermore the probability to map A, B to B, A in Fig. 18 is also $2^{-72+51} = 2^{-22}$. The sum is $d = 2^{-21}$. This is half of $d = 2^{-20}$ obtained in Fact 57 because here we assume additional differences after 4 rounds (cf. Fig. 18).

Now we are going to show how to break these weak keys.

Fact 59 (Key Recovery For Family 5.2). Given 2^{64} KP we can obtain 4 KP for 8 rounds which is correct with probability 2^{-73} and accordingly we can recover a weak key in Family 5.2 in time of 2^{167} GOST encryptions and with small memory.

Justification: We guess A, B and our guess will be true with probability not $2^{64+14-1}$ as the reader might expect but about $2^{-64-9} = 2^{-73}$ as explained above due to the low entropy. Then we determine C, D from the encryption data. Overall we can obtain A, B, C, D which is correct with probability 2^{-73}. In this attack by definition we always define $C = Enc_k(B)$ and $D = Enc_k(A)$. We have obtained 4 pairs for 8 rounds of GOST which are either $A \mapsto A$, $B \mapsto B$, $C \mapsto \overline{B}$, $D \mapsto \overline{A}$, or in the other case we have included in the definition of Family 5.2, where A, B are swapped, which is not the case shown in Fig. 22, we have the following 4 pairs for 8 rounds: $A \mapsto B$, $B \mapsto A$, $D \mapsto \overline{B}$, $C \mapsto \overline{A}$.

Now we might be tempted to apply Fact 7. With 4 KP it allows to recover the key in time of 2^{94} GOST encryptions. However this is **not** going to work. Two out of four pairs share 50 plaintext and ciphertext bits and many internal bits. This means that individual key are less likely to be rejected with such data and there is many more correct keys than 2^{64} which is the case with unrelated encryptions in Fact 6. However, even if we have more solutions, it is very easy to see that each solution is easier to find: this is because two out of the three encryptions share many internal bits with higher probability, there is basically less variables or they have smaller joint entropy.

So we need a more precise dedicated attack to see what is the best attack in this case. In Appendix J.2 we show that with two couples related by differences it can be done in 2^{94} GOST encryptions in the same way as with Fact 7. In this case we have a situation between these two cases, and we conjecture that it can be done by a similar dedicated method with 2^{94} GOST encryptions. Overall we need $2^{94+73} = 2^{167}$ GOST encryptions.

Fact 60 (Family 5.2 in the Multiple Key Scenario). Given a population of 2^{20} different random 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^{188} GOST encryptions.

Justification: With Fact 59 above, we can break weak keys of Family 5.2 in time of 2^{167} for each key and 2^{64} KP per key. We need to repeat this attack 2^{21} times, for both weak and strong keys.

Remark In these attacks we use differential properties. Therefore the complexity of these attacks depends on the set S-boxes used and the attack requires a very careful adaptation and optimisation for each set of S-boxes. No systematic method to do this is known, and the degree of freedom for the attacker is vast, see [36–38, 40]. We refer to [42, 44, 41, 81] for some results and discussion on alternative S-boxes.

25.5 Second Attack With Periodic Differentials $d = 2^{-51}$

This attack is based on the previous attack but we are going to assume that in addition one of the points A, B is symmetric. This will decrease the number of weak keys however it will provide very substantial gains in later steps of the attack and arguably overall better attacks, though working only if the key diversity is large enough.

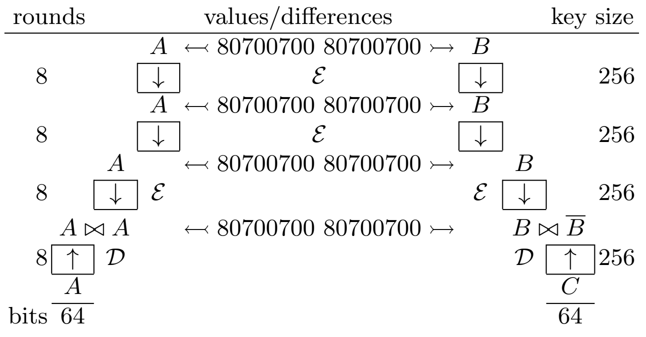

rounds
values/differences
key size
A ⟻ 80700700 80700700 ⟼ B
8
ℰ
256
A ⟻ 80700700 80700700 ⟼ B
8
ℰ
256
A ⟻ 80700700 80700700 ⟼ B
8
ℰ
ℰ
256
A ⋈ A ⟻ 80700700 80700700 ⟼ B ⋈ B̄
8
𝒟
𝒟
256
A
C
bits 64
64

Fig. 23. A Differential-One-Symmetric-Two-Fixed-Points Attack on GOST

Fact 61 (Weak Keys Family 5.3, $d = 2^{-52}$). We define the Weak Keys Family 5.3 (it was Family 10 in earlier versions of this paper) by keys such that there exists two points A, B such that $B = \mathcal{E}(A)$, $A = \mathcal{E}(B)$ OR $A = \mathcal{E}(A)$, $B = \mathcal{E}(B)$, and such that the pair A, B have the differences with 14 active bits depicted in Fig. 18. The first of the two cases is shown in Fig. 23. Moreover we assume that one of these two values A, B which without loss of generality we can assume to be A, is symmetric with two identical 32-bit halves. This occurs with density $d = 2^{-52}$.

Justification: Again if a pair A, B propagates as in Fig. 18, it will give the same pair A, B with probability 2^{-73}. We have 2^{51} pairs such as in Fig. 18 and for a random GOST key the probability with to map A, B to themselves OR to B, A in Fig. 18 is twice $2^{-73+51} = 2^{-22}$ with sum being 2^{-21} total. Furthermore with probability 2^{-31} one of the two values A, B is symmetric. Overall our weak key occurs with probability 2^{-21-31}.

Now we are going to show how to break these weak keys. Can we for example simultaneously do better than 2^{128} GOST encryptions needed in Fact 49, require less data, and work for a higher proportion of keys d? Or require the same quantity of data but work for more keys than in Section 24.1 and [77]?

Fact 62 (Key Recovery For Family 5.3). Given 2^{32} ACP (Adaptively Chosen Plaintexts) we can obtain 3 KP valid with probability 2^{-9} and recover a weak key in Family 5.3 in time of 2^{119} GOST encryptions and with small memory.

Justification: We guess A which is a symmetric fixed point. Our guess is expected to be true with probability 2^{-0} because we do not expect additional symmetric fixed points to occur by accident. To determine A we need 2^{32} CP because A is symmetric.

Then we guess B and determine C by flipping up to 14 bits of A inside the mask we assumed. Because we have already determined A, this requires further up to 2^{14} ACP (Adaptively Chosen Plaintexts). Now the entropy of $A \oplus B$ is only about 9 bits. Overall we obtain A, B, C and our triple is correct with probability only 2^{-9}.

We have obtained 3 pairs for 8 rounds of GOST which are either $A \mapsto A$, $B \mapsto B$, $C \mapsto \overline{B}$, or in the other case we have included in the definition of Family 5.3, where A, B are swapped, which is not the case shown in Fig. 23, we have the following 3 pairs for 8 rounds: $A \mapsto B$, $B \mapsto A$, $C \mapsto \overline{B}$.

Now we might be tempted to apply Fact 6. With 3 KP it allows to recover the key in time of 2^{110} GOST encryptions. Again, this is **not** going to work. Two out of three pairs share 50 plaintext and ciphertext bits and many internal bits. This means that individual key are less likely to be rejected with such data and there is many more correct keys than 2^{64} which is the case with unrelated encryptions in Fact 6. However, even if we have more solutions, it is very easy to see that each solution is easier to find: this is because two out of the three encryptions share many internal bits with higher probability, there is basically less variables or they have smaller joint entropy. So we need a more precise dedicated attack to see what is the best attack in this case. In Appendix J.1 we show that it can be done in 2^{110} GOST encryptions, cf. Fact 124. The fact the the final result is very similar to the result of Fact 6 should be considered as a coincidence. Overall we need $2^{110+9} = 2^{119}$ GOST encryptions.

Fact 63 (Family 5.3 in the Multiple Key Scenario). Given a population of 2^{52} different random keys, with access to 2^{32} ACP per key, one can recover **one** of these 256-bit keys in total overall time of about $\mathbf{2^{139}}$ GOST encryptions.

Justification: With Fact 62 above, we can break weak keys of Family 5.3 in time of 2^{119} for each key and 2^{32} ACP per key. As in many attacks such as Fact 92 or Fact 50 and unlike in Fact 60, we can reject keys which are not weak at a low cost: we do not have to run the whole attack for a majority of these non-weak keys. This is true because for a random permutation, the probability that $Enc_k()$ has a symmetric fixed point A is about 2^{-32}, and we can reject all the other cases by examining the encryption of all symmetric points and with only 2^{32} KP for non-weak keys. Only for strong keys we require 2^{32} ACP as in Fact 62. Time spend examining a majority of 2^{52} non-weak keys is only 2^{52+32} and can be neglected compared to 2^{119} GOST encryptions spent on one weak key. However additionally we need to run the whole attack on 2^{52-32} non-weak keys for 2^{119} GOST encryptions. All this requires $2^{52-32+119} = 2^{139}$ GOST encryptions total.

25.6 The Future of Periodic Differential Cryptanalysis of GOST

Our basic Family 5.2 attack works for 2^{-21} of weak keys. It uses one family of differential properties based on [38, 40]. Many more such results exist. For example with the situation in Fig. 18 we get 2^{51} pairs which satisfy all the differences and 2^{-21} of weak keys. Here is another situation of type 1+5 using the vocabulary of [40], with the same number of active bits and for which there is also about 2^{51} such pairs:

```
0x07070008 0x08007070
      (4 Rounds)
0x07070008 0x08007070
      (4 Rounds)
0x07070008 0x08007070
```

Fig. 24. Another Event With 3x14 active Bits For 8 Rounds

This gives another similar proportion of about 2^{-21} of weak keys for Family 5.2. Now the point is that these weak key families are rather disjoint and many other such patterns in GOST cipher are expected to exist. It is in fact not excluded that various families of weak keys **might cover the whole key space of GOST**. Therefore a systematic exploration of such families of attacks could lead to efficient single key attacks.

This provides a motivation for further study of differential attacks on GOST which is vast topic with already a substantial history of finding more and more powerful attacks, see [62, 63, 74, 36–40] and this paper. Some other sets from [100] with 40 active bits are studied in Section I.3.

25.7 More Combined Differential Complexity Reduction Attacks

Some additional attacks which use a differential property for 16 rounds of GOST instead of 8 rounds, which is not periodic for 8 rounds, and which use the involution property of the last 16 rounds or a double reflection are described in Appendix H.

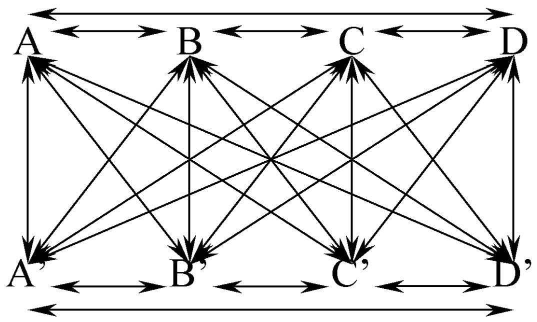
In Section 27.1 we present Family 5.4 which is a straightforward extension of Family 5.3 with one more symmetric fixed point and which leads to a much faster attack in the multiple key scenario. In order however to achieve that we are going to study more advanced and higher order differential properties. We are going to study differentials with 3 and 4 points which is very rare in cryptanalysis.

26 Multiple Approximate Fixed Points, Single Key Bicliques, Higher-Order Truncated Differential Attacks and Black Box Complexity Reductions

In this section we introduce a new general concept which is closely related but distinct from many well known attacks in symmetric cryptanalysis. The main motivation for the introduction of this new concept is the periodicity of GOST and the large variety of self-similarity attacks on GOST which we have already discovered. All these attacks start by guessing many relations such as $A = B$ or $A = \overline{C}$ or $A \oplus \overline{B} \in 0x8070070080700700$ and subsequently they obtain several I/O relations for 8 rounds of GOST which are (jointly) solvable in practice.

We are looking for a sort of magical trick which would be able to enhance such attacks: make the cost of such assumptions simultaneously lower, even though if overall this would have a certain price to pay. Very surprisingly due to the internal structure and poor diffusion inside GOST such a property exists for 8 rounds of GOST and will lead to many interesting new attacks on GOST. Our new concept is essentially an **invariant affine space** common for many encryptions at both sides or a **multiple approximate fixed point** concept.

Definition 26.0.1 (An approximate fixed point biclique). An approximate fixed point biclique with k points and dimension D and for $r = 8$ rounds of GOST is defined as a set of $k{+}k$ values $A, B, \ldots$ and $A', B', \ldots$, such that A' is the encryption of A after 8 rounds etc.. , see Fig. 25, and such that all these $k{+}k$ values lie in the same affine space of dimension D.



$$\begin{tikzcd}
A \arrow[r, leftrightarrow] \arrow[rrr, leftrightarrow, bend left] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] \arrow[drr, leftrightarrow] \arrow[drrr, leftrightarrow] & B \arrow[r, leftrightarrow] \arrow[dl, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] \arrow[drr, leftrightarrow] & C \arrow[r, leftrightarrow] \arrow[dll, leftrightarrow] \arrow[dl, leftrightarrow] \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & D \arrow[dlll, leftrightarrow] \arrow[dll, leftrightarrow] \arrow[dl, leftrightarrow] \arrow[d, leftrightarrow] \\
A' \arrow[r, leftrightarrow] \arrow[rrr, leftrightarrow, bend right] & B' \arrow[r, leftrightarrow] & C' \arrow[r, leftrightarrow] & D'
\end{tikzcd}$$

Fig. 25. An approximate fixed point biclique with $k = 4$

We have discovered a new form of **approximate self-similarity** of GOST which happens to occur in the real life. For example, for one set we can have four input points ($k = 4$) and four output points which share some fixed set of 50 bits, so that $D = 14$, and that for every arrow in Fig. 25, the 50 bits in question are the same. This can be seen as a simultaneous truncated differential property in which every pair has small Hamming weight differences. Surprisingly this occurs for GOST and 256-bit keys generated at random with a non-negligible probability. The key observation is that, as we will see later, the number of such events with 3,4 points is NOT much lower than the number of events with 2 points, see Section 26.3 and Appendix I.3.

26.1 The Power of the New Concept

Our new concept can be seen in terms of **subspace selection**: it brings the two sets of input and output points closer together, for example 8 points as shown in Fig. 25 and this subspace is the same for the inputs and the outputs. This is interesting and allows one to design a plethora of new self-similarity attacks such as all the attacks in this paper, but possibly at an overall lower cost. Now all sort of assumptions which we are used to make in attacks with Complexity Reduction and which relate these points in various ways, are expected to be **less costly**. For example:

1. For example when we assume that $A = B'$, it will now have a much higher probability, of 2^{-14} instead of 2^{-64}.
2. We may also wish to guess B knowing A. In theory this will be of 2^{-14} instead of 2^{-64}. This result however is **pessimistic**. In the real life we have observed that when approximate fixed point bicliques occur, the distribution of the differences are very far from random, on the contrary, they tend to follow a much smaller number of special differential patters which results in **much lower** entropies for each of differences $A \oplus B$ etc. and further lower joint entropy if we need to predict several such values.
   For example in the case where we show the existence of multiple fixed point bicliques such as in fact 64 we have observed that the entropy of the random variable $A \oplus B$ is only about 9 bits instead of 14.
   This entropy is computed over different (unknown) random GOST keys (it might be even lower for a fixed key but this fact may be difficult to use for the attacker). This makes guessing easier than expected.
3. This low entropy phenomenon does NOT concern relations with variables taken from both sides, such as $A = A'$, i.e. A is a fixed point for the first 8 rounds of GOST. It will still be valid with probability 2^{-14} instead of 2^{-64}.
4. In some attacks on 32 rounds, it is possible to show that the probability that in the first two encryptions the state in the middle after the first 16 rounds is simultaneously respectively B' and A', given that it is A' and B' after 32 rounds is now only 2^{-14} instead of 2^{-128}. We can note that the probability is already reduced to 2^{-64} instead of 2^{-128} due to the fact that the last 16 rounds of GOST are an involution, for cryptanalytic applications of this property see for example Section G and Appendix H.3.
5. Other assumptions can have a lower **relative** cost w.r.t. other assumptions, for example if the 50 bits in question are the inactive bits inside $0x8070070080700700$, (cf. [36–40]) and if we already assumed that A is symmetric, then the probability that for example C' is also symmetric, is as low as 2^{-7} because their difference has 14 active bits and positions of these bits are symmetric.
6. In addition some differential assumptions inside the cipher are greatly facilitated by this assumption. For example it is possible to see that for 2 pairs for 8 rounds, if both the input difference and output difference are of the form $0x8070070080700700$ in the sense of [36–40], we may want to study the probability that the middle difference after 4 rounds is also of the form

$0x8070070080700700$, as shown in Fig. 18 in Section 25. This probability is quite high, about 50 %. Such additional differences with very low entropy can make our key recovery attacks substantially faster.

The main idea of our "approximate fixed point biclique" assumption as depicted in Fig. 25 is that it decreases the cost of making many other self-similarity assumptions at an overall reasonable amortized cost. This however needs yet to be demonstrated: we need to show that it leads to attacks on GOST faster than other known attacks. For example in later Fact 69 we show that some full 256-bit GOST keys generated at random can be recovered in total time as low as 2^{117} GOST encryptions, then we will achieve 2^{110} in later Fact 81, and even 2^{101} in Fact 89 which attack also appears in [34]. All these attacks are substantially faster than any other known attacks on GOST.

Before that we need to show that such configurations exist and estimate the probability with which they occur. Even before that we need to find an appropriate set of say 50 bits, which is another highly non-trivial task and which leads to new important optimization problems in cryptanalysis which in general have exponential complexity and therefore we do not expect that it is possible to systematically explore all possible sets of this type. Consequently it is very hard to find a really good attack on GOST of this type, and it is also very hard to know if GOST is secure against such attacks.

26.2 Relation to Known Cryptanalytic Concepts

Our idea of "approximate fixed point biclique" attacks is closely related to several well known concepts in cryptanalysis.

1. In one sense, we have here **multiple approximate fixed points**, where the input and the input differ by for example only 14 bits.
2. The term biblique means complete bipartite graph. Bicliques with many different keys are a very famous recent concept in the cryptanalysis of AES [11]. However we only look at **bicliques with one single key**.
3. Moreover connections in our graph do NOT have the same meaning as in [11]. Our connections with arrows in both directions mean that the two points are related or similar, for example their difference lies in a small linear space. For every arrow in Fig. 25 we can have some interesting relations, for example A and B' will differ by very few bits with a rather high probability, cf. Section 26.1. This also means that $A = B'$ can occur with high probability, and it can occur for any connection in our graph. This being subject to logical relations between such constraints which will affect the probabilities if the attacker wants to impose several conditions simultaneously.
4. Our configurations are a form of a **higher-order truncated differential attack** which works for any number of points $2, 3, 4, \ldots$ and where more than the XOR of all input values is constrained, in fact all the output values are constrained and lie in the same affine space.
5. If we relax the condition that the shared D bits for the k inputs must be the same as the shared D bits for the k outputs then simple configurations with 2 points could also be called **truncated collisions** or (better) **truncated**

I/O collisions (cf. collision test in [52]) and are essentially the same as truncated differentials of Knudsen. For more than 2 points we could call them **truncated I/O multi-collisions**. The term **I/O** emphasizes the fact that collisions occur simultaneously on subsets of bits in the **I**nputs and subsets of bits in the **O**utputs. Some such events are studied in section I.1.

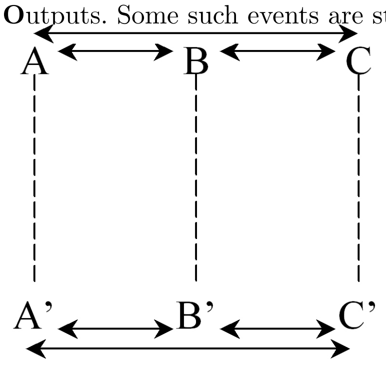


Fig. 26. Truncated I/O multi-collisions with $k = 3$: the k inputs share $n - D$ bits and k outputs share $n - D$ bits which are not necessarily the same $n - D$ bits.

26.3 Approximate Fixed Point Bicliques for 8 Rounds

We can remark that we are looking at a relatively strong property and sets of points which satisfy it are unlikely to exist for many more rounds of GOST or for a random permutation. However the internal structure and poor diffusion inside GOST allows many sets which satisfy the Definition 26.0.1 for some k.

Can this sort of event happen for $k = 2, 3, 4$ a random 256-bit GOST key and a value D much smaller than 64? Quite surprisingly it can.

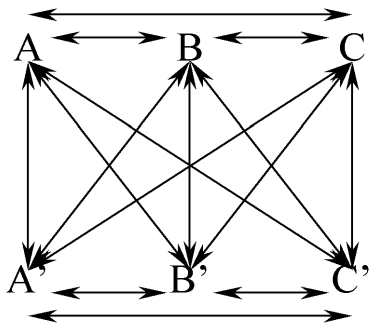


Fig. 27. An approximate fixed point biclique with $k = 3$

Fact 64 (A 2/3/4-point approximate fixed point biclique for 8 rounds of GOST and D = 14). For a typical GOST key we have on average 2^2 possibilities for the set of 4 points A, B, A', B', such that A', B' are the encryption of A, B after 8 rounds, AND which have differences with up to 14 bits and all the points A, B, A', B' share the same set of 50 bits, following the pattern using the mask $0x8070070080700700$ as depicted in Fig. 21.

For a proportion of at least 2^{-8} of GOST keys there exists a set of 50 bits and a set of 6 points A, B, C and A', B', C', such that A' is the encryption of A after 8 rounds etc. , and which have differences with up to 14 bits and share the same set of 50 bits as depicted in Fig. 27.

For another proportion of at least about 2^{-9} of GOST keys there exists a set of 50 bits such and a suitable set of 8 points A, B, C, D and A', B', C', D', which all share 50 bits as depicted on earlier Fig. 25.

Justification: The justification takes several steps. We say that the result will be **at least** the one we compute below because there are many potentially many more interesting affine spaces with 2^{14} elements which should lead to overall higher number of events/a higher proportion of keys. Our construction is based on one fixed differential set from [36–40] which are the same we used in Section 25. We consider the affine space of $D = 14$ defined by the popular mask $0x8070070080700700$

It is possible to observe that for one encryption A, A' for 8 rounds of GOST, if $A \oplus B \in 0x8070070080700700$ then if $A' \oplus B' \in 0x8070070080700700$ with probability $2^{-25.0}$ cf. [37–40]. There is about $2^{64+14}/2! \approx 2^{77}$ couples A, B with suitable input differences, then with probability 2^{-25} for one pair we have the correct output difference $A' \oplus B'$. Furthermore then common 50 bits are the same for the input and for the output with probability 2^{-50}. Thus we expect that there are on average about $2^{77-25-50} \approx 2^2$ pairs which share a fixed 50 common bits as depicted in Fig. 21.

For $k > 2$ the existence of such configurations can be seen as a simultaneous truncated differential attack with 3 or 4 points respectively. As above, if $A \oplus B \in 0x8070070080700700$ then if $A' \oplus B' \in 0x8070070080700700$ with probability $2^{-25.0}$ cf. [37–40]. Then if A, A' and B, B' are the suitable pairs for 8 rounds, it is **easier** to find more such pairs C, C'. Our computer simulation shows that now if $C \oplus A \in 0x8070070080700700$ then $C' \oplus A' \in 0x8070070080700700$ with probability of only about 2^{-22} instead of $2^{-25.0}$. This is due to the fact that some bits of A already have interesting suitable values.

In the same way, there will be a fourth pair D, D' with even better probability of about 2^{-13} which means that for the remaining 2^{-14} values D, there could be several solutions. This probability is quite high and suggests that several additional points will frequently exist and there isn't much difference in probability that a set of 3 suitable A, B, C exist with all the possible pairwise equalities on 50 bits as depicted in Fig. 27 below, and the probability that we have 4 suitable points such as depicted in Fig. 25 for $k = 4$.

There is about $2^{64+14+14}/3! \approx 2^{89.4}$ triples A, B, C with suitable input differences, then with probability 2^{-25} for one pair we have the correct output difference $A' \oplus B'$, and then the third point also has the same 50 bits with probability of only about 2^{-22}. Thus we expect that there are on average about $2^{89-25-22-50} \approx 2^{-8}$ cases which share a fixed 50 common bits which have differences with up to 14 bits as depicted in Fig. 27.

In the same way, there is about $2^{64+14+14+14}/4! \approx 2^{101.4}$ quadruple A, B, C, D with suitable input differences, with probability 2^{-25} for one pair we have the correct output difference $A' \oplus B'$, and then the third point is correct with probability of only about 2^{-22}, this multiplied by 2^{-13} for the fourth point and finally by 2^{-50} for the 50 bits on both sides being equal. Thus we expect on average about $2^{101-25-22-13-50} \approx 2^{-9}$ cases which share a fixed 50 common bits and have differences with up to 14 bits as depicted on earlier Fig. 25.

