An Improved Differential Attack on Full GOST

Nicolas T. Courtois

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

Abstract

GOST 28147-89 is a well-known block cipher and the official encryption standard of the Russian Federation. A 256-bit block cipher considered as an alternative for AES-256 and triple DES, having an amazingly low implementation cost and is becoming increasingly popular [17, 13]. Until 2010 researchers unanimously agreed that: "despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken", see [17] and in 2010 it was submitted to ISO 18033 to become a worldwide industrial encryption standard. In 2011 it was suddenly discovered that GOST can be broken and is insecure on more than one account. There is a substantial variety of recent attacks on GOST [3, 8, 14, 6, 4, 5]. We have reflection attacks [14, 8], attacks with double reflection [8], self-similarity guess then determine attacks which do not use any reflections [8, 3] and advanced differential attacks [18, 6, 4, 5]. The final key recovery step in various attacks is in many cases a software algebraic attack [8, 3], frequently also a Meet-In-The-Middle attack [14, 8, 9] and in differential attacks key bits are guessed and confirmed by the differential properties [18, 6, 4, 5].

Ignoring the very large data complexity of most of these attacks, and comparing only the time, until October 2011 the fastest attacks on GOST were two attacks by Courtois with the same complexity of 2^{216} [8]. Then Shamir *et al* presented an improved alternative final step for one of these attacks and obtained an attack on GOST with overall time complexity of about 2^{192}, cf. FSE 2012 [9]. However arguably, the best attack on full GOST when used in encryption with random keys is another attack in 2^{185} described in Appendix L of [8].

In this paper we consider some recent differential attacks on GOST [18, 6, 4, 5] and show how to further improve them. We present a new single-key attack against full 32-round 256-bit GOST with time complexity of 2^{178} which is substantially faster than any previous result.

Key Words: Block ciphers, GOST, differential cryptanalysis, sets of differentials, aggregated differentials, iterative differentials

1 Outline and Main Stages

This paper describes essentially one advanced differential attack on GOST.

1.1 Key Schedule in GOST

The key structural property of GOST which makes it suitable for cryptanalytic attacks of the specific kind and specific form, is that the last 8 rounds are identical to the fist 8 rounds run in the opposite direction (however this symmetry does not follow for more inner rounds).

rounds	1							8	9							16
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
rounds	17							24	25							32
keys	$\mathbf{k_0}$	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	$\mathbf{k_0}$

Figure 1: Key schedule in GOST

This property has a big impact on security of GOST and is already exploited in [4, 5]. Because 32 bits of the whole key, a fairly small proportion, is used in one round, and for every 32 bits guessed we can remove two full outer rounds, instead of 1 round for a similar cipher without a weak key scheduling. Thus for example if we guess 192 key bits, we can remove 12 full rounds of GOST and still get an attack faster than brute force, see [4].

In this paper we will exploit this symmetry even further: we will look at differential which are a member of a certain set of differentials which is totally symmetric for the first 8 rounds and the last 8 rounds. Our key guesses will be far more precise than guessing keys for full rounds and specially adapted to this highly symmetric situation, in order to maximize the number of cases which can be safely discarded due to these guesses.

1.2 Preliminary Remarks

In our attack we will split GOST into three parts with 6+20+6 rounds. Previous advanced differential attacks were based on a statistical distinguisher which exploited a number of differential propagations to distinguish up to 20 rounds of GOST from a random permutation [4, 5] and needed to guess complete 32-bit keys for several outer rounds in order to fully reconstruct these internal differentials.

We will guess only some well-chosen key bits which will be used to filter P/C (Plaintext,Ciphertext) pairs used later in the attack. As in [4] the attack runs through many stages with great many filtering/guessing steps, where at each step we reduce the number of cases to consider (the plaintext space, some key bits already guessed and pre-computed relations between all these) and only after this reduction of number of cases we make additional guesses. In contrast to the attack presented in [5] our attack will use symmetric differential sets, which is dictated by the symmetry in the key schedule.

Large parts of this whole process can be viewed as an adaptive Depth-First-Search (DFS) attack on a tree of possibilities which is constructed adaptively depending on the assumptions currently considered as valid. This type of process is very widely used in cryptanalysis.