Real Life Events

No theory can replace to verify if the events which we study really happen as predicted. For example it is very surprising to see that events with 4 points will occur more or less as frequently as events with 3 points. To validate this we have tried 2^{39} encryptions for 8 rounds with random keys an initial random difference with 14 active key bits within $0x8070070080700700$ and if after the 8 rounds the final difference was also within $0x8070070080700700$, then we count how many other plaintexts with the same 50 bits also produce the same 50 bits as the two cases. In our simulation we have seen exactly 21 events with 4 points and also exactly 21 events with 4 points with random keys and random 50 bits.

Examples: We exhibit one event with 4 points from our simulation. This event have been generated strictly at random and has no special properties other than those which might occur naturally at random for such events. The data are self-explanatory. We also display the difference between each plaintext and the last plaintext, and the same for the ciphertexts.

```
8 rounds 4 points 50 inactive bits 8070070080700700
key=C4EEEC4D9FC4A3C55DB81B7BEE470567396682007AE8D9B59E3FD9A3225BC7B4
P=6492F05231436EBF E4D2F152317368BF 64D2F452B14368BF E492F552B1736EBF
  8000030080300000 8040000000000200 0040030080300200 0000000000000000
C=89C3449606C28E22 09C3409606C28F22 89C3409686A28822 09C3449686A28922
  8000030080000400 0000030080000000 8000000000000400 0000000000000000
```

Events With Other Masks and More Points Or/And Single Keys

Fact 64 is only a lower bound, the probability is in fact higher, as several other interesting affine spaces of dimension 14 can exist, which contribute another proportion of GOST keys. One example of such additional space is given in Fig. 24 page 103, and some other possibilities are discussed in Section 3.1. in [40]. This is for linear spaces of the same size with $D = 14$. In Appendix I.3 we present a complete of analogue of Fact 64 with $D = 24$. In Appendix I.5 another case with $D = 28$.

More Points and Single Keys In our simulation with 2^{39} encryptions with random keys we more than 4 points have occurred only once. Moreover the events studied in Section 26 happen only for some keys, for example 2^{-9} GOST keys. However if we take a large mask we can obtain events with more points or/and which happen for all GOST keys, see Appendix I.5.

27 New Improved Attacks With Multiple Reflection

In this section we propose several new attacks. Our goal is to see if we can achieve results similar or better than 2^{129} total cost per 256-bit key with more data overall but possibly with a small amount of data per key. The first attack uses approximate fixed point bicliques, some other attacks are traditional attacks without any differential properties, and interestingly several attacks exist which can be constructed both using approximate fixed point bicliques and without them, and we will be able to compare both sorts of attacks. These additional differential properties lead to overall lower proportion of keys for which the attack will work, however we obtain substantially faster running times in the version with internal correlations.

27.1 A Triple Fixed Point Biclique Attack With Double Reflection

This attack can be seen as a straightforward extension of Family 5.3. with one more symmetric fixed point like on the left hand side of Fig. 23. Since it has two symmetric fixed points it also is an extension of attacks from Section 24 which allows the attacker to use the fact that symmetric fixed points happen with higher frequency than expected. More precisely we consider a new Family 5.4 defined as follows.

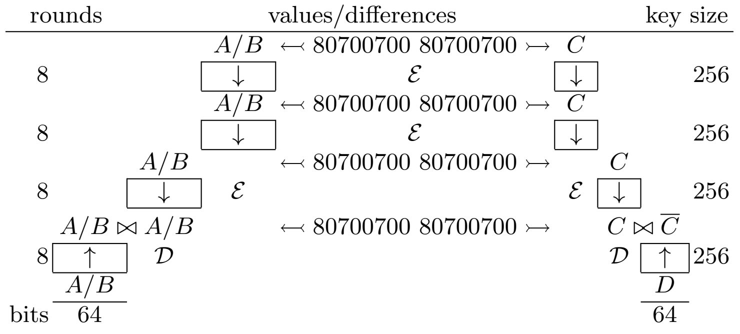


Fig. 28. A Differential-Two-Symmetric-Three-Fixed-Points Attack on GOST

Fact 65 (Weak Keys Family 5.4, $d = 2^{-77}$). We define the Weak Keys Family 5.4 by keys such that there exists three points A, B, C such that $A = \mathcal{E}(A)$, $B = \mathcal{E}(B)$, $C = \mathcal{E}(C)$, AND such that A, B, C lie in some affine space of dimension 14 defined by some set of fixed 50 inactive bits which follow the pattern $0x8070070080700700$, AND such that two points A, B are symmetric but not C. This occurs with density approximately $d = 2^{-77}$.

Justification: We recall Fact 64. For a proportion of at least 2^{-8} of GOST keys there exists a set of 50 bits and a set of 6 points A, B, C and A', B', C', such that $A \mapsto A'$, $B \mapsto B'$ and $C \mapsto C'$ after 8 rounds, and such that all the 6 points share the same set of 50 bits as depicted in Fig. 27.

Then the probability that $A \oplus B = A' \oplus B'$ and simultaneously $A \oplus C = A' \oplus C'$ is maybe about 2^{-16} due to low joint entropy of these differences, cf. Section 26.1. Then the probability that $A = A'$ is about 2^{-14} see Section 26.1. This implies that also $B = B'$ and $C = C'$. Furthermore, the probability that A is symmetric is about 2^{-32}. Finally given the fact that and A is symmetric, for B to become symmetric it is sufficient for the difference $A \oplus B$ is symmetric which occurs with expected probability of at most 2^{-7}, because these values differ only on up to 14 bits with differences such as $A \oplus B$ being of type $0x8070070080700700$ which is a symmetric pattern. Overall we obtain $d = 2^{-8-16-14-32-7} = 2^{-77}$ over all GOST keys and we get the situation depicted in Fig. 29.

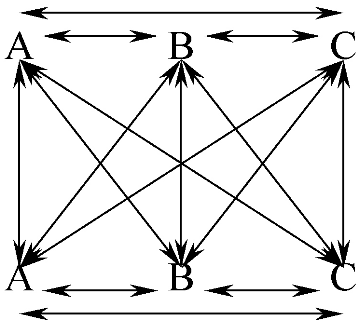


Fig. 29. Key assumption in Family 5.4.

Now we are going to show how to break these weak keys.

Fact 66 (Key Recovery For Family 5.4). Given 2^{32} ACP (Adaptively Chosen Plaintexts) we can obtain 4 KP for 8 rounds valid with probability 2^{-9} and recover a weak key in Family 5.4 in time of 2^{103} GOST encryptions and with small memory.
Justification: We guess A, B which are symmetric fixed points for 32 rounds of GOST. Our guess is expected to be true with probability 2^{-0} because we do not expect any symmetric fixed points to occur by accident. To determine A, B we need 2^{32} CP because they are symmetric. Then we guess C by flipping up to 14 bits of A inside the mask we assumed and because we have already determined A, this requires further 2^{14} ACP (Adaptively Chosen Plaintexts). Let D be such that $\mathcal{E}(D) = \overline{C}$, see Fig. 28. Given C we determine D by encryption. Overall we can obtain A, B, C, D and our quadruple is correct with probability 2^{-9} due to low entropy of $B \oplus C$ cf. Section 26.1. We have obtained 4 pairs for 8 rounds of GOST which are $A \mapsto A$, $B \mapsto B$, $C \mapsto C$, $D \mapsto \overline{C}$.

Now we might be tempted to apply Fact 6. We need a dedicated attack with special differences which works in the same way as another dedicated attack described in Fact J.2. With Fact J.2 we obtained 2^{94} and we conjecture that with even more differences than in Fact 125 for which we also could obtain 2^{94}, we estimate that key recovery can also be done in at most 2^{94} GOST encryptions. Further research is needed to validate this claim exactly and improve the running time.

Overall we need $2^{94+9} = 2^{103}$ GOST encryptions.

Fact 67 (Family 5.4 in the Multiple Key Scenario). Given a population of 2^{77} different random keys, with access to 2^{32} ACP per key, one can recover **one** of these 256-bit keys in total overall time of about $\mathbf{2^{116}}$ GOST encryptions.

Justification: With Fact 66 above, we can break weak keys of Family 5.4 in time of 2^{103} for each key and 2^{32} ACP per key. We can reject keys which are not weak at a low cost by using the fact that for a random permutation, the probability that $Enc_k()$ has two symmetric fixed points A, B is about 2^{-64}, and we can reject all the other cases by examining the encryption of all symmetric points and with only 2^{32} KP for non-weak keys. Only for strong keys we require 2^{32} ACP as in Fact 66. Time spend examining a majority of 2^{77} non-weak keys is only 2^{75+32} and can be neglected compared to 2^{103} GOST encryptions spent on one weak key. However additionally we need to run the whole attack on 2^{77-64} non-weak keys for 2^{103} GOST encryptions. All this requires $2^{77-64+103} = 2^{116}$ GOST encryptions total.

27.2 A Quadruple Fixed Point Biclique Attack With Double Reflection

In this attack we are going to assume an approximate fixed point biclique (cf. Fig. 25) such that in addition A, B, C, D are mapped to B, C, D, A after 8 rounds.

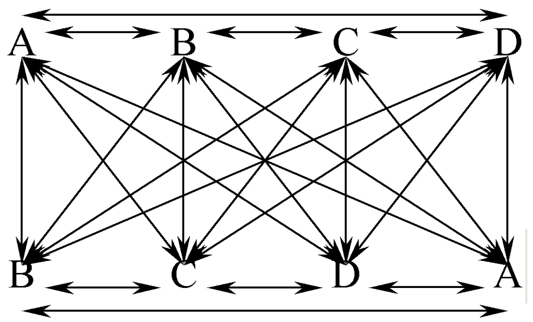


Fig. 30. Key assumption in Family 6

Fact 68 (Weak Keys Family 6, $d = 2^{-84}$).

We define the Weak Keys Family 6 by keys such that we have the situation depicted in Fig. 30 with A, B, C, D becoming B, C, D, A after 8 rounds of GOST, AND A, B, C, D lie in some affine space of dimension 14 defined by some set of fixed 50 inactive bits which follow the pattern $0x8070070080700700$, AND such that both C, D are symmetric.

Such keys occur with density $d = 2^{-84}$.

Justification: Following Fact 64 a basic configuration occurs for a proportion of at least 2^{-9} GOST keys. Then the probability that $A' = B$ is about 2^{-14} see Section 26.1. Likewise following Section 26.1 we estimate that the joint sets of differences $A' \oplus B', B' \oplus C', C' \oplus D'$ follows the same probability distribution as $A \oplus B, B \oplus C, C \oplus D$ with low entropy which is probably not bigger than 20 bits. This following our preliminary rough simulations on the surprisingly low entropy of such sets of differences which occur in such configurations. Overall we estimate that the situation depicted in Fig. 30 with A, B, C, D lying in some affine space of dimension 14 defined by some set of fixed 50 inactive bits which follow the pattern $0x8070070080700700$, occurs with probability of only roughly about $2^{-9-14-20} = 2^{-43}$ over all GOST keys.

Furthermore, the probability that C is symmetric is about 2^{-32} and the probability that both C, D are symmetric is only about 2^{-41} as these values differ on up to 14 bits which difference $C \oplus D$ is symmetric with probability of at least 2^{-7}, this given the fact that set of 14 bits comes from the pattern $0x8070070080700700$ which is symmetric.

Overall we obtain $d = 2^{-43-41} = 2^{-84}$ over all GOST keys.

Improvements. Moreover such keys occurs with a higher probability if we consider other patterns than $0x8070070080700700$, which we leave for future research on how to improve our attack.

Fact 69 (Weak Key Family 6 Attack).
For each key in Family 6 given 2^{33} CPCC (Chosen Plaintexts and Chosen Ciphertexts) we can obtain 4 KP for 8 rounds true with probability 2^{-0} and can recover the key in time of at most 2^{94} GOST encryptions.

If we dispose of 2^{84} devices with distinct keys, one of the keys can be recovered in total time of 2^{117} GOST encryptions.

Justification: We obtain pairs for 8 rounds as illustrated in Fig. 31.

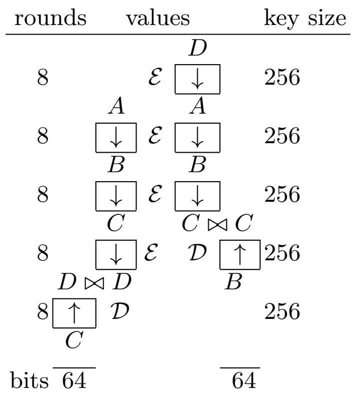


Fig. 31. Family 6 of Weak Keys

Normally with arbitrary A, B, C, D and without approximate fixed point bicliques the situation depicted in Fig. 31 occurs with probability $2^{64-32-32-64}$ which is $d = 2^{-64}$ over the GOST keys. It can also be seen as the situation from Fig. 9 with an additional assumption $Z = D$. Here it occurs with $d = 2^{-84}$ and we get a much stronger property with $150 = 3 \times 50$ additional equalities on bits of A, B, C, D. This is very good and this observation alone justifies all our work on approximate fixed point bicliques.

It is possible to see that we are unlikely to see observe any A, B, C, D such as in Fig. 31 and with shared 50 bits which are not correct. The attacker does NOT need to guess any of A, B, C, D. They can be obtained with near certainty as follows. For a random permutation, the probability that for some symmetric point C after decryption, the plaintext A has the same 50 bits is about $2^{32-50} = 2^{-18}$. In addition the probability that for some symmetric point D after encryption, the ciphertext B has the same 50 bits is about further $2^{32-50} = 2^{-18}$. Finally the probability that the 50 fixed bits shared by both symmetric C and D are the same is further 2^{-25}. Overall the situation visible to the attacker implied by the observable inputs and outputs in Fig. 31 AND C, D being symmetric AND the fact that A, B, C, D should lie in the same space with shared 50 bits, assuming also that these shared bits must be symmetric set of 50 bits, occurs with probability $2^{-18-18-25} = 2^{-61}$ for various GOST keys. Thus for weak keys we obtain 4 pairs for 8 rounds for free, because a second such event is unlikely to happen.

Unhappily we cannot just apply Fact 7 in 2^{94} for key recovery because our 4 pairs are quite special. We need a dedicated attack with special differences which works in the same way as another dedicated attack described in Fact J.2. With Fact J.2 we obtained 2^{94} and we conjecture that with even more differences than in Fact 125 for which we also could obtain 2^{94}, we estimate that key recovery can also be done in at most 2^{94} GOST encryptions. Further research is needed to validate this claim.

Now if we dispose of 2^{84} devices with distinct keys, we see that only with probability 2^{-61} the attacker needs to run the attack. He can reject all the other cases after examining some 2^{33} of data for this key which can be checked in time of about 2^{84+33} CPU clocks, which is about 2^{107} GOST encryptions. Then the attacker needs to run the analogue of Fact 125 for about 2^{84-61} times 2^{94} GOST encryptions. This is about $2^{84-61+94} = 2^{117}$ GOST encryptions which dominates our attack.

Future Research. We expect that even better attacks following the same framework exist, by combination of several different spaces of dimension 2^{14}.

27.3 Quadruple Reflection Attack Family 7.1

We have the following new attack with quadruple reflection.

Fact 70 (Weak Keys Family 7.1, $d = 2^{-64}$, 6 KP for 8R).

We define the Weak Keys Family 7.1 by keys such that there exists a point A such that all the four points A, $B = \mathcal{E}(A)$, $C = \mathcal{E}^2(A)$, $D = \mathcal{E}^3(A)$ are all symmetric. Such keys occur with density $d = 2^{-64}$, as there are 2^{64} possible A and then we have four consecutive events each occurring with probability 2^{-32}.

For every key in Family 7.1, and given 2^{32} ACC (Adaptively Chosen Ciphertexts) we obtain 6 P/C pairs for 7 rounds of GOST correct with probability about 2^{-33}.

Justification: We proceed as follows.

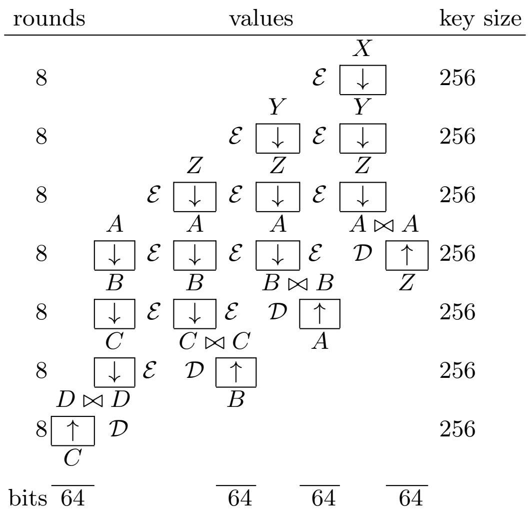


Fig. 32. A weak-key quadruple reflection attack with 6 pairs for 8 rounds

1. We observe that $Enc_k(A) = C$ and both A and C are symmetric. These pairs are known to the attacker given 2^{32} CP or 2^{32} CC. Moreover the attacker can obtain the values A, C essentially and almost "for free".
2. This is because it is an exceptional event, which for a random permutation occurs only once on average: there are 2^{32} symmetric A and the encryption is symmetric with probability 2^{-32}. With our assumption, there is about two pairs of symmetric A, C, one being the correct one.
3. We can guess which pair A, C is the correct one as in Fig. 32 with probability 2^{-1}.
4. We can determine Y for free by decryption. At this stage we need 2^{32} CC.
5. Now we guess B, which is correct with probability 2^{-32}.
6. Now we can determine both Z and X be double decryption. This requires only 2^{32} ACC (Adaptively Chosen Ciphertexts) because we decrypt two times from a symmetric value B.

7. Thus overall given 2^{32} ACC we have obtained 6 P/C pairs for 8 rounds of GOST correct with probability about 2^{-33}.
   These pairs are: $X \mapsto Y$, $Y \mapsto Z$, $Z \mapsto A$, $A \mapsto B$, $B \mapsto C$, $C \mapsto D$.

From here we obtain a very interesting attack:

Fact 71 (Key Recovery for Weak Keys Family 7.1, $d = 2^{-64}$).
One can recover the keys for the Weak Keys Family 7.1 with 2^{32} ACC, running time of $\mathbf{2^{116}}$ GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 70 and Fact 127 which gives 2^{83} GOST encryptions for 6 KP and 8 rounds. Overall we can recover the key in time of 2^{83+33} GOST encryptions. There is no false positives expected in this attack.

27.4 Conversion of Family 7.1 Into A "Regular" Attack with Multiple Keys Generated At Random

We have the following conversion with early rejection of non-weak keys:

Fact 72 (Key Recovery for A Diverse Population of 2^{64} Keys).
If we have a diverse population of at least 2^{64} different 256-bit GOST keys generated at random, with access to 2^{32} ACC per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{180} GOST encryptions.

Justification: Unhappily we need to run 2^{64} times the attack of Fact 71.

27.5 A Better Quadruple Reflection Attack Family 7.2

The Family 7.2 is the same as Family 7.1 in which the same situation as in Fig. 34 is achieved with additional constraints on A, B, C, D which are shown to be possible due to the existence of approximate fixed points bicliques.

Fact 73 (Weak Keys Family 7.2, $d = 2^{-84}$, 6 KP for 8R).
We define the Weak Keys Family 7.2 by keys such that there exists a point A such that all the four points A, $B = \mathcal{E}(A)$, $C = \mathcal{E}^2(A)$, $D = \mathcal{E}^3(A)$ are all symmetric. and such that all the four points A, B, C, D lie in the same affine space of 2^{50} elements. This occurs with density $d = 2^{-84}$.

For every key in Family 7.2, and given 2^{32} ACC (Adaptively Chosen Ciphertexts) we obtain 6 P/C pairs for 7 rounds of GOST correct with probability about 2^{-4}.

Justification: We start with Fig. 27. Following Fact 64 a basic configuration with $k = 3$ occurs for a proportion of at least 2^{-8} GOST keys. Then the probability that $A' = B$ is about 2^{-14} see Section 26.1. Now the probability that $B' = C$ is about not 2^{-14} but only about 2^{-9}. This is our observations concerning the surprisingly low entropy of differences such as $A \oplus B$ which occur in such configurations for 8 rounds of GOST. Since now $A' = A$ in order to have $B' = C$ it is sufficient to have $A' \oplus B' = B \oplus C$ which occurs with probability of rather 2^{-9} than 2^{-14}, see Section 26.1. Overall we estimate that the situation depicted in Fig. 33 with A, B, C, D lying in some affine space of dimension 14 defined by some set of fixed 50 inactive bits which follow the pattern $0x8070070080700700$, occurs with probability of only roughly about 2^{-31} over all GOST keys.

Furthermore, the probability that A is symmetric is about 2^{-32}. We observe that given the fact that and A is symmetric, for B to become symmetric it is sufficient for the difference $A \oplus B$ is symmetric with probability 2^{-7}, because these values differ on up to 14 bits with differences such as $A \oplus B$ being of type $0x8070070080700700$ which is a symmetric pattern. Accordingly the probability that all the A, B, C, D are symmetric is only about $2^{-32-7-7-7} = 2^{-53}$.

Overall we obtain $d = 2^{-31-53} = 2^{-84}$ over all GOST keys which can be compared to 2^{-64} in Family 7.1. Here for a higher price of 2^{-84} we get a much stronger property with $150 = 3 \times 50$ additional equalities on bits of A, B, C, D.

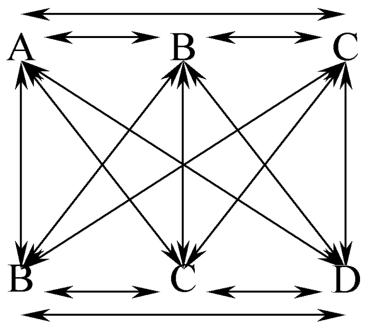


Fig. 33. Key assumption in Family 7.2

As in Fact 70 we can determine A, C for free. However it is easier to guess B, see Fig. 32. Against due to reduced entropy which we have observed we expect that the entropy of the symmetric B knowing symmetric A is only 4 bits instead of 7 in the worst case. Thus our triple A, B, C can be guessed easily and will be correct with probability of roughly about 2^{-4}.

Now we are going to investigate the complexity of breaking such weak keys.

Fact 74 (Key Recovery for Weak Keys Family 7.2, $d = 2^{-84}$).
One can recover the keys for the Weak Keys Family 7.2 with 2^{32} ACC, running time of about $\mathbf{2^{87}}$ GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 70 and Fact 127 which gives 2^{83} GOST encryptions for 6 KP and 8 rounds. Due to abundant amount information provided by 6 pairs, even if related, we do NOT expect that this complexity of 2^{83} increases in our cases with A, B, C, D lying in a smaller affine subspace. Overall we can recover the key in time of 2^{83+4} GOST encryptions.

Furthermore we have the following conversion with early rejection of non-weak keys:

Fact 75 (Key Recovery for A Diverse Population of 2^{64} Keys).
If we have a diverse population of at least 2^{64} different 256-bit GOST keys generated at random, with access to 2^{32} ACC per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{146} GOST encryptions.

Justification: We do NOT need to run 2^{82} times the attack of Fact 74. For a given random key the probability that there exists a pair of symmetric texts A, C such that $Enc_k(A) = C$ is about 1. Furthermore the probability that $A \oplus B$ is zero on our symmetric set of 50 bits is equal to 2^{-25} because of the symmetry. Thus we can reject a proportion of cases and we need to apply Fact 74 only 2^{84-25} times. Overall we expect to recover one key in time of about $2^{84-25+87} = 2^{146}$ GOST encryptions.

28 New Attacks With Multiple Symmetric Points

In this section we propose several further new attacks. An in the previous Section 27 we exploit approximate fixed point bicliques and attempt to try to improve the attacks with additional differential properties. In Section 27.5 these additional differential properties lead to overall lower proportion of keys for which the attack works however overall the attack is substantially faster.

Could it lead to a higher proportion of keys? In this paper sometimes we are really very surprised to find out that with additional properties the appropriate events are expected to happen with a higher probability than expected, which means that we have underestimated the frequency of these events for the real-life GOST cipher by assuming that certain events are independent, while they are strongly correlated. This unexpected reduction in probability has already happened in Section 24.5 and made our multiple key attack about 2^4 times slower because the complexity of the last step of the attack would be increased if there are relations between points, or we have to avoid these cases and concentrate on cases where the two points are uncorrelated. Later in this section we will see examples where there is enough data for the last step and the attacker is able to filter out all the non-weak keys with near certainty. This will lead to particularly efficient attacks.

First we describe one basic attack without internal correlations.

28.1 Small Size Cycle With Symmetric Points - Family 8.1

Here is another family of weak keys. We define weak key family Family 8.1. Later we will define a slight variant of it which will be called Family 8.2.

Fact 76 (Weak Keys Family 8.1, $d = 2^{-98}$, Getting 3 KP for 8R). We define the Weak Keys Family 8.1 by keys such that there exists A such that $\mathcal{E}^3(A) = A$, and all the three values $A, B = \mathcal{E}^2(A), C = \mathcal{E}^3(A)$ are symmetric (both 32-bit halves are equal). This occurs with density $d = 2^{-98}$.

For every key in Family 8.1, given 2^{32} CP we can obtain 3 P/C pairs for 8 rounds of GOST, correct with probability close to 1.

Justification: The expected number of cycles of length 3 for a random permutation is $1/3 = 2^{-1.6}$ cf. [85] for detailed statistics on random permutations. The probability that a random 64-bit permutation has a cycle of length 3 with 3 symmetric points is roughly about $2^{-96-1.6} \approx 2^{-98}$.

We proceed as follows:

1. We observe that since $\mathcal{E}^3(A) = A$, we also have $\mathcal{E}^3(B) = B$ and $\mathcal{E}^3(C) = C$.
2. We have

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

   In the same way, $Enc_k(B) = A$ and $Enc_k(C) = B$.
   If we consider the 3-fold iteration $Enc_k^3(\cdot)$ we observe that all the three points A, B, C are fixed points for $Enc_k^3(\cdot)$ and they are symmetric fixed points of $Enc_k^3(\cdot)$.

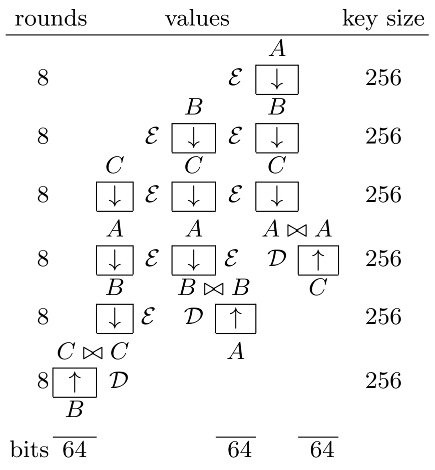


Fig. 34. Weak Key Family 8.1 With Triple Reflection And 3 KP For 8R

3. Given 2^{32} CP (all values we use are symmetric) we can identify A, B, C because they form a cycle of length 3 for $Enc_k(\cdot)$ with 3 elements and we don't expect that more such cycles exist in which all these elements would also be symmetric. It would be an unlikely event.
4. We get 3 P/C pairs for 8 rounds which are $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$, $\mathcal{E}(C) = A$ and these are correct with probability close to 1.

Fact 77 (Key Recovery for Weak Keys Family 8.1, $d = 2^{-98}$).
For the weak keys of Family 8.1 one can enumerate 2^{64} key candidates given 2^{32} CP, with running time of 2^{110} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 76 and 2^{110} of Fact 6 for 3 KP.

28.2 Conversion of Family 8.1 Into A "Regular" Attack with Multiple Keys Generated At Random

We have the following conversion with early rejection of non-weak keys:

Fact 78 (Key Recovery for A Diverse Population of 2^{98} Keys).
If we have a diverse population of at least 2^{98} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{120} GOST encryptions.

Justification: We consider $2^{97.6}$ random devices and we do NOT apply the Fact 77 for each device. Instead we examine the data, 2^{32} CP with the encryption of all symmetric points, to see if our attack is likely to be applicable. We scan for all cases where the encryption is also symmetric and list all cases where this is true. For a random permutation the probability that one symmetric point gives another symmetric point after encryption is 2^{-32} and there are 2^{32} points. Therefore we expect to find one on average but frequently also 0 or several such cases. We examine these few cases for a cycle of length 3. This is going to happen only with probability $2^{-97.6}$. For all except a proportion of about $2^{-97.6}$ GOST keys, all those where the key is not weak, we can reject them by checking 2^{32} plaintexts as explained here in time of $2^{97.6+32}$ CPU clocks, which happily is significantly less than $2^{97.6+32}$ GOST encryptions, but rather about $2^{119.6}$ GOST encryptions.

For the remaining proportion of $2^{-97.6}$, we expect them to come from Family 8.1 with overwhelming probability. Therefore for all the cases where the key is weak, we enumerate 2^{64} candidates for the correct key in time 2^{110} GOST encryptions due to Fact 77, and check which key is correct with additional pairs for full 32 rounds. Overall we expect to recover one key in time of about $2^{110} + 2^{119.6} \approx 2^{120}$ GOST encryptions.

28.3 Improved Small Size Cycle Attack - Family 8.2

The Family 8.2 is the same as Family 8.1 in which the same situation as in Fig. 34 is achieved with additional constraints on A, B, C which are shown to be possible due to the existence of approximate fixed points bicliques.

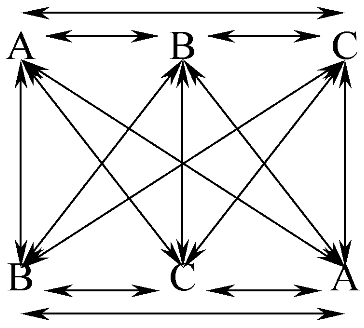


Fig. 35. Key assumption in Family 8.2

Fact 79 (Weak Keys Family 8.2, $d = 2^{-84}$, 3 KP for 8R). We define the Weak Keys Family 8.2 by keys such that there exists A such that $\mathcal{E}^3(A) = A$, and all the three values $A, B = \mathcal{E}^2(A), C = \mathcal{E}^3(A)$ are symmetric, and such that all the three points lie in the same affine space of 2^{50} elements. This occurs with density $d = 2^{-84}$.

For every key in Family 8.1, given 2^{32} CP we can obtain 3 P/C pairs for 8 rounds of GOST, correct with probability close to 1.

Justification: Following Fact 64 a basic configuration with $k = 3$ occurs for a proportion of at least 2^{-8} GOST keys (an alternative justification can be found in Appendix I). Then the probability that $A' = B$ is about 2^{-14} see Section 26.1. Likewise we estimate that the joint sets of differences $A' \oplus B', B' \oplus C'$ follows the same probability distribution as $A \oplus B, B \oplus C$ which entropy is probably not bigger than further 16 bits, then following our preliminary rough simulations on the surprisingly low entropy of such sets of differences which occur in such configurations, see Section 26.1. Overall we estimate that the situation depicted in Fig. 35 with A, B, C lying in some affine space of dimension 14 defined by some set of fixed 50 inactive bits which follow the pattern $0x8070070080700700$, occurs with probability of only roughly about $2^{-8-14-16} = 2^{-38}$ over all GOST keys.

Furthermore, the probability that B is symmetric is about 2^{-32}, in the same way as in Family 7.2 the probability that the three points A, B, C are all symmetric is only about $2^{-32-7-7}$. Overall we obtain $d = 2^{-38-32-7-7} = 2^{-84}$ over all GOST keys which is surprisingly low compared to 2^{-98} in Family 8.1.

Now we are going to investigate the complexity of breaking such weak keys.

Fact 80 (Key Recovery for Weak Keys Family 8.2, $d = 2^{-84}$).
For the weak keys of Family 8.2 one can recover the key given 2^{32} CP, with running time of 2^{110} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 79 and a dedicated version of Fact 6 for 3 KP such as Fact 124 in Section J.1, for which we expect still about 2^{110} GOST computations even though we will need to enumerate and check with additional pairs for more than 2^{64} key candidates, which however probably still takes significantly less time than 2^{110} GOST computations. This is a preliminary rough and simplified analysis and the calculation of the exact complexity of this attack requires a further study.

Furthermore as for Family 8.1, we have the following conversion with early rejection of non-weak keys:

Fact 81 (Family 8.2 Attack For a Population of 2^{84} Random Keys).
If we have a diverse population of at least 2^{84} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{110} GOST encryptions.

Justification: As in Section 28.2 the probability that a random 64-bit permutation has a cycle of length 3 with 3 symmetric points is about 2^{-98}. Thus for all except the correct device out of 2^{84} we can reject this device right away after checking 2^{32} plaintexts in time of 2^{84+32} CPU clocks which is about 2^{106} GOST encryptions. This not counting additional differences on A, B, C which we also have in our attack.

For the remaining one case which really is our weak key with correct additional differences we apply Fact 80 which gives about 2^{110} GOST encryptions and with negligible memory. Overall we expect to recover one key in total overall time of about 2^{110} GOST encryptions.

28.4 On Triple and Quadruple Symmetric Fixed Points in GOST

In Section 24.3 we have established that pairs of symmetric points for 8 rounds and also the full 32-bout GOST occur with probabilities higher than expected. In Section 28.3 we have seen that cycles of length 3 with 3 symmetric fixed points occur with high probability. In this Section we look at another closely related question: triples and quadruples symmetric fixed points which are going to be used in later the Family 8.3. and 8.4. attacks.

Fact 82 (Frequency of Triple and Quadruple Symmetric Fixed Points). Following Section 45 we have established that for the default set of GOST S-boxes the probability that for 8 rounds there are two symmetric fixed points sharing as many as 50 bits is at least 2^{-60} instead of about $2^{-2-64-50}$ for a random permutation.

The probability that there are three symmetric fixed points sharing the same set of 50 bits is at least 2^{-70} instead of roughly about $2^{-4-96-50}$ for a random permutation.

For four symmetric fixed points sharing the same set of 50 bits it is at least 2^{-79} instead of roughly about $2^{-6-128-50}$ for a random permutation.

Justification: Let $A' = \mathcal{E}(A)$ and $B' = \mathcal{E}(B)$ and $C' = \mathcal{E}(C)$ as in Fig. 27 page 107. Following Fact 64 for a proportion of at least 2^{-8} of GOST keys there exists A, B, C such that all the 6 points share the same set of 50 bits following the mask $0x8070070080700700$. Now the probability that A is symmetric is 2^{-32}. Then the probability that $A = A'$ is 2^{-14}. Then the probability that $A \oplus B = A' \oplus B'$ and simultaneously $A \oplus C = A' \oplus C'$ is maybe about 2^{-16} due to low joint entropy of these differences, cf. Section 26.1.

Overall for a proportion of at least $d = 2^{-8-32-14-16} = 2^{-70}$ of GOST keys we have three symmetric fixed points A, B, C sharing the same 50 bits.

In the same way for a proportion of maybe $d = 2^{-9-32-14-24} = 2^{-79}$ of GOST keys we have four symmetric fixed points A, B, C, D sharing the same 50 bits.

Warning: These are very rough estimations and they require further research.

Remark 1. This is much higher than what would get for having just 3/4 symmetric fixed points without any extra condition which is roughly about $2^{-4-96}/2^{-6-128}$ for a random permutation.

Remark 2. Again this reduction in probability was only demonstrated for the default set of GOST S-boxes. However, this is a conservative estimate with just one mask $0x8070070080700700$ and these probabilities get higher if take into accounts other masks. We expect similar results for other sets of S-boxes.

Remark 3. Again symmetric fixed points for 8 rounds are also symmetric fixed points for the full 32-round GOST [75, 77]. As in Fact 47 page 87 we obtain another distinguisher attack on full GOST which is stated below.

Fact 83 (More Distinguisher Attacks on Full GOST). This distinguisher extends the result of [77] and of Fact 47 page 87 from two to three points. It works is we have access to many instances of GOST with different keys. For the default set of GOST S-boxes the probability that there are three symmetric fixed points is at least 2^{-70} instead of about 2^{-100} for a random permutation. Moreover the probability that there are three symmetric fixed points sharing as many as 50 bits in $0x8070070080700700$ is also at least 2^{-70} instead of about 2^{-150} for a random permutation.

The probability that there are four symmetric fixed points is at least 2^{-79} instead of about 2^{-134} for a random permutation. Moreover the probability that there are four symmetric fixed points sharing 50 bits in $0x8070070080700700$ is also at least 2^{-79} instead of about 2^{-184} for a random permutation.

Warning: These are **very rough estimations** and they require further research. Unhappily we do not dispose of a computing power of 2^{66+32} necessary to check if GOST keys selected at random will really have 2,3 or 4 symmetric fixed points as predicted. So we need to find a special approximation methodology to validate this result. Our early simulations trying to validate this fact give diverging results and do NOT confirm the exact figures given above. These results should be therefore considered as a first approximation.

28.5 Triple Symmetric Fixed Point Attack Family 8.3

The Family 8.3 is an extension of Family 0 attack from [75, 77] and Family 4.2 attack based on [77] with yet another (third) symmetric fixed point. It is also somewhat related to Family 8.1 and 8.2 attacks as in all these attacks we are dealing with symmetric points lying in cycles of small size, so that they can be identified by the attacker.

Fact 84 (Weak Keys Family 8.3, $d = 2^{-70}$, 3 KP for 8R). We define the Weak Keys Family 8.3 by keys such that there exists three symmetric fixed points A, B, C for the 8 rounds $\mathcal{E}()$. With the default GOST S-boxes this occurs with density of at least $d = 2^{-70}$.

For every key in Family 8.3, given 2^{32} CP we can obtain 3 P/C pairs for 8 rounds of GOST, correct with probability close to 1.

Justification: We get $d = 2^{-70}$ by applying Fact 82. The three pairs are then simply $\mathcal{E}(A) = A$, $\mathcal{E}(B) = B$ and $\mathcal{E}(C) = C$. How do we break these keys?

Fact 85 (Key Recovery for Weak Keys Family 8.3, $d = 2^{-70}$).
For the weak keys of Family 8.3 one can recover the key given 2^{32} CP, with running time of 2^{120} GOST encryptions and with negligible memory.

Justification: We need to design a special dedicated version of Fact 124 from Section J.1. We expect at most 2^{120} GOST computations. Finally as previously we have the following conversion with early rejection of non-weak keys:

Fact 86 (Family 8.3 Attack For a Population of 2^{70} Random Keys).
If we have a diverse population of at least 2^{70} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in a total overall time of about 2^{120} GOST encryptions.

Justification: The probability that a random 64-bit permutation has 3 symmetric fixed points is about 2^{-100}. This not counting additional differences on A, B, C which we also have in our attack which could be used to reject with even higher probability. Thus for all except the correct device out of 2^{70} we can reject this device right away after checking 2^{32} plaintexts in time of 2^{70+32} CPU clocks which is about 2^{92} GOST encryptions.

For the remaining one case which we expect to be really our weak key with correct additional differences we apply Fact 85 which gives about 2^{120} GOST encryptions and negligible memory. Overall we expect to recover one key in time of about 2^{120} GOST encryptions.

28.6 Quadruple Symmetric Fixed Point Attack Family 8.4

The Family 8.4 is a further extension with one more symmetric fixed point which occurs with surprisingly high probability.

Fact 87 (Weak Keys Family 8.4, $d = 2^{-79}$, 4 KP for 8R). We define the Weak Keys Family 8.4 by keys such that there exists four symmetric fixed points A, B, C, D for the 8 rounds $\mathcal{E}()$. With the default GOST S-boxes this occurs with density of at least $d = 2^{-79}$.

For every key in Family 8.4, given 2^{32} CP we can obtain 4 P/C pairs for 8 rounds of GOST, correct with probability close to 1.

Justification: We get $d = 2^{-79}$ by applying Fact 82. The four pairs are then $\mathcal{E}(A) = A$, $\mathcal{E}(B) = B$, $\mathcal{E}(C) = C$ and $\mathcal{E}(D) = D$. Accordingly:

Fact 88 (Key Recovery for Weak Keys Family 8.4, $d = 2^{-79}$).
For the weak keys of Family 8.4 one can recover the key given 2^{32} CP, with running time of 2^{99} GOST encryptions and with negligible memory.

Justification: We have designed special dedicated version of Fact 125 from Section J.2. We obtain at most 2^{99} GOST computations J.3. The difference between Fact 125 in 2^{94} and 2^{99} from from Fact 126 is that in this case the points have many more solutions in common which make it harder to reject sets of key bits.

Again we have a conversion step with early rejection of non-weak keys:

Fact 89 (Family 8.4 Attack For a Population of 2^{79} Random Keys).
If we have a diverse population of at least 2^{79} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in total overall time of about $\mathbf{2^{101}}$ GOST encryptions.

Justification: We have an attack where given 2^{32} CP per device, namely the encryption of all symmetric plaintexts, the weak key become immediately "visible" for the attacker. The probability that a random 64-bit permutation has 4 symmetric fixed points is about 2^{-134}. This not counting additional differences on A, B, C, D which we also have in our attack which could be used to reject with even higher probability. Thus for all except the correct device out of 2^{79} we can reject this device right away after checking 2^{32} plaintexts in time of 2^{79+32} CPU clocks which is about 2^{101} GOST encryptions.

For the remaining one case which we expect to be really our weak key with correct additional differences we apply Fact 85 which gives about 2^{99} GOST encryptions and negligible memory. Overall we expect to recover one key in time of about 2^{101} GOST encryptions.

Remark: These two attacks will also appear in [34].

29 Summary: Comparison of Principal Known Attacks on GOST, Weak and Strong Keys

In Table 4 we compare our attacks with weak keys compared to selected other attacks with regular keys in such a way that we can compare them fairly. We also include most recent Differential-Reflection-Fixed Point Family 10 attack from Section 25 which is a particularly strong attack in the multiple key scenario.

For each weak-key attack the table shows TWO applications of it: one if we know that the key is weak, and another if we have a mix of weak and strong keys where weak keys occur with their natural probability.

Attack Ref.	§15.3/[50]	§18.1/[50]	Red. 3 §17	[39, 40]	F.0 [75]	Fam. 2	Fam. 2	Fam. 3	Fam. 4.X.
Keys density d	0.63		0.63	1	2^{-32}			2^{-64}	2^{-64}
Data/key 32R	2^{32} KP	2^{64} KP	2^{64} KP	2^{64} KP	2^{32} CP	2^{32} CC	2^{32} ACC	2^{64} KP	2^{32} CP/2^{64}
Obtained for 8R	**2** KP		**3** KP	-	1 KP	**3** KP	**4** KP		**2** KP
Valid w. prob.	2^{-96}	2^{-64}	2^{-64}	-	2^{-1}	2^{-64}	2^{-64}	2^{-1}	2^{-0}
Storage bytes	$2^{46}/2^{39}$	$2^{46}/2^{39}$	2^{67}	2^{70}	small			2^{67}	for data
♯ False positives	2^{128}		2^{128}		2^{192}	2^{64}	2^{-0}	2^{64}	2^{128}
Time for 8 R	$2^{127}/2^{128}$	$2^{127}/2^{128}$	2^{110}	-	2^{192}	2^{110}	$\mathbf{2^{94}}$	$\mathbf{2^{94}}$	2^{128}
Attack time 32 R	$\mathbf{2^{223}}/2^{224}$	$\mathbf{2^{191}}/2^{192}$	2^{206}	$\mathbf{2^{179}}$	2^{192}	$\mathbf{2^{174}}$	$\mathbf{2^{158}}$	$\mathbf{2^{95}}$	$\mathbf{2^{128}}$
Cost of 1 key, if	$2^{224}/2^{225}$	$2^{192}/2^{193}$	2^{207}	$\mathbf{2^{179}}$	$\mathbf{2^{193}}$	2^{206}	2^{190}	$\mathbf{2^{159}}$	$\geq \mathbf{2^{129}}$
key diversity $\geq$	single key attacks or for $> 50\%$ of keys				2^{32}				2^{65}
Data x keys	2^{33}	2^{64}	2^{65}	2^{64}	2^{64}				$2^{96\ /\ 128}$

Family cf.	Fam. 5.3	Fam. 5.4	Fam. 6	Fam. 7.2	Fam. 8.1	Fam. 8.2	Fam. 8.3	F. 8.4 / [34]
Keys density d	$\mathbf{2^{-52}}$	2^{-77}	2^{-84}	2^{-84}	2^{-98}	2^{-84}	2^{-70}	2^{-79}
Data/key 32R	2^{32} ACP	2^{32} ACP	2^{33}CPCC	2^{32} ACC	2^{32} CP	2^{32} CP	2^{32} CP	2^{32} CP
Obtained for 8R	**3** KP	**4** KP	**4** KP	**6** KP	**3** KP	**3** KP	**3** KP	**4** KP
Valid w. prob.	2^{-9}	2^{-9}	2^{-0}	2^{-4}	2^{-0}	2^{-0}	2^{-0}	2^{-0}
Storage bytes	small							
♯ False positives	?	small		0	2^{64}	$> 2^{64}$	?	small
Time for 8 R	$\mathbf{2^{110}}$	$\mathbf{2^{94}}$	$\mathbf{2^{94}}$	$\mathbf{2^{83}}$	$\mathbf{2^{110}}$	$\mathbf{2^{110}}$	$\mathbf{2^{120}}$	$\mathbf{2^{94}}$
Attack time 32 R	$\mathbf{2^{119}}$	$\mathbf{2^{102}}$	$\mathbf{2^{94}}$	$\mathbf{2^{87}}$	$\mathbf{2^{110}}$	$\mathbf{2^{110}}$	$\mathbf{2^{120}}$	$\mathbf{2^{94}}$
Cost of 1 key, if	$\mathbf{2^{139}}$	$\mathbf{2^{116}}$	$\mathbf{2^{117}}$	2^{146}	$\mathbf{2^{120}}$	$\mathbf{2^{110}}$	$\mathbf{2^{120}}$	$\mathbf{2^{101}}$
key diversity $\geq$	2^{52}	2^{75}	2^{84}	2^{84}	2^{98}	2^{84}	2^{70}	2^{79}
Data x keys	2^{84}	2^{107}	2^{121}	2^{116}	2^{130}	2^{116}	2^{102}	2^{111}

Table 4. Major attacks on full GOST cipher: single vs. multiple random keys scenario. Various attacks are here compared according to their capacity to find some keys when weak keys occur at random with their natural probability. In lower table we see that if we allow higher key diversity requirements and more data collected in total (for all keys), the overall time cost to recover one key, this **including** the cost to examine keys which are not weak, decreases down to 2^{101} and beats all known single key attacks.

A crucial point is that the multiple key scenario is stronger and much more versatile than the single key scenario. If we assume that keys are generated at random and weak key occur naturally, and if the computing power of the attacker is limited, it is cheaper and easier to break some keys with special properties first, rather than not to break any keys with a given computational effort.

29.1 Analysis: Which Attack on Full GOST is the Best? Why Weak Keys Are Important for Attacks on Regular Random Keys

In our research we have discovered that single key attacks are NOT the most practical attacks on GOST (cf. also [31]). The security of GOST decreases very substantially in the multiple key scenario, and we obtained several attacks which can find keys at a cost which is much lower than for the best single key attack. The price to pay for this is an access to a larger diversity and more data overall, but not always a lot of data per key: many attacks use only 2^{32} of data per key. This cost decreases with key diversity but it reaches a natural limit. We cannot hope to obtain an even better attack with 2^{96} or more different keys. This can be seen as follows: with at least 2^{32} data per key we need to spend at least 2^{96+32} steps in examining all the data for the keys for which the attack does not work, at the beginning we don't know for which keys the attack will work.

There are several interesting new results with only 2^{32} of data per key (but more data total) which were developed in late 2012. For example with 2^{32} of data per key and 2^{64} of data total, one of our Family 2.1 attacks allows to recover one full 256-bit key in time of 2^{190} GOST encryptions. This has never been achieved before September 2012. Now with 2^{32} of data per key and 2^{96} of data total, one can achieve as little as 2^{129} per key with recent Family 4.X attacks related to or resulting from the work of [77]. Furthermore with 2^{32} of data per key and 2^{128} of data total, one can achieve as little as 2^{101} per key with our Family 8.X and many other results below 2^{120}.

29.2 Lessons Learned

These recent results are very surprising and are due to the fact that, in the same way as in Fact 50 and in Fact 92, in many other attacks such as Family 0,4.X.,5.3,5.4,6,8.1,8.2, etc. the weak keys are "visible" when looking at the data, and the attacker can efficiently discard many keys which are not weak, without running the whole attack for these keys, but by looking only at the data for the [non-]existence of appropriate data points.

The initial idea of **Algebraic Complexity Reduction** was to identify a highly non-trivial property H of the values which occur inside the encryption process such that if H holds, the key can be efficiently recovered. However now we see that there is another way of great practical importance. It is more important that **if H does NOT hold**, the key can efficiently discarded. The second property seems harder to achieve because in arguably most realistic attacks which we have studied this rejection process is done more frequently than the case when H holds. It appears that the main problem in the cryptanalysis of GOST is about being to reject large sets which contain almost all GOST keys of type such as $2^{256} - 2^{192}$ of all keys, in one single relatively very fast step. Such a step does not yet yield any sort of exploitable information about they key and it is not easy to see how it helps to break the cipher. It exists in order to pre-select encryption instances for which special properties on internal data do hold, so that we can apply "key recovery techniques knowing H" in fewer cases in which they are more likely to succeed.

Part VII

Attacks With 128-bit Keys

30 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 [8]. Our attack uses a modified variant of the attack from [75] which we named Family 0, see Fact 30. 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 30.0.1 (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 6 (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 30: 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 with high probability of about 1/2.

Fact 90 (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.

30.1 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 91 (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 90. 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 [83], 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.

30.2 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 92 (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 90. 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 $X = 0 \ldots 32$.