There is a substantial difficulty in differential attacks where the key size is much larger than the block size as in GOST: there are false positives, differentials which do not propagate but occur naturally, by accident. The key point is that for a very long time the false positives are not eliminated in a differential attack on GOST. In fact, for a long time we are just dealing with assumptions on internal difference bits in GOST, their consequences and relations between these assumptions but for a long time, none of the steps of the attack is able to see if the inner 20 rounds are 20 rounds of GOST, more rounds of GOST, or maybe just a random permutation. This can only be seen at a later stage of the attack.

2 The Structure of GOST

We will only study, by far most popular set of GOST S-boxes known as the "GostR3411_94_TestParamSet" in [13] which was published as early as in 1994 and which according to Schneier [19] is actually the version used by the Central Bank of the Russian Federation. This is exactly what most researchers call just "the GOST cipher" (without any additional mention) in the cryptographic literature. This choice of S-boxes greatly affects all the differential probabilities we use in this paper.

Note: It is possible to see that similar attacks exist for other variants of GOST.

2.1 Notation

In this paper we consider the traditional "Twisted" representation of GOST where each round of GOST looks exactly the same:

$$(L, R) \mapsto (R, L \oplus f_k(R))$$

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 by numbers between $(11 + 4i \bmod 32) + 1$ and $(11 + 4i + 3 \bmod 32) + 1$. For example the inputs of S6 are $20, 21, 22, 23$ and the outputs are $32, 1, 2, 3$.

On our picture below (Fig. 2) the $\boxplus$ denotes the addition modulo 2^{32}. On this picture we do NOT represent the final circular shift by 11 positions modulo 32 which occurs in GOST after the S-boxes. It is represented in a different way, by numbering the output bits of S-boxes, to see directly where they are connected.

GOST has 32 rounds such as one described on Fig. 2.

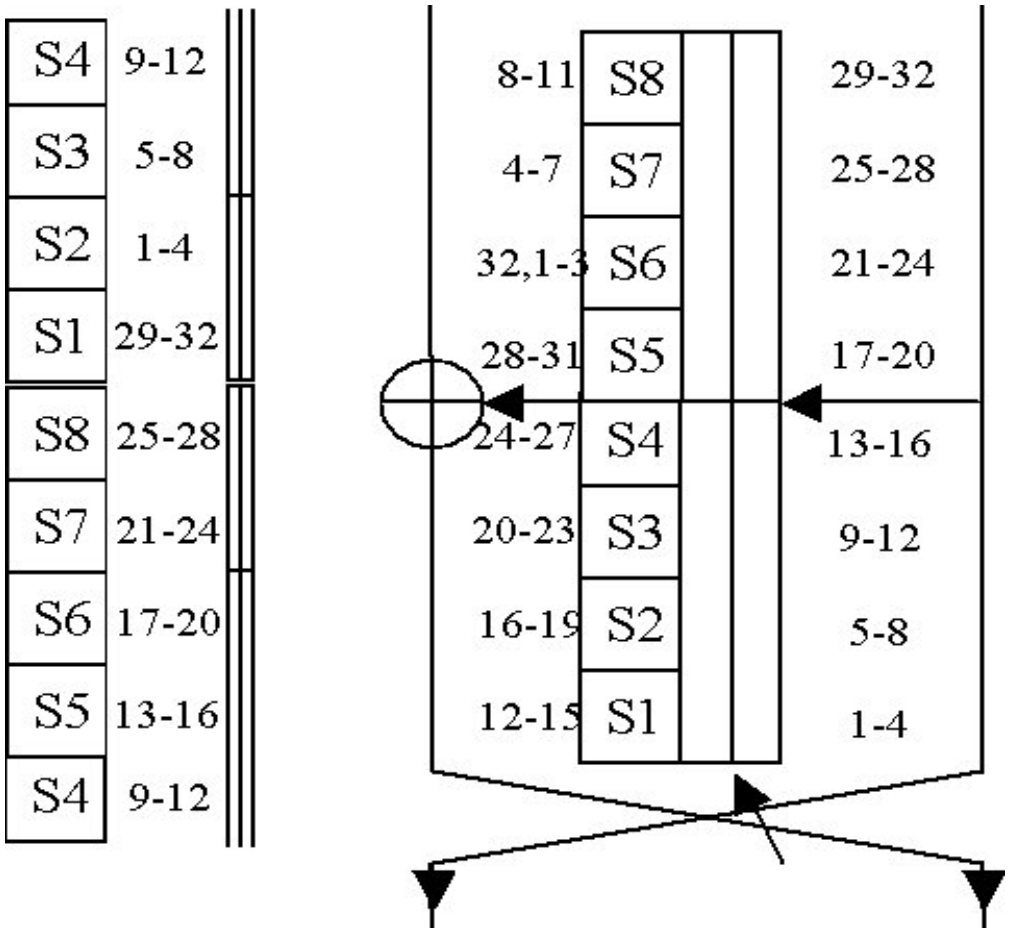

Figure 2: One Round of GOST And Connections in The Following Round

Given the key scheduling on Fig. 1 we have a complete description of GOST.

At the left margin on Fig. 2 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. A more detailed explanation of how these bits in the next round depend on the bits in the previous round will be developed later on Fig. 7.

2.2 Aggregated Differentials in GOST

For a detailed history, vocabulary discussion and literature survey on the development of advanced differential attacks on GOST we refer to [4, 5]. In this paper we start with some advanced properties previously identified by researchers.

We define *an aggregated differential* A, B as the transition where any non-zero difference $a \in A$ will produce an arbitrary non-zero difference $b \in B$ with a certain probability. In particular we consider the case when A is a set of all possible non-zero differentials contain within a certain mask. For example for

$$\Delta = 0x80700700$$

we obtain a set of all differences with between 1 and 7 active bits (but not 0) and where the active bits are contained within the mask 0x80700700. Similarly, the set denoted by (Δ, Δ) is a set of 14 active bits, where any non-zero difference is allowed, including also differences where the difference is zero on one half. There are $2^{14} - 1$ differences in this set and we have $|A| = |B| = 2^{14} - 1$. The following fact was established in [6]:

Fact 2.2.1 *The aggregated differential* (Δ, Δ) *with uniform sampling of all differences it allows, produces an element of the same aggregated differential set* (Δ, Δ) *after 4 rounds of GOST with probability about* $2^{-13.6}$ *on average over all possible keys.*

For 6 rounds the probability is $2^{-18.7}$ *on average over all possible keys.*

Now we look at a one particular differential set, which we have noticed, arrives with a particularly large probability:

Fact 2.2.2 *The set* $(0x80700700, 0x80700700)$ *produces a differential of the form* $(0x00000700, 0x80780000)$ *with probability of* $2^{-22.19}$ *for 7 rounds of GOST.*

This was obtained by a computer simulation.

3 Propagation of Differentials in GOST

It is possible to see that among differentials of the form $(0x80700700, 0x80700700)$ there exist several interesting non-trivial classes. On Fig 3 we denote these classes by 6-letter abbreviations for example $(0x00000700, 0x80700000)$ would be abbreviated as _787_). It is the possible to see that not all transitions are possible. This leads to a certain particular graph of possible transitions which can be used to find efficient differential attacks on GOST, see Fig 3.