31 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 also one of those weak variants explicitly discussed on page 603 of [8]. 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 93 (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 5. 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:

$$\mathcal{E} = \mathcal{S} \circ \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \tag{9}$$
$$\mathcal{E}^{-1} = \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F} \circ \mathcal{S} \tag{10}$$

   This is a product of two involutions.
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:

$$Y = \mathcal{E}(X)$$
$$\Updownarrow$$
$$\overline{X} = \mathcal{E}(\overline{Y}).$$

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 $\mathcal{G} \overset{def}{=} \mathcal{F}^{-1} \circ \mathcal{S} \circ \mathcal{F}$ which is an involution.
9. 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.
10. For every $k \geq 0$ the function $\mathcal{E}^k$ is a product of two involutions.
11. For every $k \geq 0$ the function $\mathcal{S} \circ \mathcal{E}^k$ is an involution and 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.
12. The whole encryption process is perfectly periodic provided that we "undo" the final "irregular swap" and we have:
$$Enc_k = \mathcal{S} \circ \mathcal{E}^4 \tag{11}$$
13. In particular, the encryption function Enc_k is an involution.
14. If the attacker has access to the encryption oracle, he can use it to decrypt any message.
$$Y = Enc_k(X)$$
$$\Updownarrow$$
$$X = Enc_k(Y).$$
15. Consequently the encryption function Enc_k can be distinguished from a random permutation in constant time.
16. For every k X lies on a cycle of length exactly k for $\mathcal{S} \circ Enc_k$ if and only if $\overline{X}$ also lies on cycle of length exactly k for $\mathcal{S} \circ Enc_k$. This can be also obtained using Fact 19 page 49.
17. X is a fixed point of $\mathcal{S} \circ Enc_k$ if and only if $\overline{X}$ is also a fixed point for $\mathcal{S} \circ Enc_k$.
18. 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.

31.1 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 significantly less data.

Reduction 7 (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 93. 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 $\overline{B} = \mathcal{F}(A)$ 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 7, will be used to recover keys for Family B.

Fact 94 (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 7 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 significantly less data. This type of improved attacks are possible, because $\mathcal{E}$ has another particularly interesting property.

31.2 On The Existence of Very Special Fixed Points for Family B

We have the following non-trivial fact:

Fact 95 (Special Symmetric Fixed Points for Family B). Given a GOST key in Family B chosen at random the first 8 rounds $\mathcal{F}$ have a symmetric fixed point with probability as large as about $2^{-0.7}$ this instead of a probability about 2^{-32} which we would expect 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 attack in Section 17.

31.3 Fixed Point and Multiple Reflection Attacks

The main reason why keys in Family B are particularly weak is that, following Fact 95, $\mathcal{E}$ 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 96 (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 95, that such symmetric fixed points exist for $\mathcal{E}$ with very high probability $\approx 63\%$ over all keys in Family B.

Fact 96 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 [75]) 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 30: 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 6. We don't study these attacks because they are not very fast and slower than our previous fixed point attack above (cf. Fact 94).

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 8 (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 9 (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 97 (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 significantly 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 [20, 85, 23, 24] 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.

31.4 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 [58, 7, 8].

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 98 (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 93.9. 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 $\mathcal{S}$ 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 95 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 93.8. 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 7 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.

31.5 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 10 (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 $T \approx 2^{32}$ 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 T is odd and $GCD(T, 4) = 1$. We have verified experimentally with random GOST keys from Family B and with many random starting points X that $T \approx 2^{32}$ is a reasonable assumption and that the probability that T 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 T and if T is odd, we have $GCD(T, 4) = 1$ and the cycle size for $\mathcal{E}^4$ is equal to T. 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 $d = 4^{-1} \bmod T$ 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 T is odd, but if T 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 $T = 2U$ and U odd, and the attacker will see a sub-cycle of length U, and can be mistaken to believe that $T = U$,

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

This Reduction 10 will be used to recover keys for GOST Family B given only 2^{32} chosen plaintexts. This is done as follows.

Fact 99 (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 10, 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 T 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.

31.6 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 100 (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 99).

Fact 101 (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 10, 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 T was odd). This in time of roughly 2^{32} operations, each time.

Here on the contrary to the last attack given in Fact 99, 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 T 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 T are odd, which requires on average 2^3 trials, and for each trial we apply Fact 100 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.

31.7 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	Regular	Family 8.1	Family 0'	Fam. B, Sec. 31 and also in [30]			
Reduction cf.	DC	Fact 76	Red. 6 / [30]	Red. 7	Red. 9	Reduction 10	
Attack	[39, 40]	Fact 77	Fact 90	Fact 94	Fact 97	Fact 99	Fact 101
The density d	1	2^{-98}	2^{-160}	2^{-129}	2^{-130}	2^{-129}	
From (data 32 R)	2^{64} KP	2^{32} CP	2^{32} CP	2^{64} KP	2^{32} CP	2^{33} CP	2^{35} CP
Obtained 8R	-	**3** KP	1	≥ 2	≥ 2	2^{32}	2^{33}
Selected 8R	-	**3** KP	-	1	2	1	3
Valid w. prob.	-	2^{0}	-	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^{70}	small	-	2^{67}	2^{35}	2^{36}	2^{38}
♯ False positives	large	2^{64}	2^{64}	2^{64}	small	2^{64}	2^{32}
Attack time 32 R	2^{179}	2^{120}	2^{65}	2^{90}	2^{91}	2^{89}	$\mathbf{2^{81}}$
Cost of 1 key, if	2^{179}	2^{120}	$\mathbf{2^{66}}$	2^{91}	2^{93}	2^{90}	$\mathbf{2^{83}}$
key diversity $\geq$	1	2^{98}	2^{32}	$2^{0.7}$	2^{2}	2^{1}	2^{2}

Table 6. 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 94) 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 99 and Fact 101) 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 97). Then the time complexity gets smaller in each case with a moderate increase in data complexity. This last improved attack of Fact 101 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 4 page 128.

We omit possible improvements to the second result (cf. Fact 97) by using also Fact 100, 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.

Note: Our principal (but not all) attacks on 128-bit keys also appear in [30].

Part VIII

How Secure Is GOST?

32 Should GOST Become An International Encryption Standard?

In 2010 GOST was submitted to ISO to become a worldwide encryption standard. Should it be standardized? There are two quite different points of view on this question.

From the cryptography research point of view we have broken GOST, and many algorithms have been rejected by various standardization bodies for significantly less than an actual key recovery attack faster than brute force.

However it does NOT 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. [83], 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. [101, 70]). Very few encryption algorithms have ever been standardized by ISO. The international standard ISO/IEC 18033-3 specifies the following algorithms. Four 64-bit block ciphers: TDEA, MISTY1, CAST-128, HIGHT, and three 128-bit block ciphers: AES, Camellia, SEED. Recently PRESENT was also added to the same standard.

To summarize, GOST can be used and the risks remain quite low however it is clear that ISO should not standardize GOST, as this algorithm is structurally flawed, and does not provide the very high security level required by ISO.

33 Conclusion

The Russian encryption standard GOST is implemented in OpenSSL and other crypto libraries [70, 105, 88], used by Russian banks, and increasingly also on the Internet. It appears that GOST has a lower gate count than any comparable cipher, cf. [83]. 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 [102, 71]. 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 [16, 13, 14]. However for block ciphers only a few rounds can be broken, see [19], and only one full-round real-life block cipher KeeLoq could so far be shown to be weak enough, to be broken using an algebraic attack [20]. 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? Can we ever hope to apply algebraic attacks to modern ciphers with many more rounds knowing that they may be able to deal with low quantities of data but only produce attacks with few rounds?

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 but leaves the attacker with extremely few data pairs for the reduced cipher. Very few low-data complexity attacks are known: mostly software "algebraic attacks" and meet in the middle attacks (MITM). In this paper and in [35] we present a dozen of highly competitive results obtained by combination of these techniques, cf. also [32]. For regular GOST keys and single key attacks we obtain five non-trivial black-box complexity reductions on full GOST which are summarized in Table 3 on page 53.

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 able to exploit similarities of individual sub-blocks and their inverses. We are the first to propose attacks with double triple and quadruple reflection. We are able to relax the conditions necessary in slide attacks [58, 7, 8, 6, 57] in nearly arbitrary ways. We use fixed points in innovative ways, different than in previous fixed point attacks (cf. [20, 75]). We present numerous attacks on GOST which don't use any reflections [74, 75, 77, 76] whatsoever. In Section 25 we combine all these with advanced differential attacks and in Section 26 we will exploit multi-point differential properties.

In this paper we present some 50 different new attacks on GOST which are faster than brute force in the single or the multiple key scenario and many additional attacks. The last step of these attacks have been very substantially improved in the last 2 years. Many very good results are now obtained through highly optimized strategies inspired by MITM attacks, a careful analysis of the internal connections inside the cipher, and software attacks with a SAT solvers. Several of our new attacks described in this paper achieve substantially lower memory requirements and faster running times than known MITM attacks, cf. [50, 35, 32], see Section 9 and Appendix 12.

For single key attacks, six of our attacks are shown in Table 3 on page 53 and are compared to other known attacks in Table 7 below. The fastest single key attack on GOST in this paper requires 2^{64} known plaintexts and 2^{191} GOST computations. The fastest single-key attack on GOST known is in 2^{179} cf. [39, 40].

Attack Ref.	Red.1 §15.1		§15.3/[50]	§18.1/[50]	Red. 3 §17	[39, 40]
Type	Internal Reflection		Refl+MITM	FP+MITM	2x Refl.	DC
Data	2^{32} KP		2^{32} KP	2^{64} KP	2^{64} KP	2^{64} KP
Memory bytes	2^{132}	small	$2^{46}/2^{39}$	$2^{46}/2^{39}$	2^{67}	2^{70}
Time	2^{224}	2^{227}	$\mathbf{2^{223}}/2^{224}$	$\mathbf{2^{191}}/2^{192}$	2^{206}	$\mathbf{2^{179}}$

Table 7. Principal single key attacks on GOST

33.1 Conclusion - Multiple Key Attacks

Ciphers are not used in practice with single keys, on the contrary. The multiple key scenario is how ciphers are used in practice. In order to evaluate the security of GOST in this scenario, we have done an extensive study of weak key classes in GOST which are made possible by our complexity reduction methodology. It is a very complex picture and our key results are summarized in Table 4 on page 128. We deliberately compare these attacks to single key attacks. 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. Many other attacks **beat best regular attack on running time** also when the keys are not weak. In order to achieve this, we consider a realistic scenario of encryption with a population of devices with random diversified keys, with weak keys occurring naturally. This allows compare all the attacks on one single scale. Thus we are able to break full GOST in overall **total** time of about 2^{159}, and more recently even $\mathbf{2^{101}}$, which includes checking all the keys and breaking one of the weaker keys (this attack also appears in [34]). These have now become the best attacks on GOST ever found in the general multiple key scenario. If the computing ressources are bounded, an attack which finds one key in time 2^{101} or less is infinitely better and more realistic than all single key attacks on GOST known: 2^{192} with 2^{64} of data by Shamir *et al* [50] and 2^{179} with 2^{64} of data by Courtois [39, 40].

In later 2012 there were many new attacks and important improvements to older attacks. Many attacks now require only 2^{32} of data per key, e.g. Section

22.1. Many recent attacks allows for early rejection of non-weak keys in the multiple key scenario which leads to many surprisingly realistic and powerful attacks. For example the recent attack on GOST proposed by Kara and Karakoç, cf. Section 23.2 and [77] can be extended to the multiple key scenario as we define it in this paper and achieves an attack with time of about 2^{13X} per key, 2^{96} data overall and only 2^{32} of data per key.

Additionally, in Section 25 we introduce a new family of differential/complexity reduction attacks on GOST which exploits differential cryptanalysis, reflection, fixed points and/or involution properties simultaneously, which creates additional degrees of freedom for the attacker and allows to enhance existing attacks. For example with Family 5.3 we obtain 2^{139} GOST encryptions per key with only 2^{84} of data total and only 2^{32} of data per key. Furthermore we discovered new forms of advanced combined self-similarity attacks. In Section 26 we introduced a new notion of **approximate fixed point biclique**, which is a single-key advanced invariant simultaneous truncated differential property which can be defined for 2,3,4 and more plaintexts. With these new techniques with Family 5.4 we obtain 2^{113} per key with 2^{107} of data total. Our Family 8.1 can be now modified to reduce the cost per key to 2^{111} with 2^{110} of data overall and only 2^{32} of data per key, and with Family 8.4. we can achieve even faster time of 2^{101} with 2^{111} of data cf. Fact 89 page 127 which attack also appears in [34]. Thus we obtain an attack with a total cost being **a remotely feasible total cost of 2^{101} GOST encryptions per key for 256-bit keys generated at random** given some 2^{111} of data overall and yet only 2^{32} of data per key.

In addition, in this paper we also present several attacks on some major reduced key size variants of GOST. Both “natural” methods of using GOST with 128-bit keys which were previously suggested in the literature [8] are shown to be **broken in practice**. With the “inversed” method we can recover arbitrary 128-bit keys within 2^{81} GOST encryptions and given 2^{35} CP, cf. Fact 101. 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 91 in Section 30. 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. Our results on 128-bit GOST are summarized in Table 6 on page 147.

Most of our attacks are expected to work for any choice of GOST S-boxes but some have been optimized in just one case. Since most of our attacks require 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 could have a very significant impact on decryption of messages encrypted with some weaker variants of GOST, if such variants of GOST are used in practice.

References

1. Martin Albrecht: *Algebraic Attacks against the Courtois Toy Cipher,* In Cryptologia, Vol. 32, Iss. 3 July 2008 , ppp. 220–276.
2. Martin Albrecht and Gregor Leander: *An All-In-One Approach to Differential Cryptanalysis for Small Block Ciphers,* preprint available at `eprint.iacr.org/2012/401/`.
3. Ludmila K. Babenko, Evgeniya Ishchukova, Ekaterina Maro: *Research about strength of GOST 28147-89 encryption algorithm,* In SIN 2012: pp. 138-142, ACM, 2011.
4. Ludmila K. Babenko, Evgeniya Ishchukova, Ekaterina Maro: *Algebraic analysis of GOST encryption algorithm,* In SIN 2011, pp. 57-62, ACM, 2011.
5. Lyudmila K. Babenko, Evgeniya Ishchukova: *Differential analysis of GOST encryption algorithm,* In SIN 2010, pp. 149-157, ACM, 2010.
6. Eli Biham, Orr Dunkelman, Nathan Keller: *Improved Slide Attacks,* In FSE 2007, LNCS 4593 Springer 2007, pp. 153-166.
7. A. Biryukov, D.Wagner: *Slide Attacks,* In proceedings of FSE'99, LNCS 1636, pp. 245-259, Springer, 1999.
8. Alex Biryukov, David Wagner: *Advanced Slide Attacks,* In Eurocrypt 2000, LNCS 1807, pp. 589-606, Springer 2000.
9. Alex Biryukov: *Analysis of Involutional Ciphers: Khazad And Anubis,* In FSE 2003, pp. 45-53 LNCS.
10. Eli Biham, Adi Shamir, *Differential Cryptanalysis of DES-like Cryptosystems,* Journal of Cryptology, vol. 4, pp. 3-72, IACR, 1991.
11. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger: *Biclique cryptanalysis of the full AES,* In Asiacrypt 2011, LNCS 7073, pp. 344-371, 2011.
12. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
13. 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`
14. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
15. Nicolas Courtois *CTC2 and Fast Algebraic Attacks on Block Ciphers Revisited* Available at `http://eprint.iacr.org/2007/152/`.
16. 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.
17. 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/`.
18. Nicolas Courtois: *100 years of Cryptanalysis: Compositions of Permutations* slides about cryptanalysis of Engima and block cipher cryptanalysis, used teaching GA18 Cryptanalysis course at University College London 2014-2016, `http://www.nicolascourtois.com/papers/code_breakers_enigma_block_teach.pdf`
19. 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/`.
20. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.

21. Nicolas Courtois, Gregory V. Bard and Andrey Bogdanov: *Periodic Ciphers with Small Blocks and Cryptanalysis of KeeLoq,* In Tatra Mountains Mathematic Publications, 41 (2008), pp. 167-188, post-proceedings of Tatracrypt 2007 conference, The 7th Central European Conference on Cryptology, June 22-24, 2007, Smolenice, Slovakia.
22. Nicolas Courtois: *Self-similarity Attacks on Block Ciphers and Application to KeeLoq,* In Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday. LNCS 6805, Springer, 2012, pp. 55-66, David Naccache editor.
23. Nicolas T. Courtois and Gregory V. Bard: *Random Permutation Statistics and An Improved Slide-Determine Attack on KeeLoq,* In Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday. LNCS vol. 6805, Springer, 2012, pp. 35-54, David Naccache editor.
24. Gregory V. Bard, Shaun V. Ault and Nicolas T. Courtois: *Statistics of Random Permutations and the Cryptanalysis Of Periodic Block Ciphers,* In Cryptologia, Vol. 36, Issue 03, pp. 240-262, July 2012.
25. 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.
26. Nicolas T. Courtois, Pouyan Sepherdad, Petr Susil and Serge Vaudenay: *ElimLin Algorithm Revisited,* In FSE 2012, LNCS, Springer.
27. 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/`.
28. Nicolas Courtois: *Cryptanalysis of GOST,* a very long extended sets of slides about the cryptanalysis of GOST, 2010-2014, `http://www.nicolascourtois.com/papers/GOST.pdf`. An earlier and shorter version was presented at 29C3, see [29].
29. Nicolas Courtois: *Cryptanalysis of GOST,* (Security Evaluation of Russian GOST Cipher; Survey of All Known Attacks on Russian Government Encryption Standard.) Presentation at 29th Chaos Communication Congress (29C3), December 27th to 30th, 2012, Hamburg, Germany, `http://events.ccc.de/congress/2012/Fahrplan/attachments/2243_GOST_29C3_long.pdf`
A video is available on: `www.youtube.com/watch?v=o_sP0qJam-4`
An MP3 audio recording is available at: `http://blademp3.com/mp3/4709a02_Security-Evaluation-of-Russian-GOST-Cipher.html`.
30. Nicolas Courtois: *Cryptanalysis of Two GOST Variants With 128-bit Keys,* In Cryptologia vol. 38(4), pp. 348-361, 2014. At `http://www.tandfonline.com/doi/full/10.1080/01611194.2014.915706`.
31. 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`.
32. Nicolas Courtois, Jerzy A. Gawinecki, Guangyan Song: *Contradiction Immunity and Guess-Then-Determine Attacks On GOST,* In Tatra Mountains Mathematic Publications, Vol. 53 no. 3 (2012), pp. 65-79.
33. Nicolas T. Courtois: *Cryptanalysis of GOST In the Multiple Key Scenario,* In post-proceedings of CECC 2013, Tatra Mountains Mathematical Publications. Vol. 57, no. 4 (2013), p. 45-63. At `http://www.sav.sk/journals/uploads/0124133006Courto.pdf`

34. Nicolas Courtois: *On Multiple Symmetric Fixed Points in GOST,* In Cryptologia, Volume 39, Issue 4, 2015, pp. 322-334, `http://www.tandfonline.com/doi/full/10.1080/01611194.2014.988362`.
35. Nicolas T. Courtois: *Low-Complexity Key Recovery Attacks on GOST Block Cipher,* In Cryptologia, Volume 37, Issue 1, pp. 1-10, 2013.
36. Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* In CECC 2011, 11th Central European Conference on Cryptology. In Periodica Mathematica Hungarica Vol. 65 (2), 2012, pp. 1126, Springer.
37. Nicolas Courtois, Michał Misztal: *First Differential Attack On Full 32-Round GOST,* in ICICS'11, pp. 216-227, Springer LNCS 7043, 2011.
38. Nicolas Courtois, Michał Misztal: *Differential Cryptanalysis of GOST,* In Cryptology ePrint Archive, Report 2011/312. 14 June 2011, `http://eprint.iacr.org/2011/312`.
39. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* in "The New Codebreakers a Festschrift for David Kahn", LNCS 9100, Springer, 2015.
40. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* In Cryptology ePrint Archive, Report 2012/138. 15 March 2012, updated September 2015, `http://eprint.iacr.org/2012/138`.
41. Nicolas Courtois, Theodosis Mourouzis, Anna Grocholewska-Czurylo and Jean-Jacques Quisquater: *On Optimal Size in Truncated Differential Attacks,* In CECC 2014, Budapest, Hungary, 21 - 23 May 2014. Slides presented: `http://www.nicolascourtois.com/papers/GOST_CECC2014.pdf`. Post-proceedings in print (Studia Scientiarum Mathematicarum Hungarica).
42. Nicolas T. Courtois, Theodosis Mourouzis, Michał Misztal, Jean-Jacques Quisquater, Guangyan Song: *Can GOST Be Made Secure Against Differential Cryptanalysis?,* In Cryptologia, vol. 39, Iss. 2, 2015, pp. 145-156.
43. Nicolas T. Courtois, Theodosis Mourouzis: *Advanced Differential Cryptanalysis and GOST Cipher,* accepted for a 30 minute oral presentation at the 3rd IMA Conference on Mathematics in Defence At Tom Elliott Conference Centre, QinetiQ, Malvern, UK on Thursday 24 October 2013. 6-pages paper in CD-ROM and web proceedings planned.
44. Nicolas T. Courtois, Theodosis Mourouzis: *Enhanced Truncated Differential Cryptanalysis of GOST,* in SECRYPT 2013, Reykjavik, July 2013, `http://www.nicolascourtois.com/papers/sec13.pdf`
45. Nicolas T. Courtois, Theodosis Mourouzis: *Propagation of Truncated Differentials in GOST,* in proc. of SECURWARE 2013, `http://www.thinkmind.org/download.php?articleid=securware_2013_7_20_30119`
46. Nicolas T. Courtois, Daniel Hulme and Theodosis Mourouzis: *Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis,* In (informal) proceedings of SHARCS 2012 workshop, pp. 179-191, `http://2012.sharcs.org/record.pdf`. Earlier preprint is available at, `http://eprint.iacr.org/2011/475`, and an abridged version appears in the electronic proceedings of the 2nd IMA conference Mathematics in Defence 2011, UK.
47. 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.
48. 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/`.

49. Gustaf Dellkrantz: *Cryptanalysis of Symmetric Block Ciphers, Breaking Reduced KHAZAD and SAFER++*, Royal Institute of Technology, Sweden, supervised by Johan Håstad and Christophe De Cannière, `http://www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2003/rapporter03/dellkrantz_gustaf_03110.pdf`
50. Itai Dinur, Orr Dunkelman and Adi Shamir: *Improved Attacks on Full GOST*, FSE 2012, LNCS 7549, pp. 9-28, 2012, early version available at `http://eprint.iacr.org/2011/558/`.
51. Itai Dinur, Orr Dunkelman, Nathan Keller and Adi Shamir: *Reflections on Slide with a Twist Attacks*, 16 Oct 2014, At `https://eprint.iacr.org/2014/847`
52. Ali Doganaksoy, Bariş Ege, Onur Koçak and Fatih Sulak: *Cryptographic Randomness Testing of Block Ciphers and Hash Functions,* In `http://eprint.iacr.org/2010/564`.
53. 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.
54. Philippe Flajolet, Robert Sedgewick *Analytic Combinatorics* , Cambridge University Press.
55. I. J. Good and Cipher A. Deavours, *Afterword to: Marian Rejewski, "How Polish Mathematicians Deciphered the Enigma",* Annals of the History of Computing, 3 (3), July 1981, 229-232.
56. 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.
57. Soichi Furuya: *Slide Attacks with a Known-Plaintext Cryptanalysis,* In ICISC 2001, LNCS 2288, 2002, pp. 11-50.
58. 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.
59. L. V. Kovalchuk: *Upper-bound estimation of the average probabilities of integer-valued differentials in the composition of key adder, substitution block, and shift operator,* In Cybernetics And Systems Analysis Vol. 46, Number 6 (2010), pp. 936-944, Springer.
60. L. V. Kovalchuk and O. A. Sirenko: *Analysis of mixing properties of the operations of modular addition and bitwise addition defined on one carrier,* In Cybernetics And Systems Analysis Vol. 47, Number 5 (2011), pp. 741-753, Springer.
61. A. N. Alekseychuk and L. V. Kovalchuk: *Towards a Theory of Security Evaluation for GOST-like Ciphers against Differential and Linear Cryptanalysis,* Preprint 9 Sep 2011, `http://eprint.iacr.org/2011/489`.
62. V.V. Shorin, V.V. Jelezniakov, E.M. Gabidulin *Security of algorithm GOST 28147-89,* (in Russian), In Abstracts of XLIII MIPT Science Conference, December 8-9, 2000.
63. 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
64. 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 [65].