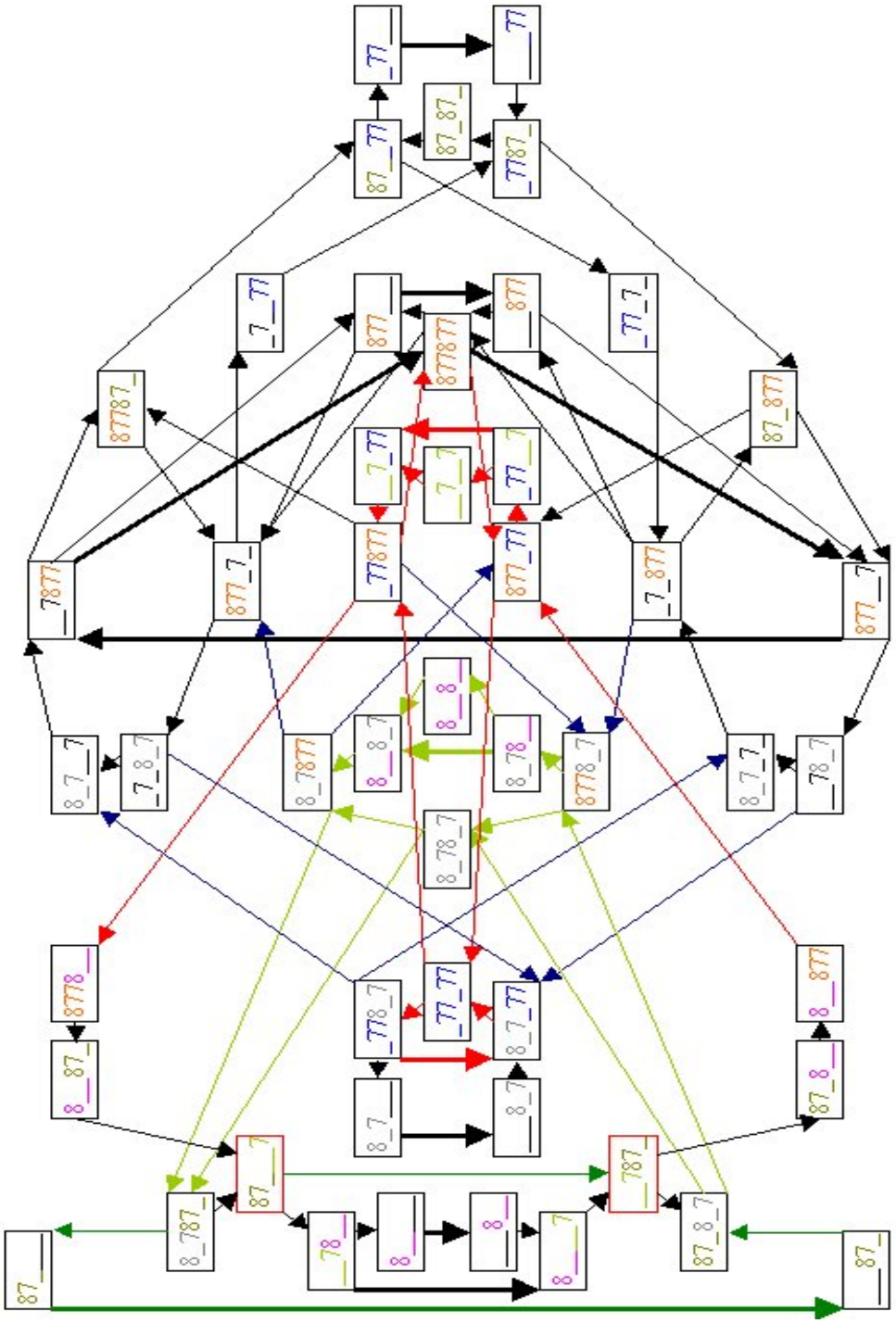

Figure 3: Some Important Sets of Differentials Which Occur in GOST

3.1 On Important Classes of Aggregated Differentials in GOST

It is possible to see that many very good differential attacks on GOST are related to the important structure which exists within GOST. Basically, half of GOST S-boxes are very closely connected, and loosely connected with the other half, which is shown on Fig. 4.

In this respect some very good attacks on GOST known to us can be classified in several major categories which we call 8+0, 5+1, and 3+3.

Differential Attacks of Type 3+3: The meaning of 3+3 is that the attack uses ONLY 3 fixed S-boxes from Fig. 4, for example S3,S6,S8, and another fixed 3 S-boxes from the complement of Fig. 4, and no other S-boxes. Moreover all these S-boxes are connected in a cyclical structure such that non-zero differences circulate within the structure, and do not leave it. The structure must be closed and circular because the GOST S-boxes are bijective and non-zero differentials cannot be canceled totaly. Many interesting differentials we exploit in this paper are of the type 3+3.

Other non-trivial interesting classes of differentials exist in GOST. We give one example of a differential set of type 1+5. It is as follows: 0x07070008,0x08007070. This set of $2^{14} - 1$ single differentials propagates for 8 rounds with an average probability of $2^{-26.0}$ for 8 rounds of GOST.

Discussion: The key point we want to make here is that these classes of differentials do NOT depend that much on the S-boxes, they depend much more on the structure of the cipher and the exact connections inside. This is especially true for sets which include many differentials such as studied in this paper where individual very strong differentials will matter less.

A common misconception is that that the security of a cipher such as GOST against differential cryptanalysis (DC) depends extensively on the S-boxes, and some "optimal" S-boxes would make it more secure, see [16, 17]. Researchers in the academia study S-boxes in isolation [16] and propose new ciphers [16, 17] but when S-boxes are connected together, everything changes. We are **not certain if it is possible at all** to make a cipher such as GOST secure against differential cryptanalysis by changing only the S-boxes [16, 17], which idea was discussed during the ISO standardization process of GOST. The question of how far one can go with advanced differential attacks such as studied in this paper remains widely open.

We conjecture that the security of GOST against advanced forms of DC depends "essentially" on the connections on the cipher, and though some S-boxes make it weaker against DC, none will make it really very strong.