65. An English translation of [64] by Aleksandr Malchik with an English Preface co-written with Whitfield Diffie, was published in 1994, at `193.166.3.2/pub/crypt/cryptography/papers/gost/russian-des-preface.ps.gz`
66. 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`
67. GOST R 34.11-94, the Russian hash function standard, the original Russian version can be found at `http://protect.gost.ru/document.aspx?control=7&id=134550` and an English transation can be found at `ftp.funet.fi/pub/crypt/cryptography/papers/gost/russian-des-preface.ps.gz`.
68. Vasily Dolmatov, Editor, RFC 5831: *GOST R 34.11-94: Hash Function Algorithm,* IETF. ISSN: 2070-1721. March 2010. `http://tools.ietf.org/html/rfc5831`.
69. V. Popov, I. Kurepkin, S. Leontie: *RFC 4357: Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms,* IETF January 2006. `http://tools.ietf.org/html/rfc4357`
70. 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/`
71. J. Hulsbosch: *Analyse van de zwakheden van het DES-algoritme door middel van formele codering,* Master thesis, K. U. Leuven, Belgium, 1982.
72. Florian Mendel, NorbertPramstaller and Christian Rechberger: *A (Second) Preimage Attack on the GOST Hash Function,* In Kaisa Nyberg editor, FSE 2008, LNCS 5086, pp. 224234, Springer, 2008.
73. 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.
74. Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011.
75. Orhun Kara: *Reflection Cryptanalysis of Some Ciphers,* In Indocrypt 2008, LNCS 5365, pp. 294-307, 2008.
76. Jialin Huang and Xuejia Lai: *What is the Effective Key Length for a Block Cipher: an Attack on Every Block Cipher,* `eprint.iacr.org/2012/677`.
77. Orhun Kara and Ferhat Karakoç: *Fixed Points of Special Type and Cryptanalysis of Full GOST.* In CANS 2012, LNCS 7712, pp 86-97, 2012.
78. 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.
79. Lars R. Knudsen: *Truncated and Higher Order Differentials,* In FSE 1994, pp. 196-211, LNCS 1008, Springer.
80. Nick and Alex Moldovyan: *Innovative Cryptography,* textbook, 2nd edition, Charles River Media, Boston, 2007.
81. Theodosis Mourozis: *Optimizations in Algebraic and Differential Cryptanalysis,* PhD thesis, under superivsion of Dr. Nicolas T. Courtois, University College London, January 2015, `http://discovery.ucl.ac.uk/1462141/2/PhD_Thesis_Theodosis_Mourouzis.pdf`
82. Klaus Pommerening: *Permutations and Rejewskis Theorem,* `http://www.staff.uni-mainz.de/pommeren/MathMisc/Permut.pdf`
83. 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.

84. 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.
85. Random Permutation Statistics – wikipedia article, 22 January 2008, available at `http://en.wikipedia.org/wiki/Random~permutation~statistics`.
86. J.-J. Quisquater and J.P. Delescaille: *How Easy is Collision Search. New Results and Applications to DES,* In Crypto89, LNCS 435, pp. 408-413.
87. 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.
88. RSA Labs PKCS #11: Cryptographic Token Interface Standard, ver. 2.30, Sep 2009, mechanisms part 1, Sections 6.39-6.40.7`ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30m1-d7.pdf`
89. Marian Rejewski: *How Polish Mathematicians Deciphered the Enigma,* Annals of the History of Computing, vol. 3, number 3, July 1981, 213-234.
90. Marian Rejewski: *Mathematical Solution of the Enigma Cipher,* In Cryptologia, vol. 6, number 1, January 1982, pp. 1-37.
91. Marian Rejewski. An application of the theory of permutations in breaking the Enigma cipher. Applicaciones Mathematicae, 16(4), Warsaw, 1980. At `http://www.impan.pl/Great/Rejewski/article.html`
92. Marian Rejewski: *Memories of My Work at the Cipher Bureau of the General Staff Second Department 1930-45,* second edition, Adam Mickiewicz University Press, Poznan, Poland, 2011.
93. Frank Carter: *The First Breaking of Enigma: Some of the Pioneering Techniques Developed by the Polish Cipher Bureau* Report No 2, Bletchley Park Trust, new edition September 2008.
94. 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`
95. Vladimir Rudskoy, Andrey Dmukh: *Algebraic and Differential Cryptanalysis of GOST: Fact or Fiction,* In CTCrypt 2012, Workshop on Current Trends in Cryptology, affiliated with 7th International Computer Science Symposium in Russia (CSR-2012), 2 July 2012, Nizhny Novgorod, Russia. Full papers will be submitted and published in a special issue of Russian peer-review journal Mathematical Aspects of Cryptography. An extended abstract is available at: `https://www.tc26.ru/invite/spisokdoc/CTCrypt_rudskoy.pdf` slides are available at: `https://www.tc26.ru/documentary%20materials/CTCrypt%202012/slides/CTCrypt_rudskoy_slides_final.pdf`
96. Vladimir Rudskoy and Andrey Chmora: *Working draft for ISO/IEC 1st WD of Amd1/18033-3: Russian Block Cipher GOST, ISO/IEC JTC 1/SC 27 N9423, 2011-01-14, MD5=feb236fe6d3a79a02ad666edfe7039aa*
97. Igor Semaev: Sparse Algebraic Equations over Finite Fields, SIAM J. Comput. 39(2): 388-409 (2009).
98. 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/`
99. Markku-Juhani Saarinen: *A chosen key attack against the secret S-boxes of GOST,* unpublished manuscript, 1998.
100. Haruki Seki and Toshinobu Kaneko: *Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, LNCS 2012, pp. 315-323, Springer, 2000.*

101. Bruce Schneier: Section 14.1 GOST, in *Applied Cryptography,* Second Edition, John Wiley and Sons, 1996. ISBN 0-471-11709-9.

102. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in particular page 704.

103. Niklas Sörensson, Niklas Eén: MiniSat 2.06. an open-source SAT solver package.

104. Mate Soos: CryptoMiniSat 2.92, an open-source SAT solver package based on earlier MiniSat software, at `http://www.msoos.org/cryptominisat2/`

105. Wei Dai: Crypto++, a public domain library containing a reference C++ implementation of GOST and test vectors, `http://www.cryptopp.com`

106. Pavol Zajac: *Solving Trivium-based Boolean Equations Using the Method of Syllogisms,* Fundam. Inform. 114(3-4): 359-373 (2012)

107. Pavol Zajac, Radoslav Cagala: *Local reduction and the algebraic cryptanalysis of the block cipher gost.* In Periodica Mathematica Hungarica 65(2): 239-255 (2012).

108. Marcel Zanechal: *An algebraic approach to fix points of GOST-algorithm,* Mathematica Slovaca 51 (2001), no. 5, 583-591.

109. Otokar Grosek, Pavol Zajac: Two papers in Encyclopedia of Artificial Intelligence *Automated Cryptanalysis,* on pages 179-185 and *Automated Cryptanalysis of Classical Ciphers,* pages 186-191.
Rabuñal, Dorado, Pazos (Eds.), 3 Volumes, IGI Global 2009, ISBN 9781599048499

110. 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/`, 17 Feb 2012, the initial attack was apparently incorrect and later versions of this paper do NOT study GOST cipher at all.

Part IX

Additional High-Level Attacks

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

In this paper we introduce many 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 on of the very few attacks in this paper which **do 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 [20], 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 [27].

We consider texts D such that $\mathcal{E}(D) = \overline{D}$. Then we just look a few steps backwards:

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 102 (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 103 (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. 36.
2. $Enc_k(B) = C$ This can be seen on the left hand side of Fig. 36.

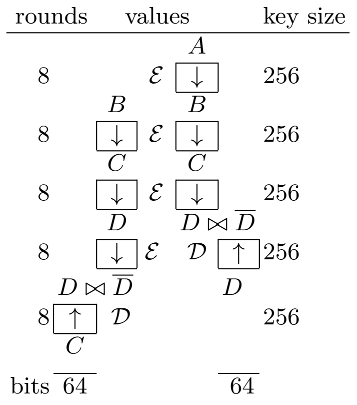


Fig. 36. An alternative attack with reduction to 4 pairs and no internal reflection

This leads directly to our new reduction:

Reduction 11. [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 in Fig. 36. 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 11 to Break GOST

Both our previous Reduction 4 and this new Reduction 11 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 17 with additional guess of D, as described in Section 17.1 which is also summarized in the next to last column in Table 3 and the complexity is exactly the same: 2^{222} GOST encryptions.

Summary. Overall our attack requires 2^{64} known plaintexts, time is 2^{33} 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 17 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 17.1 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 35. 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 39.

A.3 Generalizations of Reduction 11

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.

A.4 One More Reduction To 4 KP Without Internal Reflections

There is also another method to get the same result as in Reduction 11,

We combine Reduction 5 to get 2 pairs for 8 rounds, then we use the amplification property of Fact 105 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.

B 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 A.4 and Reduction 11.

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 17.1 and the next to last column in Table 3). 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 12. [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 [20, 85, 23, 24]. 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 7 and compute the key in time of 2^{94} GOST encryptions.

Fact 104. 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.

B.1 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 B which gives in some cases identical, and in some cases different cases (there is a non-trivial intersection of both attacks).

Reduction 13. [From 2^{64} KP for 32 Rounds to 4 KP for 8 Rounds]
We assume that $\mathcal{E}$ has a cycle of length 2 which occurs with probability 50%. 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 $\mathcal{E}^2$. It is easy to see that points of order two for $\mathcal{E}$ come in pairs and the expected number of cycles of length 2 is 1/2 for $\mathcal{E}$, cf. [20, 85, 23, 24]. 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 in Fig. 37. 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:
$X \mapsto Y, Y \mapsto X, Z \mapsto \overline{Y}, T \mapsto \overline{X}$.

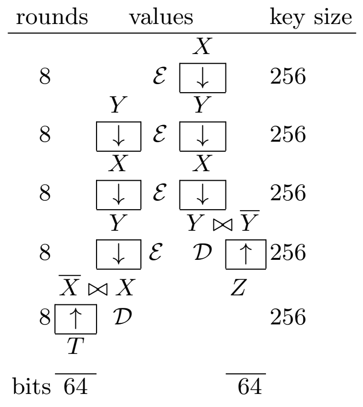


Fig. 37. A slightly cheaper alternative attack with no internal reflection

Resulting Attack. Again, if we combine this with Fact 6 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 104). The storage required is for the 2^{64} known P/C pairs.

C 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 many the attacks described in this paper and reminiscent of slide attacks [58, 7] 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 [58, 7, 8] 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 [20], 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 [58, 7, 8] is deeply broken by the inversion of keys which occurs in the last 8 rounds. However an analogous property for GOST still exists.

Fact 105 (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 $T = Dec_k(Y)$.

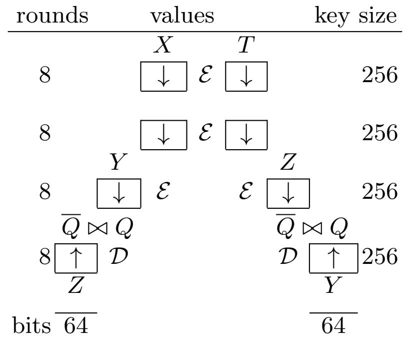


Fig. 38. Involution property: one-time amplification at zero cost

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

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

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.

General Method More generally, this amplification method can be applied to any combination of type $\mathcal{I} \cdot \mathcal{P}$ where $\mathcal{I}$ is an involution, cf. later Fig. 39.

C.1 Additional Remarks - Can Amplification Be Iterated?

In Fact 105, 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 105).

C.2 Can This Fail - The Guessing Paradox

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 the method of Fact 105 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 105 is guaranteed to work.

This leads to the following observation which is sort of paradox, what we gain on one side, we lose on another.

Consider $\mathcal{E}^2$ which is the fist 16 rounds of GOST. Normally, a random pair X, Y is valid $\mathcal{E}^2$ for with probability 2^{-64}, i.e. $\mathcal{E}^2(X) = Y$ with probability 2^{-64}. Such pairs can be amplified, i.e. another pair will be obtained with high probability close to 1 (in fact to be precise it is equal to $1 - 2^{-32}$).

Now there is also a method for generating pairs X, Y which are valid with probability 2^{-32} i.e. $\mathcal{E}^2(X) = Y$ can be guessed and are true with probability 2^{-32}. For this we need consider pairs $X, Enc_k(X)$ for any X. It is easy to see that such pairs are valid with probability 2^{-32}, which is because the last 16 rounds have as many as 2^{32} fixed points. However, bad luck, these pairs cannot be amplified, as explained above.

C.3 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.

C.4 The General Amplification Property

Fact 105 is not the only possible amplification property inside GOST. There are many other cases where an amplification property can exist. Amplification can basically occur each time we have a permutation of type $\mathcal{I} \cdot \mathcal{P}$ where $\mathcal{I}$ is an involution and $\mathcal{P}$ is an arbitrary permutation and such that the adversary has access to $\mathcal{I} \cdot \mathcal{P}$.

More precisely, let $Ac()$ be the oracle access to $\mathcal{I} \cdot \mathcal{P}$. Then from one pair X, Y for $\mathcal{P}$ we can obtain by using the encryption and decryption oracles another par for $\mathcal{P}$ which will be $Ac^{-1}(Y), Ac(X)$ which is illustrated on the following picture:

$$\begin{tikzcd}
X \arrow[d, "\mathcal{P}"] & T = Ac^{-1}(Y) \arrow[d, "\mathcal{P}"'] \\
Y \arrow[d, leftrightarrow, "\mathcal{I}"] & Z \arrow[d, leftrightarrow, "\mathcal{I}"'] \\
Z = Ac(X) & Y
\end{tikzcd}$$

Fig. 39. General framework for amplification for $\mathcal{I} \cdot \mathcal{P}$

Now we are going to show two new ways to decompose GOST in order to obtain involutions which will lead to two new amplification properties in later sub-section C.6.

C.5 Additional Involutions in GOST

Now let $\mathcal{A}$ be the first 4 rounds of GOST, $\mathcal{B}$ be the next 4 rounds, and the full GOST can be written as:

$$Enc_k = \mathcal{A}^{-1} \circ \mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B} \circ \mathcal{A} \circ \mathcal{B} \circ \mathcal{A} \circ \mathcal{B} \circ \mathcal{A} \tag{12}$$

We recall that $\mathcal{E} = \mathcal{B} \circ \mathcal{A}$ is the first 8 rounds of GOST and let $\mathcal{E}' = \mathcal{A} \circ \mathcal{B}$.

There are three interesting ways of writing Enc_k in which what is between two parentheses will be an involution:

$$Enc_k = \left(\mathcal{A}^{-1} \circ \mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B} \circ \mathcal{A}\right) \circ \mathcal{E}^2 \tag{13}$$

and also

$$Enc_k = \mathcal{A}^{-1} \circ \left(\mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B}\right) \circ \mathcal{E}'^2 \circ \mathcal{A} \tag{14}$$

and also

$$Enc_k = \mathcal{B} \circ \left(\mathcal{B}^{-1} \circ \mathcal{A}^{-1} \circ \mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B} \circ \mathcal{A} \circ \mathcal{B}\right) \circ \mathcal{E}'^2 \circ \mathcal{B}^{-1} \tag{15}$$

These properties are exploited in Section C.6 below.

C.6 Additional Amplification Properties

For example imagine that we guess the 128-bit key in the first 4 rounds of GOST $\mathcal{A}$. Then the attacker has oracle access to the the following permutation:

$$\mathcal{A} \circ Enc_k \circ \mathcal{A}^{-1} = \left(\mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B}\right) \circ \mathcal{E}'^2 \tag{16}$$

Or alternatively, imagine that we guess the 128-bit key in the second 4 rounds of GOST $\mathcal{B}$. Then the attacker has oracle access to the the following permutation:

$$\mathcal{B}^{-1} \circ Enc_k \circ \mathcal{B} = \left(\mathcal{B}^{-1} \circ \mathcal{A}^{-1} \circ \mathcal{B}^{-1} \circ \mathcal{S} \circ \mathcal{B} \circ \mathcal{A} \circ \mathcal{B}\right) \circ \mathcal{E}'^2 \tag{17}$$

These two formulas can be easily obtained from the last two formulas in the previous section. In both cases, given a partial key guess we obtain a permutation of type $\mathcal{I} \cdot \mathcal{P}$ where where $\mathcal{I}$ is an involution. Consequently we have following two amplification properties which can be obtained in the same way as in Fig. 39 above.

Fact 106 (Amplification Property With Guessed First Half-Key).
If the attacker guesses the 128-bit key in $\mathcal{A}$, for each P/C pair for the 16 rounds of GOST $(\mathcal{A} \circ \mathcal{B})^2$ which is $\mathcal{E}'^2$, he can generate another such pair.

More precisely, for any X, Y we have:

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

$$\Updownarrow$$

$$\mathcal{A}(Enc_k(\mathcal{A}^{-1}(X))) = \mathcal{E}'^2\left(\mathcal{A}(Dec_k(\mathcal{A}^{-1}(Y)))\right).$$

We also have:

Fact 107 (Amplification Property With Guessed Second Half-Key).
If the attacker guesses the 128-bit key in $\mathcal{B}$, for each P/C pair for the 16 rounds of GOST $(\mathcal{A} \circ \mathcal{B})^2$ which is $\mathcal{E}'^2$, he can generate another such pair.

More precisely, for any X, Y we have:

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

$$\Updownarrow$$

$$\mathcal{B}^{-1}(Enc_k(\mathcal{B}(X))) = \mathcal{E}'^2\left(\mathcal{B}^{-1}(Dec_k(\mathcal{B}(Y)))\right).$$

Remark. It is furthermore trivial to characterize exactly when this second pair is distinct from the first pair, which will be as in Fact 105 if and only if we avoid the fixed points of respectively $\mathcal{A}(Enc_k(\mathcal{A}^{-1}(X)))$ and $\mathcal{B}^{-1}(Enc_k(\mathcal{B}(X)))$.

Applications. It is not clear if the combination of these properties could lead to some sort of meet-in-the-middle attack on GOST keys.

D 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?

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

One such reduction is already present in Step 1. in Table 3 and underlies all the attacks which are summarized in this table. For completeness, we recall this reduction in its basic form:

Reduction 14. [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 as shown in Fig. 7 in Section 15.1, which is also used in Reduction 2 in Section 16. 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 105. 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.

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

By Applying Reduction 14 twice we immediately obtain that:

Reduction 15. [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}.

D.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 105.

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

However this method as described here seems less interesting that Reduction 15 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 15.

Reduction 16. [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 105 we have the following (not totally obvious) result:

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

D.4 Potential Applications

Under certain conditions, our Reduction 15 as well as Reduction 16 could allow attacks faster than brute force on the full 32-round GOST:

Fact 108 (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 15 or Reduction 16 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.

E 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 choose 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.

E.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 14 and Reduction 15.

Reduction 17. [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 17 twice and exactly in the same way as before, twice we immediately obtain that:

Reduction 18. [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.

F Approximate Reflection in GOST

It is possible to show the following property:

Fact 109 (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}. Moreover we need to compare the result obtained not to random permutations, to a random involution.

The bar is very high because every involution will have approximate reflections of some sort. For example if we call Twin Points pairs of points A, A' such that they differ by just 1 bit and $\mathcal{E}(A) = A'$ and $\mathcal{E}(A') = A$, it is possible to see that for any involution on n bits there exists on average $0.63 \cdot n$ Twin Points.

G 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:

$$Enc_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E} \tag{18}$$

Fact 110 (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.

G.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 C.3 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.

G.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 31, it is possible to see that (cf. Fact 93) 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 this paper we precisely study several such families of weak keys in GOST.

Another straightforward application of this property is given in Appendix H.2 below.

Part X

Additional High-Level Attacks With Differential Properties

H Combined Differential Complexity Reduction Attacks With A 16 Rounds Property

This is a continuation of Section 25 which deals with a combination of advanced truncated differential attacks and complexity reduction attacks. Here we work with differential properties for 16 rounds which are harder to find than for 8 rounds.

H.1 A Differential Property For 16 Rounds of GOST

The situation we are going to study is very similar to what happens for a different number of rounds inside the so called Alpha property in Section 4.1 of [40].

We recall that when we speak of a non-zero input difference of the following type $(0xF0000787, 0x80780000)$ we mean that all $2^{16}-1$ differences with active bits under $(0xF0000787, 0x80780000)$ are allowed. All the rounds have the final swap in Fig. 40.

```
0xF0000787 0x80780000
      (1 Round)
0x80780000 0x00000700
      (7 Rounds)
0x80700700 0x80700700
      (7 Rounds)
0x00000700 0x80780000
      (1 Round)
0x80780000 0xF0000787
```

Fig. 40. Symmetric event with asymmetric differences with 16/8/14 active bits and 16 rounds

Fact 111. We look at the combination of any non-zero input difference of type $(0xF0000787, 0x80780000)$ and any non-zero output difference of type $(0x80780000, 0xF0000787)$. For a typical permutation on 64-bits (can be a random permutation, or full GOST with 32 rounds) we expect that there are 2^{31} pairs P_i, P_j with such differences.
For 16 rounds of GOST and for a given random GOST key, there exist about $2^{31}+2^{32}$ pairs with such differences. Inside these, 2^{32} pairs are due to "propagation" which can be defined in this cases as having the difference of type $(0x807007000x80700700)$ in the middle after 8 rounds which event is unlikely to happen for any of the other 2^{31} pairs.

Justification: For a random permutation on 64 bits, as in Section 4 of [40] we observe that a random couple of input and output differences of 64+64 bits occurs 0.5 times on average as we have about 2^{127} pairs and only about 2^{128} possible sets of two differentials. Now we have $(2^{16}-1)(2^{16}-1) \approx 2^{32}$ possibilities here and on average 2^{31} will be realized for a given (random) permutation.

We are going to make a conservative estimation assuming that the propagation follows a certain pattern and exhibit 2^{32} pairs which will follow the pattern.

The number of additional cases which do not follow the pattern is expected to be 2^{31} as above which will give a total of $2^{31} + 2^{32}$ pairs.

We look at Fig. 40 and start from the middle of this figure after 8 rounds. There are $2^{64+14-1} = 2^{77}$ pairs with a difference being $(0x80700700, 0x80700700)$ in the very middle of the 16 rounds.

Then following Fact 2.3.2. of [40] the set $(0x80700700, 0x80700700)$ produces a differential of the form $(0x00000700, 0x80780000)$ with probability of $2^{-22.19}$ for the previous 7 rounds of GOST going backwards from the middle difference.

In the same way, the middle difference of $(0x80700700, 0x80700700)$ propagates with probability of $2^{-22.19}$ for the previous 7 rounds of GOST and gives also $(0x00000700, 0x80780000)$ going forward from the middle.

Overall we expect that $2^{77-22.2-22.2} = 2^{32.6}$ pairs are such that the differences for the 14 middle rounds are such as in Fig. 40. Then it is very easy to see that the propagation for the first 1 round backwards occurs with a very high probability of $2^{0.26}$. Overall we expect that $2^{32.6-0.4-0.4} \approx 2^{32}$ pairs with all the differences being as in Fig. 40.

H.2 An Internal Correlation Attack on GOST

Following Fact 111 on page 183 for each GOST key there exist about 2^{32} pairs P, Q which have all the differences shown in Fig. 40 page 183, which we reproduce here in a different version with a final Feistel swap omitted, so that we represent here the function $\mathcal{S} \circ \mathcal{E}^2$ and our property becomes iterative, see Fig. 41 below. We also show the order of round keys.

```
                    0xF0000787 0x80780000
k_0                       (1 Round)
                    0x80780000 0x00000700
k_1-7                     (7 Rounds)
                    0x80700700 0x80700700
k_0-6                     (7 Rounds)
                    0x00000700 0x80780000
k_7                       (1 Round)
                    0x80780000 0xF0000787
-                      (undo the swap)
                    0xF0000787 0x80780000
```

Fig. 41. Iterative event for $\mathcal{S} \circ \mathcal{E}^2$ with 16/8/14/8/16 active bits

In this attack we are going to identify events which are visible to the attacker and such that they immediately lead to correlations inside the cipher. Very surprisingly this is possible for GOST due to its special internal structure.

We have the following result.

Fact 112 (Cycle Structure Property of GOST). The cycle structure for the full 32-round GOST is the same as for $\mathcal{S} \circ \mathcal{E}^2$, and the fixed points for the full 32-round GOST are mapped to fixed points of $\mathcal{S} \circ \mathcal{E}^2$.

Justification: As in the Reflection Property by Kara [75] cf. Fact 17, here we see another very different case of a "conjugated" structure of type $Q^{-1} \circ P \circ Q$ but here it applies to the full 32-round GOST, not to the first 16 rounds. A well known historical theorem sometimes called "The Theorem Which Won World War 2" [55] says that P and $Q^{-1} \circ P \circ Q$ have the same cycle structure. Here $P = \mathcal{S} \circ \mathcal{E}^2$ and $Q = \mathcal{E}$ and the whole GOST encryption can be written as $Enc_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E}$ which was initially shown in [75].

This property allows the attacker to identify encryptions with fixed points for $\mathcal{S} \circ \mathcal{E}^2$ in the middle 16 rounds of GOST even though he cannot see what happens inside the cipher. Following Fact 110 we have:

Fact 113 (Internal Fixed Point Structure Property of GOST). For every fixed point A for the full 32-round GOST, $\mathcal{E}(A)$ is a fixed point for the middle 16 rounds $\mathcal{S} \circ \mathcal{E}^2$.

Now we define a new weak key class which occurs very frequently with GOST:

Definition H.2.1 (Weak Key Family 10). We define the Weak Keys Family 10 as keys such that we have the situation depicted in Fig 40 for the middle 16 rounds $\mathcal{S} \circ \mathcal{E}^2$ for a couple of fixed points for the whole 32-round GOST A, B. This occurs with probability of $d = 2^{-42}$ over all GOST keys.