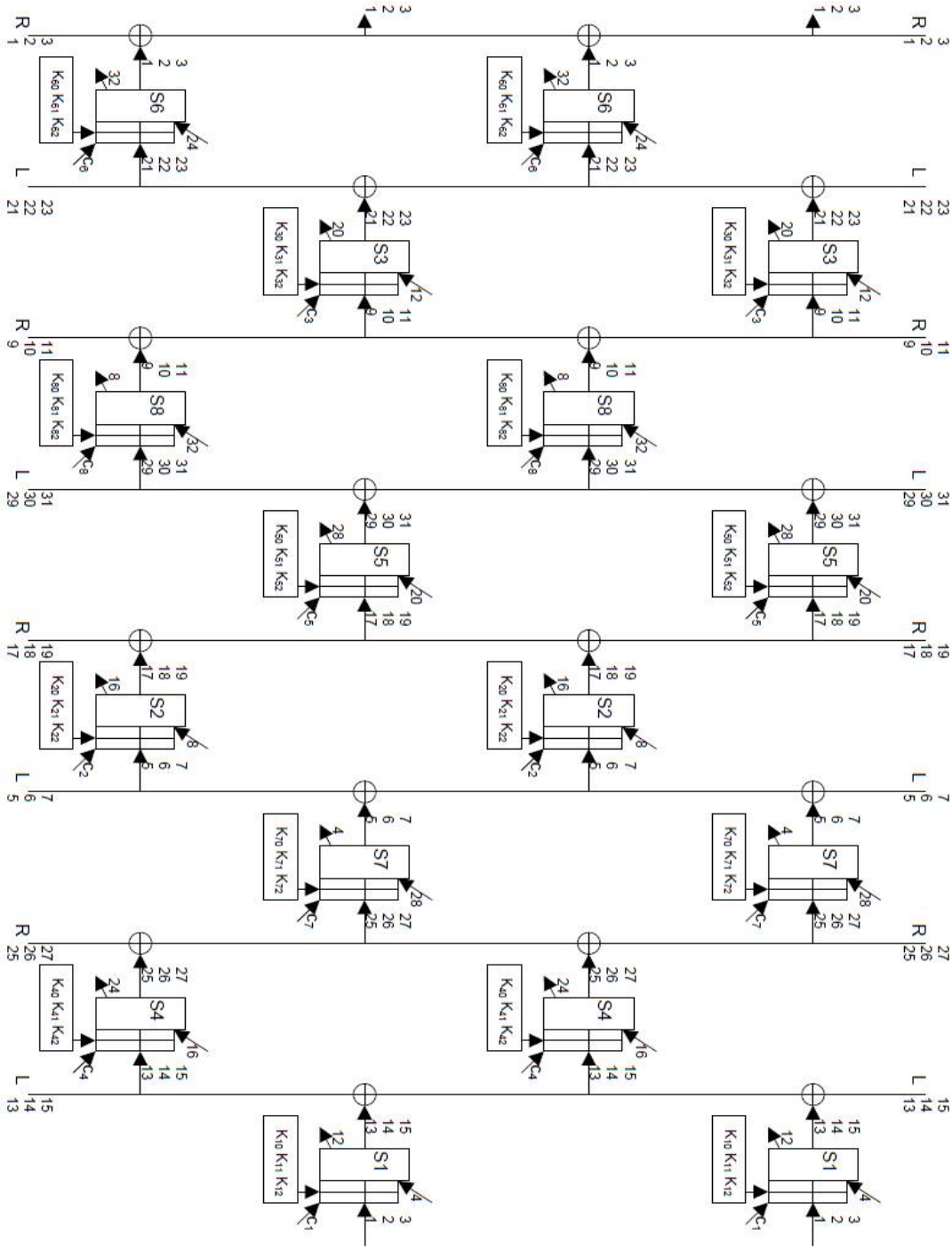

Figure 4: Connections between half of S-boxes in 4 consecutive rounds

4 Improved Differential Attacks On GOST

Our goal is to design an attack on the full 32-round GOST. We will use the same methodology as in [4, 5] guess some key bits and use a distinguisher and we also design a distinguisher for 20 rounds. However our new distinguisher is symmetric and the attack is more complicated.