Justification: The probability that GOST has 2 or more fixed points is $1 - 2/e \approx 26\% \approx 2^{-2}$, see [85, 20, 23, 24]. Now it is very surprising but nevertheless true that such fixed points exhibit very strong internal correlations. This comes from the fact that following again Fact 111 for each GOST key there exist about 2^{32} pairs P, Q which have all the differences shown in Fig. 41 above. In these pairs the difference is invariant with 16 bits under $0xF000078780780000$. Now as at many places in this paper, it is possible to see that for these 2^{32} pairs and due to the internal propagation the entropy of these differences computed over random GOST keys is lower than 16 bits, maybe about only 10 bits (more computer simulations are needed to compute it more precisely). This means that the probability that the input difference is the same as the output difference for the remaining 16 active bits in one of the 2^{32} pairs P, Q is about 2^{-10}. Thus for every GOST key we have about 2^{22} pairs P, Q with identical input and output differences on 64 bits. Now with probability 2^{-64} the image of P by the middle 16 rounds $\mathcal{S} \circ \mathcal{E}^2$ is going to be P itself which implies also another fixed point for Q. This occurs with probability $d = 2^{22-64} = 2^{-42}$ over the GOST keys.

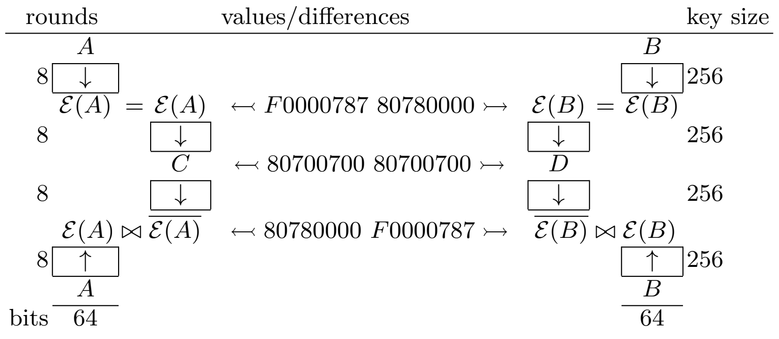


Fig. 42. Weak Key Family 10

Now it is not uncommon for the full GOST to have two fixed points, it happens for 2^{-2} of GOST keys. However for $d = 2^{-42}$ of all GOST keys we have internal relations with very high probability:

Fact 114 (Internal Double Fixed Point Correlation Property of GOST). For all Weak Keys Family 10 and for at least $d = 2^{-42}$ of all GOST keys IF A, B is a couple of fixed points for the full GOST THEN we have $\mathcal{E}(A) \oplus \mathcal{E}(B) \in 0xF000078780780000$ and moreover we also have all the other internal differences after round 9, 16, and 23 as shown in Fig. 41.

We leave the key recovery for Family 10 keys for future research. We sketch one simple attack below.

Fact 115 (Family 10 Attack). For all Weak Keys Family 10 and for $d = 2^{-42}$ we can recover the 256-bit key in time of about 2^{229}.

Justification: We guess $\mathcal{E}(A), \mathcal{E}(B)$ and our guess is correct with probability about 2^{-64-10} due to the low entropy of their difference. We guess C, D and our guess is correct with probability about 2^{-64-8} due to the low entropy of their difference. Thus we obtain 6 pairs for 8 rounds and recover the key in time 2^{83}. This is $2^{64+10+64+8+83} = 2^{229}$ total.

Further Research an Variants of Family 10. It is easy to see that this attack can be improved: we only determine parts of these values, fix a well-chosen subset of kety bits, obtain a contradiction, or extend the key and data sets, etc.. This is expected to lead to efficient attacks for Family 10.

We can also get larger d, closer to single key attacks by considering larger sets of active bits in our differentials. We leave it for future research.

H.3 A Differential-Involution Attack on GOST

We consider the first 16 rounds of GOST and the following sets of non-zero 64-bit differences at round 0,8,16 with up to respectively 16,14,16 active bits, which are shown in the middle column of Fig. 43 for the first 16 rounds.

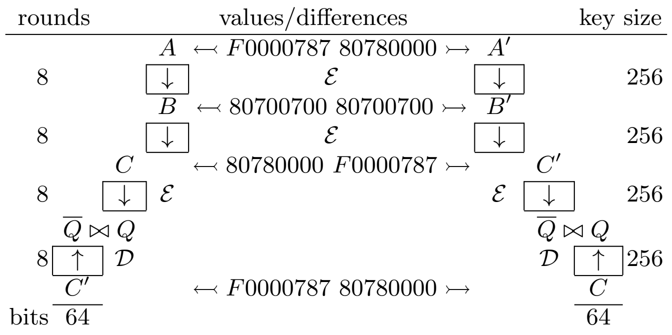

rounds
values/differences
key size
$A \leftarrowtail F0000787\ 80780000 \rightarrowtail A'$
8
$\mathcal{E}$
256
$B \leftarrowtail 80700700\ 80700700 \rightarrowtail B'$
8
$\mathcal{E}$
256
$C \leftarrowtail 80780000\ F0000787 \rightarrowtail C'$
8
$\mathcal{E}$
$\mathcal{E}$
256
$\overline{Q} \bowtie Q$
$\overline{Q} \bowtie Q$
8
$\mathcal{D}$
$\mathcal{D}$
256
$C' \leftarrowtail F0000787\ 80780000 \rightarrowtail C$
bits
64
64

Fig. 43. A Differential-Involution Attack on GOST

We have:

Fact 116 (Weak Keys Family 11, $d = 2^{-32}$). We define the Weak Keys Family 11 by keys such that there exists two points A, A' such that if $B = \mathcal{E}(A)$, $C = \mathcal{E}(B)$ and similarly $B' = \mathcal{E}(A')$, $C' = \mathcal{E}(B')$, then we have the full situation with all equalities and all the differences depicted in Fig. 43 and also additional internal differences at round 1 and 15 specified at Fig. 40. This occurs with density $d = 2^{-32}$.

Justification: Following Fact 111 there are 2^{32} pairs A, A' which satisfy the specified differences in the first 16 rounds. We denote by $Last_{16}$ the permutation which represents the last 16 rounds of GOST.

We consider the following event: can we have $Last_{16}(C) = C'$ for some pair A, A' with the desired differences in the first 16 rounds? Because we have 2^{32} such pairs, this happens with probability 2^{-32}. Therefore such a pair will exist for some weak keys with a relatively high density of $d = 2^{-32}$ over full 256-bit GOST keys. In addition, because $Last_{16}$ is an involution, if this happens we also have $Last_{16}(C') = C$, which also implies the differences in the ciphertexts. We have the full situation depicted in Fig. 43 and also additional differences at round 1 and 15 specified at Fig. 40.

Fact 117 (Key Recovery For Weak Keys Family 11). Given 2^{64} KP we can recover a weak key in Family 11 in time of very roughly about 2^{202} GOST encryptions .

Justification: In the same way in Fact 111, there exist 2^{31} cases with the outside differences which can be observed by the attacker. Just one of them will satisfy all the internal relations depicted on the upper half of Fig. 43 and also all those specified in Fig. 40 used in Fact 111. Therefore we can guess A, A', C, C' with probability 2^{-31}. Furthermore we guess B, B' with 50 difference bits already known and which will be correct with probability $2^{-64-14+1} = 2^{-77}$.

Unhappily we cannot apply Fact 7, with 2^{94} GOST encryptions because it deals with 4 unrelated encryptions which typically lead to one key on 256 bits on average. Here we have two pairs of related encryptions which share many data bits, and exactly as in Fact 117, on one side it is easier to find such individual solutions, but the number of solutions to enumerate is much higher We study this question in Appendix J.2 and we obtain 2^{94} GOST encryptions, see Fact 125. Overall in our attack on 32 rounds with Family 11, we expect to do $2^{31+77+94} = 2^{202}$ GOST encryptions to recover the full 256-bit key.

Fact 118 (Family 11 in the Multiple Key Scenario). Given a population of 2^{32} different random keys, with access to 2^{64} of data per key, we one can recover **one** of these 256-bit keys in total overall time of about 2^{234} GOST encryptions.

Justification: We run the attack 2^{32} times for both strong and weak keys.

H.4 A Differential-Double Reflection Attack on GOST

We consider the first 16 rounds of GOST and precisely those sets of non-zero 64-bit differences at round 0,8,16 with up to respectively 16,14,16 active bits, which are shown in the middle column of Fig. 44.

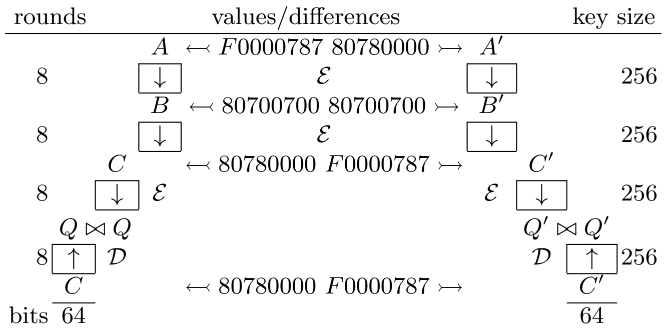

rounds
values/differences
key size
$A \leftarrowtail F0000787\ 80780000 \rightarrowtail A'$
8
$\mathcal{E}$
256
$B \leftarrowtail 80700700\ 80700700 \rightarrowtail B'$
8
$\mathcal{E}$
256
$C \leftarrowtail 80780000\ F0000787 \rightarrowtail C'$
8
$\mathcal{E}$
$\mathcal{E}$
256
$Q \bowtie Q$
$Q' \bowtie Q'$
8
$\mathcal{D}$
$\mathcal{D}$
256
$C \leftarrowtail 80780000\ F0000787 \rightarrowtail C'$
bits
64
64

Fig. 44. A Differential-Double Reflection Attack on GOST

We have:

Fact 119 (Weak Keys Family 11', $d = 2^{-32}$). We define the Weak Keys Family 11' by keys such that there exists two points A, A' such that if $B = \mathcal{E}(A)$, $C = \mathcal{E}(B)$ and similarly $B' = \mathcal{E}(A')$, $C' = \mathcal{E}(B')$, then we have the full situation with all equalities and **two reflections** and all the differences depicted in Fig. 43 and also additional differences at round 1 and 15 specified at Fig. 40. This occurs with density $d = 2^{-32}$.

With this Family 11' we do NOT expect attacks better than with Family 11.

I More Approximate Fixed Point Bicliques and Other Multiple Point Properties

In this section we continue the work of Section 26.3 and we provide additional examples and interesting special cases.

I.1 Discovery of Three Point Properties

The objective of this section it to study some basic 3-point differential properties which can be useful in order to justify certain results in earlier Section 26.3 and in Appendix I below. They are also of independent interest.

We ask the following question: let D be an integer in the range $9 \leq D \leq 24$. Is there a property with D active bits such that if we fix $64 - D$ bits at random, and look at 8-round properties which share the same $64 - D$ bits at the input, and consider an affine space of 2^D points with these $64 - D$ bits, are there any triples in this space such that the three points also share the same $64 - D$ bits at the output? In addition we would like to know which sets are the most interesting sets: we examine many different sets following the heuristic evolutionary discovery method described in [45] and report the best results we could obtain after trying for a few days.

This is exactly what we called "Truncated I/O Multi-collisions" before, cf. Fig. 26 which we reproduce here for convenience.

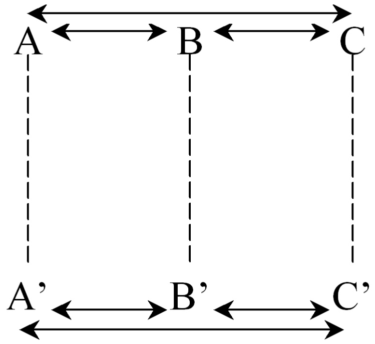


Fig. 45. Truncated I/O multi-collisions with $k = 3$: the 3 inputs live in the same space of dimension D and the 3 outputs live in another space of dimension D.

Remark. Some results are given in Table 8. For example in line 3 we find a set of active 14 bits such that given one fixed plaintext on 50+14 bits and 2^{14} variants of it with varying the 14 active bits, we expect that with probability of about $2^{-8.5}$ there will be an event with 3 points which share 50 bits at the output, i.e. their pairwise differences are contained in the same mask on 14 bits. This result can be use to provide another justification for the (approximate) figure 2^{-8} obtained for three points in Fact 64. Here for every shared 50 bits at the input the chances to have a 3-point property A, B, C, A', B', C' with the outputs sharing the specified 50 bits are $2^{-8.5}$. Overall there are $2^{50-8.5}$ such cases, and inside these there are about $2^{50-8.5-50}$ events where the input shared 50 bits and the output shared 50 bits coincide.

Table 8. Study of invariant truncated differential properties with 3 points for some values of $9 \leq D \leq 24$ bits. We display a rough approximate average number of such events per one set of 2^D plaintexts for one fixed value on $n - D$ bits.

D		GOST S-box Set Name	Truncated differential set S	3 points **8R**
14	0	GostR3411_94_TestParamSet	80700700 80700700	$2^{-8.5}$
17	0	GostR3411_94_TestParamSet	80700704 80701704	$2^{-3.2}$
19	0	GostR3411_94_TestParamSet	A4700704 80701704	$2^{-1.9}$
20	0	GostR3411_94_TestParamSet	A0780702 A0780702	2^{+3}
20	2	Gost28147_TestParamSet	A0780702 A0780702	2^{+3}
20	3	Gost28147_CryptoProParamSetA	A0780702 A0780702	2^{+2}
22	0	GostR3411_94_TestParamSet	A0780782 A0780782	2^{+6}
22	2	Gost28147_TestParamSet	A0780782 A0780782	2^{+6}
22	3	Gost28147_CryptoProParamSetA	A0780782 A0780782	2^{+5}
24	0	GostR3411_94_TestParamSet	7070707007070707	2^{+13}

I.2 Approximate Fixed Point Bicliques For 4 Rounds

For 4 rounds it is much easier to find such configurations.

Fact 120 (A 2/3/4-point approximate fixed point biclique for 4 rounds of GOST and $\mathbf{D} = 14$). For a typical GOST key we have $2^{13.4}$ possibilities for the set of 4 points A, B, A', B', such that A', B' are the encryption of A, B after 4 rounds, and which have differences with up to 14 bits and such that all 2+2 points share the same set of 50 bits, as depicted on earlier Fig. 21.

We also have $2^{10.5}$ possibilities for the set of 6 points A, B, C and A', B', C', such that A' is the encryption of A after 8 rounds etc. , and which have differences with up to 14 bits and such that all the 3+3 points share the same set of 50 bits, as depicted on earlier Fig. 27.

For proportion of at least about $2^{-1.6}$ of GOST keys there exists a set of 50 bits such and a suitable set of 8 points A, B, C, D and A', B', C', D', and such that all 4+4 points share 50 bits as depicted on earlier Fig. 25.

Justification: The justification is the same as for Fact 64, only some probabilities need to be updated. For 4 rounds if $A \oplus B \in 0x8070070080700700$ then if $A' \oplus B' \in 0x8070070080700700$ with probability $2^{-13.6}$ cf. [37, 38, 40]. Thus we expect that there are on average about $2^{77-13.6-50} \approx 2^{13.4}$ pairs which share a fixed 50 common bits and such as depicted in Fig. 21.

For $k = 3$, if A, A' and B, B' are already the suitable pairs for 4 rounds, our simulation shows that it is no longer easier to find more such pairs C, C' but a bit harder (for 4 rounds, configurations with 2 points are more frequent than those with more points) and the propagation occurs with probability of about $2^{-15.3}$. Further simulations show that there will be a fourth pair D, D' with probability of about $2^{-12.1}$ a fifth pair probability of about $2^{-11.1}$ and more than 5 with high probability. Thus, for $k = 3$ and on average for one GOST key there are on average about $2^{89.4-13.6-15.3-50} \approx 2^{10.5}$ triples such as in Fig. 27. For $k = 4$ we expect about $2^{89.4-13.6-15.3-12.1-50} \approx 2^{-1.6}$ 4-tuples as in Fig. 25.

This Fact 120 has interesting unexpected consequences:

Fact 121 (Double symmetric fixed points for 4 rounds of GOST). For 4 rounds of GOST, there exists two symmetric fixed points with probability of at least 2^{-47} instead of 2^{-64} for a random permutation.

Justification: From Fact 120 for $k = 2$ we know that there are $2^{13.4}$ possibilities for A, B, A', B' as depicted in Fig. 21 and sharing the same 50 bits. Then A is symmetric with probability 2^{-32} and B will also be symmetric with probability 2^{-7} and then $A' \oplus B' = A \oplus B$ with probability of about 2^{-7} (similar but lower than in low entropy discussion in Section 26.1). and $A = A'$ with probability of about 2^{-14}. Overall we expect to get 2 symmetric fixed points with probability at least $2^{13.4-32-7-7-14} \approx 2^{-47}$ instead of 2^{-64} for a random permutation.

Remark. The Fact 121 is only a lower bound. In the real life what is the probability that 4 rounds of GOST has 2 symmetric fixed points? Our simulations show that it is indeed roughly about 2^{-47} and that pairs of symmetric fixed points found tend to have interesting small Hamming weight differences. We have tested this fact for various known sets of GOST S-boxes and we have seen no significant difference. More research about this topic is needed.

I.3 Alternative Sets With $D = 24$

Similar results as Fact 64 and Fact 120 can be obtained for other spaces of dimension $D = 14$ some of them not symmetric, see Section 25.6. Other interesting results can be obtain for other values of D. Below we give one example.

It is interesting to discover that in terms of propagation probabilities for pairs of points, the set of differentials with 24 active points using the mask $0x70707070070707$ by Seki and Kaneko [100] is not as good as the set using the mask $0x8070070080700700$ from [36–38, 40]. According to [38] this first property propagates for 8 rounds with probability of 2^{-30} which is substantially lower than predicted by the authors, see [100, 38]. The second property propagates for 8 rounds with probability of 2^{-25} cf. [37] which may seems better. However in terms of the number of pairs A, B with respective difference propagation, both sets are quite good. In the first case we have $2^{64+24-1-30} = 2^{57}$ pairs, in the second case we have $2^{64+14-1-25.0} = 2^{52}$ pairs.

Now a major observation is that the number of triples, 4-tuples and larger sets in which all possible pairs have suitable differences is still relatively large compared to the number of pairs. Two examples were already given in Fact 64 and 120, now we look more closer at the sets with $D = 24$ based on [100]. The sets of [100] has only 40 fixed bits as opposed to 50 bits in [36–40], which was handicap in distinguishers of [36–40] but it is going to become an advantage here. Unhappily these alternative old/new difference masks with 24 active bits [100] are not symmetric so they are not suitable for the same applications, for example they are not suitable for Weak Family 8.2. attack and many other.

Fact 122 (A 2/3/4-point approximate fixed point biclique for 8 rounds of GOST and $\mathbf{D} = \mathbf{24}$). For a typical GOST key we have on average 2^{17} possibilities for the set of 4 points A, B, A', B', such that A', B' are the encryption of A, B after 8 rounds, AND which have differences with up to 24 bits following the pattern using the mask $0x70707070070707$ and share the same set of 40 bits, as depicted in Fig. 21.
For $k = 3$ we we have about 2^{14} possible sets of 6 points A, B, C and A', B', C' sharing the same set of 40 bits and such as depicted in Fig. 27.
For $k = 4$ we we have about 2^{11} possible sets of 8 points A, B, C, D and A', B', C', D' sharing the same set of 40 bits and such as depicted in Fig. 25.

Justification: We consider the affine space of $D = 24$ defined by the mask $0x70707070070707$ from [100]. This property propagates for 8 rounds with probability of 2^{-30} [38] and the number of pairs A, B with respective difference propagation for 8 rounds is $2^{64+24-1-30} = 2^{57}$ pairs. Inside these the input shared 40 bits are going to be equal to the output 40 bits with probability 2^{-40}. This gives 2^{17} approximate fixed point pairs.

As before if both $A \oplus B$ and $A' \oplus B'$ have suitable differences then it is **easier** to find more such pairs C, C'. Our early simulations show that there is roughly about 2^{14} approximate fixed point triples and further roughly about 2^{11} approximate fixed point 4-tuples. More simulations are needed to obtain exact figures.

I.4 Multiple Fixed Points for 8 Rounds

As in Fact 121, all our work on simultaneous truncated differentials has some unexpected consequences.

Fact 123 (Multiple related fixed points for 8 rounds of GOST). For 8 rounds of GOST and a random key, the probability that there is a fixed point is about 0.63 and is expected to be the same for a random permutation. However the probability that there are two fixed points sharing 40 bits as in set $0x70707070070707$ is at least 2^{-19} instead of 2^{-42} for a random permutation.
Stronger pairs of fixed points sharing as many as 50 bits as the inactive bits in mask $0x8070070080700700$ exist with probability at least 2^{-20} instead of 2^{-52} for a random permutation.
Similarly the probability that there are three fixed points for 8 rounds with shared 40 bits is at least 2^{-32} instead of 2^{-84} for a random permutation. Additionally three fixed points sharing 50 bits within $0x8070070080700700$ exist with probability at least 2^{-34} instead of 2^{-104} for a random permutation.
Finally the probability that there are four fixed points for 8 rounds with shared 40 bits is at least 2^{-45} instead of 2^{-126} for a random permutation. Additionally four fixed points sharing as many as 50 bits within $0x8070070080700700$ exist with probability at least 2^{-45} instead of 2^{-156} for a random permutation.

Justification: We refer to [20, 23, 24, 85] for the basic facts about statistics on the number of fixed points in random permutations.

We apply Fact 122. We get 2^{17} possibilities for the set of 4 points A, B, A', B', such that A', B' are the encryption of A, B after 8 rounds, AND sharing a fixed value on 40 bits Now following Section 26.1 but for a different set we estimate roughly that the entropy of $A \oplus B$ is low and the probability that $A \oplus B = A' \oplus B'$ is about 2^{-12}, this is a conservative estimation. Furthermore $A = A'$ with probability 2^{-24} which also implies $B = B'$. Overall we expect that for a proportion of $d = 2^{17-12-24} = 2^{-19}$ of GOST keys we have two fixed points A, B sharing the same 40 bits.

For 50 bits and the mask $0x8070070080700700$ we apply Fact 64 and obtain a proportion of at least $d = 2^{2-8-14} = 2^{-20}$ of GOST keys for which we have two fixed points A, B sharing the same 50 bits.

For $k = 3$ we we have about 2^{14} possible sets of 6 points A, B, C and A', B', C' sharing the same set of 40 bits and such as depicted in Fig. 27. We postulate that the probability that $A \oplus B = A' \oplus B'$ and simultaneously $A \oplus C = A' \oplus C'$ is about 2^{-22}, this is a conservative estimation. Furthermore $A = A'$ with probability 2^{-24} which also implies $B = B'$ and $C = C'$. Overall we expect that for a proportion of $d = 2^{14-22-24} = 2^{-32}$ of GOST keys we have three fixed points A, B, C sharing the same 40 bits. For 50 bits and the mask $0x8070070080700700$ we apply Fact 64 and obtain a proportion of at least $d = 2^{-8-12-14} = 2^{-34}$ of GOST keys for which we have three fixed points A, B, C sharing the set of 50 bits we defined.

For $k = 4$ we we have about 2^{11} possible sets of 8 points A, B, C, D and A', B', C', D' sharing the same set of 40 bits and such as depicted in Fig. 25. We postulate that the probability that $A \oplus B = A' \oplus B'$ and simultaneously

$A \oplus C = A' \oplus C'$ and simultaneously $A \oplus D = A' \oplus D'$ is maybe about 2^{-32}. Furthermore $A = A'$ with probability 2^{-24} which also implies $B = B'$ and $C = C'$ and $D = D'$. Overall we expect that for a proportion of roughly maybe about $d = 2^{11-32-24} = 2^{-45}$ of GOST keys we have four fixed points A, B, C, D sharing the same 40 bits. For 50 bits and the mask $0x8070070080700700$ we apply Fact 64 and obtain a proportion of at least roughly maybe about $d = 2^{-7-24-14} = 2^{-45}$ of GOST keys for which we have four fixed points A, B, C, D sharing the set of 50 bits we defined.

These probability estimations are quite imprecise and require more work.

I.5 Events With More Than 4 Points and Single Key Events

In Section 26.3 we have done a longer simulation with 2^{39} encryptions with random keys. In this simulation we have seen an event with more than 4 points only once. Moreover the events studied in Section 26 happen only for some keys, for example 2^{-9} GOST keys.

One method to obtain more points or/and events with single keys is to use less rounds. See Appendix I.2 for the study of events for 4 rounds of GOST. Another method it to takes a larger mask. Then we can easily obtain events with more points or/and which happen for all GOST keys.