The key question is how a differential attack on GOST can cope with false positives. There are differentials which occur due to propagation of small Hamming weight differentials for 20 rounds of GOST, and other which occur "by accident" for an arbitrary permutation on 64 bis. In addition we also need to quantify precisely the interaction between these two sets, which is essential in we want to reliably distinguish between 20 rounds of GOST and some other permutation. We have:

Fact 4.0.1 *We look at the combination of a non-zero input difference of type* $(0x80780000, 0x00000700)$ *and a non-zero output difference of type* $(0x00000700, 0x80780000)$.

For a typical permutation on 64-bits (does NOT have to be a random permutation, can be GOST with more rounds) we expect that there are 2^{15} *pairs* P_i, P_j *with such differences. The distribution of this number can be approximated by a Gaussian with a standard deviation of* $2^{7.5}$.

For 20 rounds of GOST and for a given random GOST key, there exists two disjoint sets of $2^{15} + 2^{13.9}$ *such pairs* P_i, P_j.

These are two entirely disjoint sets of pairs, which can be distinguished by the fact that $2^{13.9}$ *pairs will have the difference* $0x80700700, 0x80700700$ *after 6 rounds from the beginning AND 6 rounds from the end, and none of the* 2^{15} *will have such internal differences.*

The distribution of the sum can be approximated by a Gaussian with an average of about $2^{15} + 2^{13.9}$ *and the standard deviation of* $2^{7.8}$.

Justification: For any permutation, we observe that every single combination of an input differential on 64 bits, and on an output differential on 64 bits, is expected to occur about 0.5 times on average. Indeed we have 2^{127} pairs and about 2^{128} possible sets of two differentials. Now we have $(2^8 - 1)(2^8 - 1)$ possibilities within the internal 20 rounds: $(0x80780000, 0x00000700) \rightarrow (0x00000700, 0x80780000)$. Overall we expect to obtain $0.5 \cdot 2^{8+8} = 2^{15}$ pairs P_i, P_j for a given GOST key, with any of these $2^8 - 2^3 + 1$ differences.

For 20 rounds the situation is more complex. We need to distinguish between pairs which occur "by accident" and those which occur due to "propagation". We are going to develop a precise argument showing that both sets are entirely disjoint and their numbers can be added. In order to do this we are going to give a precise meaning to the word "propagation"

in this precise 20 rounds case: we say that the differential "propagates" if it goes through two additional differences in the middle as follows:

```
0x80780000 0x00000700
     (7 Rounds)
0x80700700 0x80700700
     (6 Rounds)
0x80700700 0x80700700
     (7 Rounds)
0x00000700 0x80780000
```

Figure 5: "Propagation" For 20 rounds With Specific Middle Differentials

Following Fact 2.2.1 and given $2^{64+14-1}$ pairs with the initial difference, we have $2^{77-18.7} = 2^{58.3}$ pairs for the middle 6 rounds.

Then following Fact 2.2.2 the propagation in the next 7 rounds occurs with probability $2^{-22.2}$ on average over GOST keys. Since this is a permutation, the same propagation can be applied backwards in the preceding 7 rounds. Overall, we expect that $2^{58.3-44.4} = 2^{13.9}$ pairs survive.

Now we are going to show that typically, none of these $2^{13.9}$ pairs P_i, P_j is a member of the set of 2^{15} established beforehand. This can be established as follows: for any of the 2^{15} cases which occur naturally at random, we have a non-zero input differential $(0x80780000, 0x00000700)$. Then a computer simulation shows that a differential of type $(0x80700700, 0x80700700)$ CAN occur at 7 rounds from the beginning (as on Fig. 5 which is 6+7 rounds from the beginning in GOST) but only with probability of $2^{-16.2}$. Similarly it can also occur 7 rounds from the end, but only with probability of $2^{-16.2}$. Overall we expect that only about $2^{15-16.2-16.2} = 2^{-17}$ pairs P_i, P_j on average will have the "propagation" characteristics according to Fig. 5. Therefore the two sets are entirely disjoint with a very high probability.

To summarize, we expect to get always a mix of $2^{15} + 2^{13.9}$ cases, which are unlikely have an intersection, just subject to the standard deviation for each set. Because we are dealing with a sum of a very large number of almost totally independent events, and exactly in the same way as in [4, 5], and due to the Central Limit Theorem [20] these numbers are expected to follow a Gaussian distribution and the standard deviation is expected to be equal exactly to the square root of their expected average number which will be about $2^{15.55}$ for 20 rounds and about $2^{15.0}$ for other permutations.