Example of Event With 12 Points. For example with $0xF0F80F80F0F80F80$ which has 28 active bits we can easily obtain as many as 12 points which share the same set of 64-28=36 inactive bits. We show one example found at random and without any special properties below:

```
8 rounds 12 points 36 inactive bits F0F80F80F0F80F80
key=03F5F3471CDC6376EF079890A8AD0B3BC747FB61F05E21C722F3B06C77E44D8D
P1=64E9DC846E50462A P2=04C1D1046E90462A P3=8411D6041ED04D2A
P4=A4F1D684BE90452A P5=E481D4046E60442A P6=A4C1D9846E60412A
P7=74E1D004FED0482A P8=A4C1D2846EA0472A P9=F4D1D5846EA0472A
P10=E489D7843E60442A P11=44E1DE04FEA047AA P12=24C1D404EEB04A2A
C1=9F92744EC5C905B0 C2=7F9A724EB57109B0 C3=8F9A704EB5090EB0
C4=DF92724EA5710D30 C5=FF9A744E75C104B0 C6=AFA27B4EC5D90430
C7=5FA276CE15690F30 C8=5F92764E05390C30 C9=9FAA714E65A102B0
C10=8F92734E25A907B0 C11=CF92764EF5E10630 C12=AF92704EA5090830
```

Such events are surprisingly frequent. In order to find this event we had to do try only about 2^{36} encryptions for 8 rounds with random keys. In a sense we have tried 2^{36} pairs composed of a key and input 36 bits. Therefore we expect very roughly that for every key there exists maybe one case of the input 36 bits on average which gives more than 8 points. Therefore even having as many as 8 points seems to be a single key event which happens for a majority of GOST keys, while 12 points might happen for at least 50 % of keys or similar. This is based on rough estimations.

Part XI

Additional Low-Level Attacks

J Additional Low-Data Attacks For 8 Rounds

In this section we provide additional variants of attacks from Section 12.

J.1 A Dedicated Attack for 3 KP and 8 R With Internal Differences

This attack is designed for case with 3 KP such that the first two plaintexts have specific input and output differences such as in differential attacks on GOST [36–40] and depicted in Fig. 18 page 95.

This result is NOT at all obvious given the fact we also obtained 2^{110} for ordinary case of 3 KP in Fact 6. Here two plaintexts are related and therefore these encryptions share some internal bits with high probability, and therefore there is more solutions to enumerate than in Fact 6. However on the other side each solution is easier to enumerate precisely because of these plausible difference relations on internal bits, which is a powerful thing. This case requires a very careful dedicated attack.

Fact 124. Given 3 KP for 8 rounds of GOST, with the last two of them having both input difference and output difference within the set of $2^{14} - 1$ non-zero differences of the form $0x80700700, 0x80700700$, and also a middle difference of the form $0x80700700, 0x80700700$ after 4 rounds, all candidates for the full 256-bit key can be enumerated in time of about 2^{110} GOST computations and negligible memory.

Justification: We proceed as follows.

1. We work with the right pane in Fig. 6 with 87+87 key bits.
2. We have fix $2 \cdot 32 = 64$ and NOT $3 \cdot 32$ middle bits. We discard one of the encryptions related by input/output differentials and have only two ordinary encryptions without any special properties on the data.
3. Following Fact 14, given a fixed set of 64 middle bits we would expect to get only $2^{87+2 \cdot 3.6-64} = 2^{30.2}$ possibilities for 87 bits of the key in the upper 4 rounds.
4. Furthermore, for the third encryption, we do not yet take into account the third 32 middle bits which are partly related to the middle bits already known, partly not yet known.
   However we know all the 3 inputs and for the last 2 encryptions, we only consider case where that given that in the current data, the input difference is of the form $0x80700700, 0x80700700$, for the $2^{30.2}$ various keys considered here, many of which are incorrect but this can only be seen later in the attack, the output difference at round 4 will be of the form $0x80700700, 0x80700700$, with probability $2^{-13.6}$, in the same way as in [38].
5. Overall we estimate that there is maybe about $2^{87+2 \cdot 3.6-64-13.6} = 2^{16.6}$ possibilities for 87 bits of the key in the upper 4 rounds.
6. In the same way, for the same 64 middle bits, we expect $2^{16.6}$ possibilities for the 87 key bits in the lower 4 rounds.

7. We fix 16 key bits out of 87: bits 0-11 in the first round and bits 47-50 in the second round. Then we run a SAT solver with 3 KP and 4 rounds, where we assume the 16 key bits guessed for all encryptions, guess 32+32 middle data bits for the first two plaintexts, and add equations which say that the output difference for the last 2 plaintexts is of the form $0x80700700, 0x80700700$ after 4 rounds.
8. We want to determine the missing 87-16 bits. This takes less than 0.6 s on average with CryptoMiniSat 2.92 [104]. This is about 2^{21} GOST encryptions on the same CPU. Unhappily the solution is not unique.
9. If the system is UNSAT, it means that the 16 bits can be rejected.
10. It may seems strange that we enumerate $2^{16.6}$ solutions by examining 2^{16} cases and most cases are rejected. The explanation is that solutions are NOT random but correlated, they are organized in clusters with several solutions sharing identical 16 bits. This is due to limited diffusion in GOST. Fact 8 deals with averages however in practice we have 0, 1 or several cases.
11. Total time spent in this step is about $2^{64+16+21} = 2^{101}$ GOST encryptions.
12. We do further steps only if the system is SAT which happens with probability experimentally about 2^{-3}. Overall because we accept only a proportion of 2^{-3}, we expect $2^{16-3} = 2^{13}$ cases which give SAT and $2^{16.6-13} = 2^{3.6}$ clustered solutions in each case.
13. We extend these 16 bits by 4,8,12 and more bits, four bits at a time, checking for UNSAT, which takes at most 0.5 s in each case, is done for four more bits only for about 2^{-3} of cases, for four more bits only for a smaller proportion of cases, etc...
14. Overall we expect that given the 16 bits already confirmed by SAT, we can extend them by 4, 8 etc more bits, in a tree-like search with only $2^{3.6}$ branches total for strings of 87 bits, and we claim that branching in this tree is achieved progressively due to low diffusion in the cipher.
15. Locally strings have low entropy and each time we extend by four more bits, we will frequently examine 2^2 cases on 2 bits, which takes at most 0.5 s in each case, yet we will keep most of the time only one sometimes two cases. The time to explore this "long" tree with only $2^{3.6}$ leaves at the end and depth about (87-16)/4 can be estimated to be about $2^{3.6} \cdot (87-16)/4 \cdot 2^{21} \approx 2^{29}$ GOST encryptions.

    Overall we expect that this process takes about $2^{64+16+29} = 2^{109}$ GOST encryptions.
16. At this stage we get an enumeration of $2^{64+16.6}$ cases with keys on 87 bits and 32+32 bits of internal data for 2 pairs. Now we look at the 32 bits of data for the 3rd pair.
17. Following the right pane of Fig. 6 the active bits are 12-27 on the left and 28-11 on the right. Some bits of our third set of 32 middle bits are already constrained by the differences of the form $0x80700700, 0x80700700$ with inactive being 9-11,21-23,32. New bits which are not yet determined and can be arbitrary values are 12-20,24-27 on the left, and 28-9 on the right. We have 13+14=27 bits out of 32 which are not yet determined by our assumptions.

18. Following Fact 14, we will have in each case with 32+32 data bits and 87 key bits, $2^{3.6}$ possibilities for these 27 bits, and the time of this enumeration of $2^{3.6}$ cases can be neglected, because it is done only $2^{64+16.6}$ times overall.
19. In the same way, for the same 64 middle bits, we enumerate all the $2^{16.6}$ possibilities for the 87 key bits in the lower 4 rounds, plus $2^{3.6}$ possibilities for the same middle 27 bits, for each case, enumerated at negligible additional cost. Time spent in this step is another 2^{109} GOST encryptions and 2^{110} GOST encryptions total for both.
20. We need a negligible quantity of memory to store these two sets of $2^{16.6}$ half-keys on 87 bits.
21. Now we are going to enumerate all possible cases which will agree on the 27 middle bits for the third encryption.
22. For every 32+32 middle bits, we have two lists of $2^{16.6+3.6}$ possibilities on 27 bits. For every 27 bits from the first list it will be in the second list with probability $2^{16.6+3.6-27} = 2^{-6.8}$. Therefore out of $2^{16.6+3.6}$ possibilities on 27 bits in the first list, only $2^{16.6+3.6-6.8} = 2^{13.4}$ will survive.
23. Thus in our attack given 32+32 middle bits we are able to enumerate only $2^{13.4}$ keys on 174=87+87 key bits.
24. In each case, given the 87+87 bits we run a SAT solver to determine the remaining 256-174 bits. This is done $2^{64+13.4} = 2^{77.4}$ times. It takes just 1 s which is 2^{21} GOST encryptions.
25. This step takes about $2^{77.4+21} = 2^{98.4}$ GOST encryptions.
26. Overall our attack requires 2^{110} GOST encryptions and very small memory.

J.2 A Dedicated Attack for 4 KP and 8 R With Internal Differences

In Section 12.1 we provide one particularly simple attack on 4 rounds which was initially developed and published in [35]. In this section we provide extensions and variants of this attack which are needed in this paper to establish several results which mix differential cryptanalysis and black-bow reductions. These attack are designed for case with 4 KP such that the two pairs of plaintexts have specific input and output and middle differences as for example on one half of Fig. 111 (which is the same configuration as in the lower half). As in Section J.1 two plaintexts are related and share internal bits with high probability or have low joint entropy, and therefore one one hand there will be more solutions to enumerate than in Fact 7, and on the other hand each solution is easier to enumerate. We need to develop a careful and dedicated attack. Our first result for 4 KP is as follows.

Fact 125. Given 4 KP for 8 rounds of GOST, with two of them having differences such as in the first 8 rounds of Fig. 40, all the candidates for the full 256-bit key can be enumerated in time of about 2^{94} GOST computations and negligible memory.

Justification: We proceed as follows.

1. We use the encoding of GOST described in [46].
2. We use a guess then determine approach with a highly non-trivial set of bits which are guessed, the other bits are determined by software.
3. We use the following specific set of 68 bits following [35, 32]. The exact bits used are: 0-15,51-55,64-66,128-130,179-183,192-207,224-231,244-255.
4. We convert our problem to a native SAT problem with native XORs such as accepted by CryptoMiniSat 2.92 software [104].
5. We run the software 2^{68} times for all possible assignments of the 68 bits.
6. Computer simulations on a laptop Intel i7 CPU show that if we run this software with the timeout of 27 seconds, a proportion of $1-2^{-6.5}$ of cases on 68 bits terminates with UNSAT within 0.5 s on average or 2^{21} GOST encryptions. The total time spent in this step is about $2^{68+21} \approx 2^{89}$ GOST computations.
7. Now for each surviving key on 68 bits, we are going to extend it by 12 more bits, to 80 bits. The additional 12 bits are: 16-19,128-130,160-161.
   Now with a SAT solver we observe that a further proportion of $1-2^{-8.5}$ of cases on 80 bits terminates with UNSAT within 0.25 s on average or 2^{20} GOST encryptions. Overall a proportion of 2^{-15} keys on 80 bits survives.
   Time spent in this step is about $2^{68+12-6.5+20} \approx 2^{93.5}$ GOST computations.
8. We add 4 more bits which are 34-35,161-163 to make 84 guessed bits total. A further proportion of $1-2^{-2}$ of cases on 88 bits terminates with UNSAT within 0.25 s on average or 2^{20} GOST encryptions. Time spent in this step is about $2^{84-15+20} \approx 2^{89}$ GOST encryptions. Overall a proportion of 2^{-17} keys on 84 bits survives.

9. We add 4 more bits which are 231-235 to make 88 guessed bits total. A proportion of $1 - 2^{-7}$ of cases on 88 bits terminates with UNSAT within 2^{20} GOST encryptions. Time spent in this step is about $2^{88-17+20} \approx 2^{91}$ GOST encryptions. Overall a proportion of 2^{-24} keys on 88 bits survives.
10. Our 88 guessed bits total was a threshold determined experimentally, so that when we run a SAT solver now with the correct 88 bits, a correct and unique solution is always found. This takes 40 s which is about 2^{27} GOST encryptions. Time spent in this step is about $2^{88-24+27} \approx 2^{91}$ GOST encryptions.
11. Overall the attack requires 2^{94} GOST encryptions total.

J.3 Second Dedicated Attack for 4 KP and 8 R With Internal Differences

This attack is motivated by the Family 8.4 attack from Section 28.6 which is one of our fastest attacks on GOST however we are not sure if the last step of this attack works as well as in Fact 125 because it produces 4 points with very few bit differences which in the same way as in earlier Fact J.1 and J.2 makes it easier to find individual keys but at the same time the number of solutions increases. We need another careful and dedicated attack and we need to evaluate the number of solutions.

Fact 126. Given 4 KP for 8 rounds of GOST, which all share the same 50 bits at the input as in mask $0x$8070070080700700, and which also share another 50 bits at the output as in mask $0x$8070070080700700, there is at most 2^{88} solutions for the full 256-bit key and they can be enumerated in time of about 2^{99} GOST computations and negligible memory.

Justification:

example of 4 points with the same key:

In order to recover the key we proceed as follows.

1. We use the encoding of GOST described in [46].
2. We use a guess then determine approach with a highly non-trivial set of bits which are guessed, the other bits are determined by software.
3. We use the following specific set of 68 bits following [35, 32]. The exact bits used are: 0-15,51-55,64-66,128-130,179-183,192-207,224-231,244-255.
4. We convert our problem to a native SAT problem with native XORs such as accepted by CryptoMiniSat 2.92 software [104].
5. We run the software 2^{68} times for all possible assignments of the 68 bits.
6. Computer simulations on a laptop Intel i7 CPU show that if we run this software with the timeout of 27 seconds, a proportion of $1 - 2^{-5}$ of cases on 68 bits terminates with UNSAT within 0.3 s on average or 2^{20} GOST encryptions. The total time spent in this step is about $2^{68+20} \approx 2^{88}$ GOST computations.
7. Now for each surviving key on 68 bits, we are going to extend it by 12 more bits, to 80 bits. The additional 12 bits are: 16-19,128-130,160-161.

Now with a SAT solver we observe that a further proportion of $1 - 2^{-4}$ of cases on 80 bits terminates with UNSAT within 0.1 s on average or 2^{19} GOST encryptions. Overall a proportion of 2^{-9} keys on 80 bits survives. Time spent in this step is about $2^{68-5+12+19} \approx 2^{94}$ GOST computations.

8. We add 8 more bits which are 34-35,161-163,231-235 to make 88 guessed bits total. A further proportion of $1 - 2^{-2}$ of cases on 88 bits terminates with UNSAT within 0.1 s on average or 2^{19} GOST encryptions. Time spent in this step is about $2^{88-9+20} \approx 2^{99}$ GOST encryptions. Overall a proportion of 2^{-11} keys on 88 bits survives.
9. With our 88 guessed bits total, when we run a SAT solver, a unique solution is found. **This means that there is less than 2^{88} solutions.**
   This takes 1 s which is about 2^{21} GOST encryptions. Time spent in this step is about $2^{88-11+21} \approx 2^{98}$ GOST encryptions.
10. Overall the attack requires 2^{99} GOST encryptions total.

J.4 Attacks on 6 KP and 8 Rounds

If we can obtain 2^{94} for 4 KP and 8 rounds [35] one can expect that we will obtain less for 5 KP and 6 KP. We have the following new attack which is in structure very similar to one attack given in [35]:

Fact 127. Given 6 KP for 8 rounds of GOST the full 256-bit key can be found in time of about 2^{83} GOST computations and negligible memory.

Justification: We proceed as follows.

1. We use the encoding of GOST described in [46].
2. We use a guess then determine approach with a highly non-trivial set of bits which are guessed, the other bits are determined by software.
3. We use the following specific set of 54 bits which is similar to the sets used in [35]. The bits used are: 0-15,51-55,179-181,192-203,224-229,244-255.
4. We convert our problem to a native SAT problem with native XORs such as accepted by CryptoMiniSat software [104].
5. We used the 64-bit version of CryptoMiniSat 2.92 under Windows 7.
6. We run the software 2^{54} times for all possible assignments of our 54 bits.
7. Computer simulations on a laptop Intel i7 CPU show that if we run this software with the timeout of 200 seconds, a proportion of $1 - 2^{-8}$ of cases on 54 bits terminates with UNSAT within 95 s on average.
8. However if the assignment of 54 bits happens to be correct, and if we run the same simulation with the timeout of 300 seconds, and on average over the GOST keys, the SAT solver will output the correct key with probability about 80 %.
9. Overall, it is NOT necessary to run all the 2^{54} cases for 300 seconds, this is because a proportion of $1 - 2^{-8}$ of cases terminates automatically with UNSAT within 95 s average time which is $2^{28.5}$ GOST encryptions on the same CPU.
10. Assuming that all the other cases run for 300 s (some still terminate earlier) our conservative estimate of the attack time is $2^{54+29}+2^{54+31-8} \approx 2^{83}$ GOST computations.
11. This for a success probability of at least 80 %.

Part XII

Induction Properties

K Inference of Internal Bits, Differential Induction

In this section we introduce a new guiding principle for the design of attacks on GOST. A sort of meta-attack which can be applied in a variety of ways and leading potentially to a large number of distinct attacks on GOST, at least in theory. However in fact it is not very new. On the contrary. It is another way of looking at many guess-then-determine attacks. In these attacks we work on creating additional equations or relations between various data points inside the cipher, at an overall low cost, or we try to optimize the price to pay for a set of relations inside the cipher. In this section we look at very similar situations but we emphasize something very different: once certain relations (or constraints) are imposed, other relations (or constraints) will occur nearly for free or with very high probability. We call it **“Induction”** which emphasizes the fact that attacker can impose constraints remotely at places inside the cipher which are not easily accessible.

Induction is also a form of two directional propagation of constraints. It is yet another form of making some assumptions on the state of several encryptions, and getting additional relations, hopefully at a relatively low “cost” or/and with high “Amplification” (cf. Section 8 and [32]).

Even though Induction is just another method of viewing many already known tricks and methods used in our attacks on GOST, it is a valuable scientific and cryptanalytic tool. This is because it allows for systematic study, comparison and thus further fine combinatorial optimisation of such events to find better events and even better attacks. We do not formalize this property fully, and it can exist in many different variants and flavors.

In most cases we look at situations of type 8+8 rounds and we focus on generating some interesting relations on the data in the middle of the encryption process.

K.1 The General Principle of Induction

The general principle of Induction is that we consider a combination of two key-dependent permutations $Q \circ P$, and we consider some relations or constraints involving one, two or more encryptions. We add a number of constraints at both ends, and we want to “induce” some equations in the middle by the combination of constraints propagated from both ends. We are not looking for “black box” properties but rather those which depend on special properties and internal structure of P and Q.

```
 input constraints                  input constraints
       P                                   P
 ??middle values??     becomes     induced relations
       Q                                   Q
output constraints                  output constraints
```

Fig. 46. General principle of “Induction”: relations in the middle may occur with high probability due to a combination of constrains from both sides

K.2 Induction With One Encryption

In this section we give some examples of Induction with one single encryption. These examples have already been exploited in numerous attacks on GOST in this paper.

For example we can re-visit the well-known fixed point property for an iterated permutation and apply it to the first 16 rounds of GOST.

```
1 point, first 16 rounds of GOST
                   A                                   A
                                                   (8 Rounds)
              (16 Rounds)        becomes               A
                                                   (8 Rounds)
                   A                                   A
                 1 bit                              65 bits
A is an arbitrary unknown value
```

Fig. 47. Fixed points in the first 16 rounds of GOST seen as an Induction property: the value in the middle is obtained nearly for free instead of 2^{-64}

Fact 128 (Fixed-Point Induction For 16 Rounds of GOST). If A is a fixed point for the first 16 rounds of GOST, A is also a fixed point for the first 8 rounds of GOST with probability about 50 %, cf. Fig. 47.

Justification: For a permutation, the probability to have 1 or more fixed points is about 63%, cf. [85, 20, 23, 24]. This explains why in Fig. 47 we have written 1 bit below the left hand side. We expect that it is also about 63% for 8 rounds of GOST. For 16 rounds we expect it is higher: 63% which would be obtained for any permutation and also 63% which would be inherited from 8 rounds, with substantial overlap.

K.3 Induction With One Encryption For the Last 16 Rounds

For the last 16 rounds of GOST Induction is easier, which can be established in relation to the Reflection Property of Fact 17.

```
1 point, LAST 16 rounds of GOST
               A                                   A
                                               (8 Rounds)
          (16 Rounds)        becomes            symmetric
                                               (8 Rounds)
               A                                   A
             0 bit                              32 bits
A is an arbitrary unknown value, many solutions
```

Fig. 48. Reflection in the last 16 rounds of GOST seen as an Induction property: 32 equalities in the middle are obtained nearly for free instead of 2^{-32}

Fact 129 (Reflection-Based Induction For 16 Rounds of GOST). If A is a fixed point for the last 16 rounds of GOST, we have a symmetric fixed point inside with probability 100 %, cf. Fig. 48.

Justification: We recall that the last 16 rounds of GOST can be written as: $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$. A fixed point for the whole gives also a fixed point for $\mathcal{S}$, and $\mathcal{S}$ has exactly 2^{32} fixed points which are exactly all the symmetric values with both halves equal.

K.4 Induction With Several Encryptions with Composition of Involutions

Reflection attacks use a single involution. There are also induction attacks with a composition of two involutions. Let $\mathcal{Q} \circ \mathcal{P}$ be a composition of two involutions.

One simple method is to observe that each involution will have 2^{32} fixed points, cf. Section 13.4, and their composition $\mathcal{Q} \circ \mathcal{P}$ is likely to have a shared fixed point, which allows therefore the attacker to infer a value inside the encryption process with very high probability, this is because overall we do NOT expect the whole large permutation $\mathcal{Q} \circ \mathcal{P}$ to have many other fixed points.

Now imagine that we are not lucky and the two involutions do NOT have a shared fixed point. This method can be generalized as follows.

We recall our earlier Fact 21 page 50. Consider two short cycles of length k for $\mathcal{Q} \circ \mathcal{P}$. Then the consecutive points on these cycles are going to be mapped by $\mathcal{P}$ with high probability. We just need to guess which of the two cycles is the origin, and which is the right starting point on the other cycle. Overall we see that we can generate as many as k P/C pairs for $\mathcal{P}$ which are correct with a still quite large probability of about $\frac{1}{2k}$.

This can also be interpreted as an induction property with 2k encryptions: we have 2k P/C pairs for a large permutation with $p + q$ cipher rounds, and we can guess values inside after p rounds, see Fact 21.

K.5 Differential Induction For 8 Rounds

In **Differential Induction** we will be looking at pairs of encryptions. The main idea was already explicitly discussed in Section 26.1 in relation to Fig. 18, however it was not clear if this property is very interesting. In this section we make it a central object of interest and a starting point for many attacks.

We start with the following fundamental example for 8 rounds: cf. Fig. 49 below. We look at two different encryptions for 16 rounds of GOST, regardless of the key schedule: it could be the first 16 rounds of GOST, the last 16 rounds of GOST, or any other.

We assume that the input difference is of the form $A \oplus B \in 0x8070070080700700$ which is equivalent to fixing exactly 50 bits of the difference between the two plaintexts and assuming that they are distinct. This is like adding 50 bits of entropy or knowledge about the two encryptions. Then we also assume that the same 50 “inactive” bits are also equal for the two outputs after 16 rounds cf. left hand side of Fig. 49 below. Both assumptions hold jointly with probability 2^{-100}. In practice, there is many such events for any fixed GOST key: there are 2^{127} possible pairs of plaintexts for the 16 rounds and inside these there will be about 2^{27} pairs which satisfy all the 50+50 difference bits.

Now the question is as follows: for 16 rounds of GOST, what is the probability that we have the additional 50 difference bits in the middle as shown on the right hand side of Fig. 49.

This probability is surprisingly high.

```
2 points
        A,B sharing 50 bits                 A,B sharing 50 bits
       0x80700700 0x80700700               0x80700700 0x80700700
                                                (8 Rounds)
            (16 Rounds)        becomes     0x80700700 0x80700700
                                                (8 Rounds)
       0x80700700 0x80700700               0x80700700 0x80700700
        C,D sharing 50 bits                 C,D sharing 50 bits

            50+50 bits                          150 bits
            2^28 events                         2^27 events
```

Fig. 49. Differential Induction: 50 additional differences nearly for free instead of 2^{-50}

Fact 130 (Induction Property For 16 Rounds). Given 16 rounds of GOST and regardless of the particular key scheduling, given 100 bits assumption as depicted on the left hand side of Fig. 49, the probability that we have additional 50 bits in the middle as shown on the right hand side Fig. 49 is very roughly about 50 % instead 2^{-50}.

Justification: This property is hard to verify experimentally but it is easy to establish by theory. As in Section 25.1 there are 2^{77} possible pairs with input differences in $0x8070070080700700$. Following [38] this propagates for 8 rounds and gives the middle difference in $0x8070070080700700$ with probability 2^{-25} and propagates for another 8 rounds with another 2^{-25}. Overall we predict that about $2^{77-50} = 2^{27}$ pairs which survive if the 2 propagations are independent. Moreover we claim that these 2^{27} pairs are going to be almost entirely disjoint with another 2^{27} pairs which occur by accident for any (random or not) permutation $Q \circ P$ as in Fig. 49 with all the 50+50 difference bits as on the left hand side of Fig. 49. The argument to show they are disjoint is similar as in Section 4 of [40]: out of 2^{27} pairs with the right 100 I/O differences, only a proportion of 2^{-50} will have the right 50 difference bits in the middle. The intersection of two sets of 2^{27} is expected to be empty. Therefore it is clear that for 16 rounds of GOST, we will get $2^{27} + 2^{27}$ events such as on the left hand side of Fig. 49, and out of these about half are such as on the right hand side of Fig. 49.

Remark: In practice, Induction works not only for the 50 bits which we study. The entropy of the whole difference on 64 bits after round 8 is also quite low. For example for about 50 % of the time it is only about 9 bits instead of $64 - 50 = 14$, see Section 26.1.

K.6 Stronger Induction - More Rounds

It is easy to see that stronger Induction, or Induction for more rounds, can be obtained if we reduce the space at both ends. We give here one example.

```
2 points
         A,B sharing 63 bits                     A,B sharing 63 bits
        0x80000000 0x00000000                   0x80000000 0x00000000
                                                     (10 Rounds)
              (20 Rounds)        becomes        0x80700700 0x80700700
                                                     (10 Rounds)
        0x00000000 0x80000000                   0x00000000 0x80000000
         C,D sharing 63 bits                     C,D sharing 63 bits

              128 bits                                178 bits
              2^-1 events                             2^-2 events
```

Fig. 50. Differential Induction: 50 additional differences nearly for free instead of 2^{-50}

This probability is again quite high.

Fact 131 (Induction Property For 20 Rounds). Given 20 rounds of GOST and regardless of the particular key scheduling, given 128 bits assumption as depicted on the left hand side of Fig. 50, the probability that we have additional 50 bits in the middle as shown on the right hand side Fig. 50 is very roughly about 50 % instead 2^{-50}.

Justification: For more or less any permutation, there is 2^{127} possible pairs and on average about 1/2 will have both an input difference and output difference in $0x8000000000000000$.

There are 2^{63} possible pairs with input differences in $0x8000000000000000$ and our computer simulations show that this propagates for 10 rounds and gives the middle difference in $0x8070070080700700$ with probability $2^{-25.5}$. This gives $2^{63-25.5} = 2^{37.5}$ possible pairs for the first 10 rounds. This means that each of the 2^{77} possible pairs in the middle propagates backwards with probability $2^{-39.5}$ on average. In the same way it propagates forwards with probability $2^{-39.5}$ on average. This will be about $2^{77-39.5-39.5} = 1/4$ or about 50 % of the 1/2 obtained above.

K.7 Induction With Three Points

In the same way we can consider Induction with 3,4 and more points. We present here just one particular way to do it in which all differences of 2 points are constrained.

When we write $0x80700700, 0x80700700$ for 3 points we mean all the 3 points share the same 50 bits at the input, or equivalently that every difference has 14 active bits: $A \oplus B \in 0x8070070080700700$, and the same for $A \oplus C$ and $B \oplus C$. Now the initial constraint on 3 points on the left hand side has an entropy of 200 bits, and if it happens that the 3 middle differences share the same 50 bits, this is additional 100 bits. For an ideal (very strong) cipher it would occur with conditional probability of 2^{-100}.

```
3 points
        A,B,C sharing 50 bits                 A,B,C sharing 50 bits
        0x80700700 0x80700700                 0x80700700 0x80700700
                                                   (8 Rounds)
              (16 Rounds)         becomes     0x80700700 0x80700700
                                                   (8 Rounds)
        0x80700700 0x80700700                 0x80700700 0x80700700
        D,E,F sharing 50 bits                 D,E,F sharing 50 bits

              200 bits                                300 bits
          2^-3+2^-11 events                        2^-3 events
```

Fig. 51. 3-Point Higher-Order Differential Induction: 100 additional differences obtained with near certitude instead of 2^{-100} expected for an ideal cipher

In a preliminary estimation, this probability appears to be surprisingly high.

Fact 132 (3-Point Induction Property For 16R). Given 16 rounds of GOST and regardless of the particular key scheduling, given a 200 bits I/O constraint as depicted on the left hand side of Fig. 50, the probability that we have additional 100 bits in the middle as shown on the right hand side Fig. 51 is very roughly about 99.5 % instead 2^{-100}.

Justification: Again it is difficult to estimate the probability exactly. For more or less any permutation, there is $2^{192}/6 \approx 2^{189.4}$ possible triples and on average about $2^{200-189.4} \approx 2^{-10.6}$ will have the same 50 points at the input and at the output (for all the three encryptions).

Given any fixed 50 bits and 8 rounds of GOST, our computer simulations show that with probability 2^{-6} there are 3 points sharing these 50 bits at the input, and sharing some other 50 bits at the output. Then our computer simulations show that the first two propagate for the next 8 rounds with probability of about 2^{-25}, and the third also has the same 50 bits with conditional probability of about 2^{-22}. Overall, we expect to have the situation as on the right hand side of Fig. 51 for very roughly about $2^{50-6-25-22} \approx 2^{-3}$ triples of points. This 2^{-3} will be about 99.5 % compared to the additional $2^{-10.6}$ obtained above.

Remark. It appears that triple point events impose very strong constraints. Or equivalently, with 3, 4 or more point events, we can impose stronger constraints on what happens inside the cipher. This means that with 3 points we expect to get Induction properties which work for more rounds than with 2 points, and with 4 points even better Induction properties.

K.8 Induction For 32 Rounds

For 32 rounds we can assume that 50 bits are shared at the input, the output and in the middle which happens for $2^{127-50-50-50} = 2^{-23}$ pairs A, B. Then we can apply Fact 130 twice, and we obtain about $2^{-23} \cdot 0.5 \cdot 0.5 \approx 2^{-25}$ pairs as on the right hand side of Fig. 52.

```
2 points
        A,B sharing 50 bits                   A,B sharing 50 bits
       0x80700700 0x80700700                 0x80700700 0x80700700
                                                  (8 Rounds)
                                              C,D sharing 50 bits
             (16 Rounds)                     0x80700700 0x80700700
                                                  (8 Rounds)
        E,F sharing 50 bits                   E,F sharing 50 bits
       0x80700700 0x80700700    becomes      0x80700700 0x80700700
                                                  (8 Rounds)
                                              G,H sharing 50 bits
             (16 Rounds)                     0x80700700 0x80700700
                                                  (8 Rounds)
       0x80700700 0x80700700                 0x80700700 0x80700700
        I,J sharing 50 bits                   I,J sharing 50 bits

             150 bits                              250 bits
           2^-23 events                          2^-25 events
```

Fig. 52. Differential Induction for 32 Rounds

This is an interesting configuration, mainly because it is partly "visible" to the attacker: for any given key there is $2^{127-50-50} = 2^{27}$ pairs A, B for which the outputs coincide on 50 bits. The attacker can guess which pair A, B is correct with probability 2^{-27}.

For a proportion of 2^{-25} of keys, and with probability 2^{-27} the attacker can determine 150 bits inside the cipher. This is quite good. However it is not easy to see how this can be exploited in an attack.

K.9 Weak Induction For 32 Rounds

In this section we describe one example which is substantially weaker in terms of the induction probabilities obtained. The advantage is that the attacker does not make any assumptions on the internal values after 16 rounds, only on the inputs and the outputs.

```
2 points
         A,B sharing 60 bits                    A,B sharing 60 bits
        0xF0000000 0x00000000                  0xF0000000 0x00000000
                                                    (8 Rounds)
                                                C,D sharing 50 bits
                                               0xF0700700 0x80700700

            (32 Rounds)            becomes          (16 Rounds)

                                                E,F sharing 50 bits
                                               0x80700700 0x80700700
                                                    (8 Rounds)
        0x00000000 0xF0000000                  0x00000000 0xF0000000
         G,H sharing 60 bits                    G,H sharing 60 bits

              120 bits                               220 bits
             2^7 events                            2^-40 events
```

Fig. 53. Weak Differential Induction for 32 Rounds

Part XIII

Authentication and Integrity Applications of GOST

L GOST Hash and Self-Similarity Attacks

In this section we introduce a comprehensive framework which allows to discuss some existing attacks on the GOST Hash compression function [67, 68, 72, 73, 47]. We want to define a framework which allows one to split the task of the cryptanalysis of the GOST Hash function, or just for the GOST compression function [72, 73, 47] into several independent welly-defined tasks which can be studied separately. In this paper we only look at pseudo-collisions and pseudo-pre-images for the compression function. Attacks on the full GOST Hash function are outside of the scope of this paper, see [73]. Our goal is to see if the self-similarity of the high-level structure of GOST hash can be exploited to find interesting attacks.

L.1 Cryptanalysis of GOST Compression Function

On the following picture we provide a high-level description of the GOST compression function. The fact that the the GOST compression function can indeed be represented in this way is NOT trivial and requires to carefully read the specification of the hash function [67, 68, 72, 73].

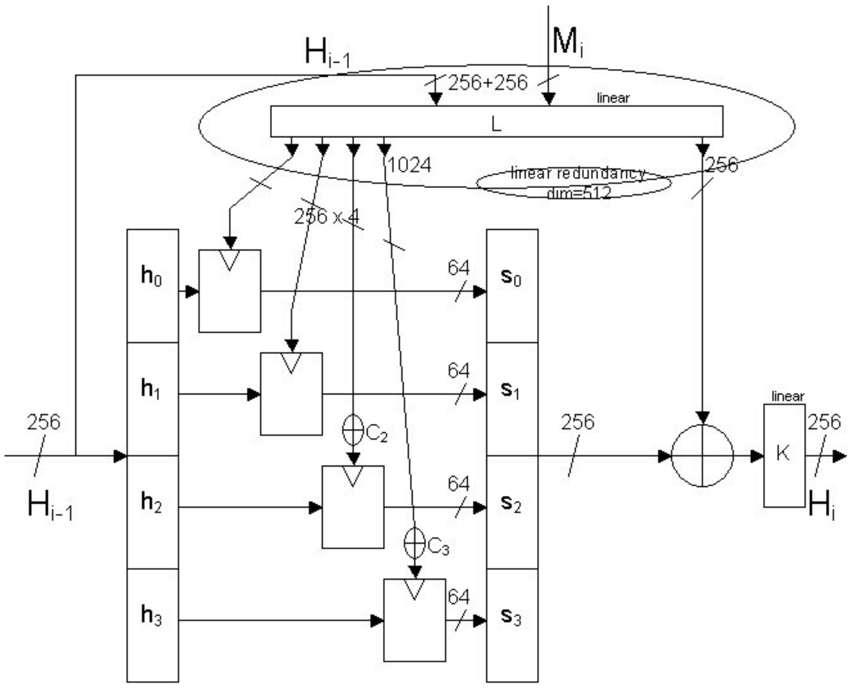


Fig. 54. One round of the GOST compression function

1. 4 instances of the full 256-bit GOST cipher,
2. One linear transformation L with 256+256 inputs and 1024+256 output bits.
3. Two 256-bit affine constants C_2 and C_3, other parts being linear.

4. One bijective linear transformation K on 256 bits which does not affect any of our attacks on one single GOST compression function however it would be important to study if we want to attack a composition of such functions.

All the attacks on this function can be derived in this abstract algebraic representation and do not depend on details of individual components.

L.2 The Main Framework and Its Applications

The main idea which unifies different attacks on this compression function is as follows. It can be seen as the special case of Constrained Inputs Constrained Outputs (CICO) attack, which is a natural and popular methodology in cryptanalysis of hash functions. Here the constraints of inputs and outputs are always sets of linear equations. This sort of property is more or less the only thing we need to develop attacks on the GOST compression function.

Definition L.2.1 (Affine CICO Reduction Property).

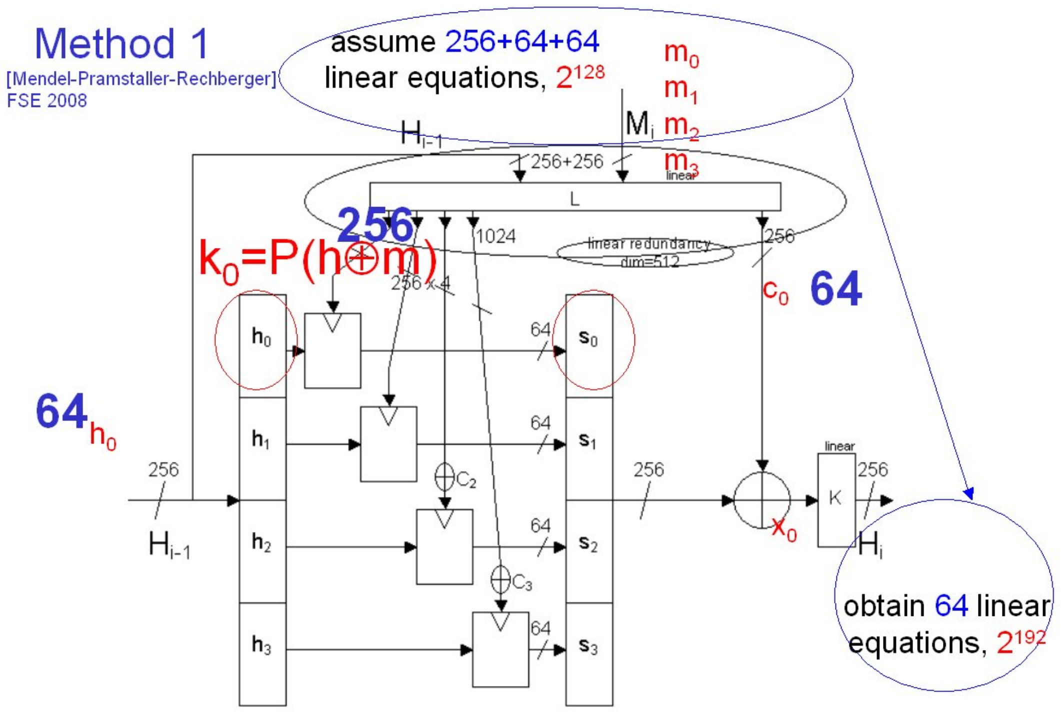
1. We assume some 256+64+64=256+256-128 linear equations to hold on the two inputs at the top of the picture: H_{i-1} and M_i.
   This reduces the input space to 2^{128} inputs.
2. Then we observe that there will be, due to specific reasons explained later, 64 linear equations true with probability 1 on the output H_i.
   In other terms if we choose carefully our linear assumptions, the output falls within a LINEAR space with 2^{192} elements EACH TIME the input is a member of the reduced input linear space with 2^{128} inputs.

There are two different methods to obtain this Affine Reduction Property.

L.3 Description of Method 1

This method of achieving our objective (Affine Reduction Property) comes from [72] and works as follows (cf. Fig. 55).

1. We fix some target on 64 bits, for example $x_0 = a$.
2. We fix the key k_0 for the first instance of the cipher GOST.
   This is 256 linear equations on the two inputs at the top H_{i-1} and M_i.
3. We fix the input h_0 of the first instance of the cipher GOST. This is another 64 linear equations.
4. We fix c_0 (cf. Fig. 55). We fix it to the value $a \oplus s_0$ in order to achieve our target $x_0 = a$. This is another 64 linear equations.
5. This allows one to make sure that once s_0 is fixed and c_0 is fixed, 64 bits of the input of K are fixed, namely $x_0 = a$, and therefore the output H_i lives in a linear space of dimension 2^{192} as required.



Method 1
[Mendel-Pramstaller-Rechberger] FSE 2008
assume 256+64+64 linear equations, 2^128
m0 m1 m2 m3
Mi
Hi-1
256+256
linear
L
256
k0=P(h⊕m)
1024
linear redundancy dim=512
256 x 4
256
c0
64
h0 h1 h2 h3
s0 s1 s2 s3
64h0
256
Hi-1
C2
C3
x0
linear
K
Hi
obtain 64 linear equations, 2^192

Fig. 55. Method 1 to achieve our objective of "Affine Reduction"

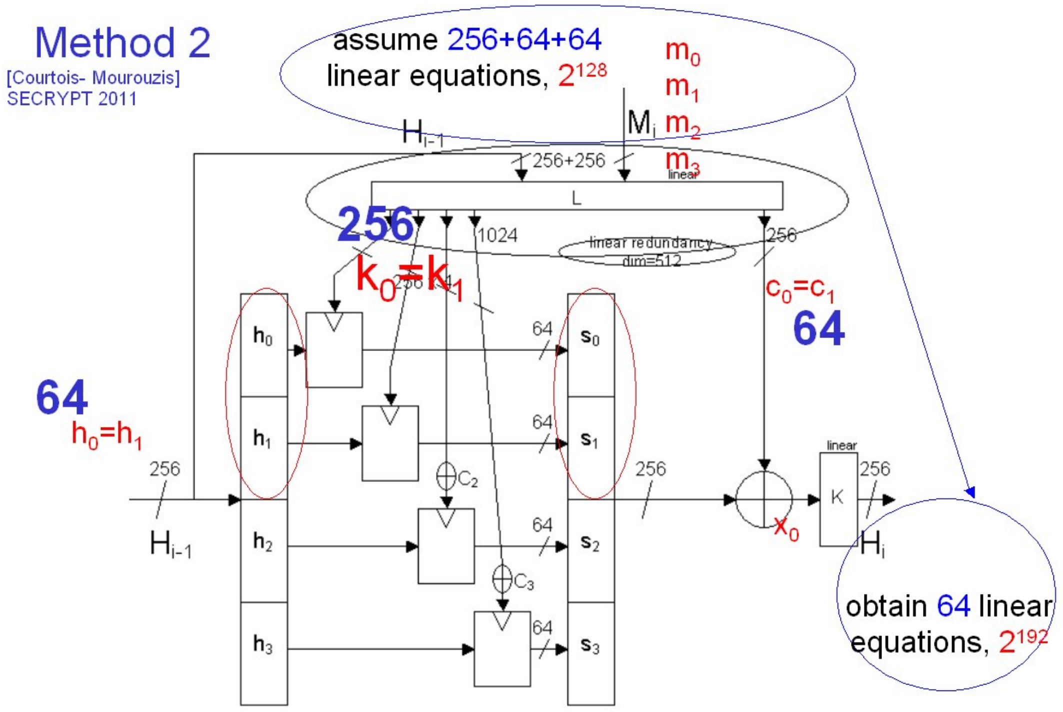


Fig. 56. Method 2 to achieve our objective of "Affine Reduction"

L.4 Description of Method 2 Based On Self-Similarity

This method of achieving our objective (Affine Reduction Property) comes from [47] and works as follows (cf. Fig. 56).

1. We assume that keys k_0 and k_1 for the first two instances of the cipher GOST are the same.
   This is 256 linear equations on the two inputs at the top H_{i-1} and M_i.
2. We assume that the inputs k_0 and k_1 for the first two instances of the cipher GOST are the same.
   This is another 64 linear equations.
3. We fix $c_0 \oplus c_1$ (cf. Fig. 56). This is another 64 linear equations.
4. This allows one to make sure that once $s_0 = s_1$ we obtain 64 linear equations on the input of K are fixed, and therefore the output H_i lives in a linear space of dimension 2^{192} as required.

L.5 How to Find Pseudo-Collisions for the GOST Compression function

This is possible very easily with both Method 1 and Method 2 and we recall the methods of [72, 73, 47].

1. With both Method 1 and Method 2 our output space has 2^{192} elements.
2. By birthday paradox we can find colliding inputs in time of 2^{96}.
3. Our input space is larger than 2^{96} which is sufficient.
4. The memory complexity can be made negligible by very widely known "memoryless" cycling techniques, see [72, 86].

A more powerful method to find collisions which are no longer pseudo-collisions and where almost the whole H_i can be "almost" an arbitrary value is described in [73].

L.6 How to Find Pseudo-Pre-Images for the GOST Compression Function in Time 2^{189}

This is possible with Method 1 and already done in [72]. We revisit this attack.

1. Again let $(x_0, x_1, x_2, x_3) = (a, b, c, d)$ be our target.
2. With Method 1, cf. Fig 55, we first chose any k_0, h_0, compute s_0 with one GOST encryption and put $c_0 = s_0 \oplus a$ which allows us to achieve $x_0 = a$.
3. This triple of values h_0, k_0, c_0 gives 256+64+64 linear equations on 512 input bits.
4. We can compute a basis for a linear space of dimension 128.
5. We can then try 2^{128} cases in this space. We need to repeat this 2^{64} times for different initial choice of h_0, k_0.
6. We need to try 2^{191} cases on average until we get $(x_1, x_2, x_3) = (b, c, d)$.
7. In each of 2^{191} steps we need to compute just one GOST encryption and most of the time we can reject this case because $x_1 \neq b$. This is only 2^{-2} evaluations of the GOST compression function.
8. Overall the expected running time $2^{191-2} = 2^{189}$ evaluations of the GOST compression function.

Here also a more powerful method to find pre-images which are no longer pseudo-pre-images and where almost the whole H_i can be "almost" an arbitrary value is described in [73].

L.7 Can We Find Pseudo-Pre-Images with Method 2?

In this section we look if it is possible to find pseudo-pre-images with a variant of our Method 2 which was successful with finding pseudo-collisions.

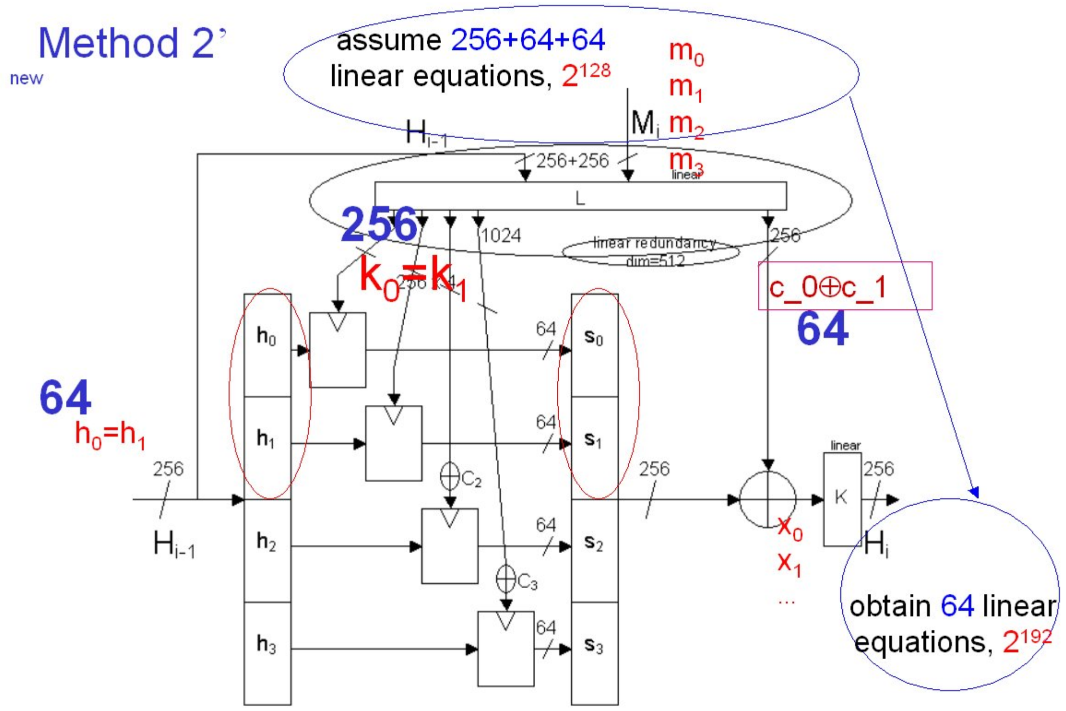


Fig. 57. Tentative Method 2' to find pseudo-pre-images for the GOST Compression function

1. With our new Method 2', cf. Fig 57, we can first of all choose $c_0 \oplus c_1$ to be such that when $s_0 = s_1$ we get the current target value for $x_0 \oplus x_1$.
   This is 64 linear equations on the inputs.
2. Then with other assumptions being the same as in Method 2, $k_0 = k_1$ and $h_0 = h_1$, a random input produces the output we want with probability 2^{-192}.
3. For one given k_0 the input space is of insufficient size 2^{128}.
   This attack only works with probability 2^{-64} over the target outputs.
4. With 6 possible choices of 2 out of 4 we get $2^{-61.4}$ over the target outputs.
5. We get another $2^{-61.4}$ if we consider the complementation property of GOST described in Appendix C of [50]: instead of assuming that $k_0 = k_1$ and $h_0 = h_1$, we assume that $k_0 = k_1 \oplus e_{31}$ and $h_0 = h_1 \oplus e_{31}$, and we XOR e_{31} also to our computed target for $c_0 \oplus c_1$.
6. Overall we achieve a proportion of $2^{-60.4}$ of target outputs.
7. This attack only works for some target outputs and allows to find pseudo-pre-images for them in time of 2^{192}. This is not as good as with Method 1. It seems that it is easier to design a pseudo-collision attack for GOST than a pseudo-pre-image attack.