Related Research We can compare this result to a distinguisher which is used in [5]: where the input difference is $(0x80000000, 0x00000000)$, and the output difference is of the form $(0x00000100, 0x80600200)$. In the case of the actual 20 rounds of GOST we expect to get $2^4 + 2^5$ such differences, while 2^4 would occur naturally at random for some other permutation.

4.1 Extending with Additional Weakly Constrained Rounds

This property can be extended with a "weakly constrained" differential propagation which occurs with quite a high probability for 6 more rounds on each side. More generally, we can look at combination of 6 rounds of GOST, some permutation, and 6 rounds of GOST with the same keys in the backwards direction, as in GOST. This could be just the full 32-round GOST with 20 rounds in the middle. But it could also be a different situation which we wrongly assumed to be the full 32-round GOST.

Definition 4.1.1 (Alpha Property) *We say that a pair of encryptions for the full 32-round GOST (or for a combination of 6 rounds of GOST, some permutation, and 6 rounds of GOST with reversed keys) has the Alpha Property if the following whole configuration of sets of differentials simultaneously holds:*

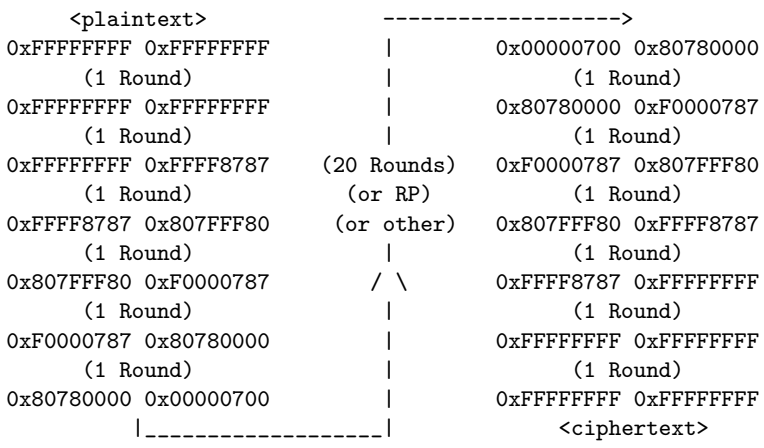

Figure 6: The Alpha Property

We note that this property is perfectly symmetric.

4.2 Alpha Property: GOST vs. Random Permutation

In a similar way as before, a key problem in our distinguisher is that unhappily the Alpha property can occur also "by accident", not at all for the reasons we expect. This question needs to be formulated more precisely, as this property is about differentials also inside GOST, and therefore we cannot just compare GOST to a random permutation. The right question which we need to ask is as follows: in our composition of 6 rounds of GOST, some permutation and then the same 6 rounds in the decryption mode, can we have a fully consistent situation with all the differences which we have in the property Alpha on the outer 2x6 rounds, exactly like on Fig. 6?

We have the following result:

Fact 4.2.1 *For the full 32-round GOST and on average over the GOST keys, there exists* $2^{13.0} + 2^{11.9}$ *distinct pairs of plaintexts* $P_i \neq P_j$ *which have the Alpha property.*

If we replace the inner 20 rounds by a random permutation or with GOST with more rounds, we expect only about $2^{13.0}$ *distinct pairs with a standard deviation of* $2^{6.5}$.

Justification: We apply Fact 4.0.1 and obtain $2^{15} + 2^{13.9}$ pairs for the inner 20 rounds with two disjoint sets as explained before. Then it is easy to verify, by a computer simulation, that this provokes the 6 difference sets in the following 6 rounds, simultaneously, with probability as large as $2^{-0.98}$, which is due to slow diffusion in GOST. The same applies in the first 6 rounds. Overall we obtain about $2^{13.9-2.0} \approx 2^{11.9}$ pairs with propagation, and a disjoint set (because subsets of disjoint sets from Fact 4.0.1) with $2^{15-2.0} \approx 2^{13.0}$ pairs which occur by accident.

Again, because we are dealing with a sum of many almost totally independent events, as in [4, 5] and due to the Central Limit Theorem [20], the standard deviation is expected to be exactly the square root of $2^{13.0}$,

5 Guess Then Determine Attacks on GOST

In this section we explain how to compute output bits for a certain round of GOST with incomplete knowledge of all the key bits on which this bit depends in this and previous rounds. We have the following basic fact:

Fact 5.0.2 *The input on 4 bits of any particular S-box in GOST 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. E.g.*

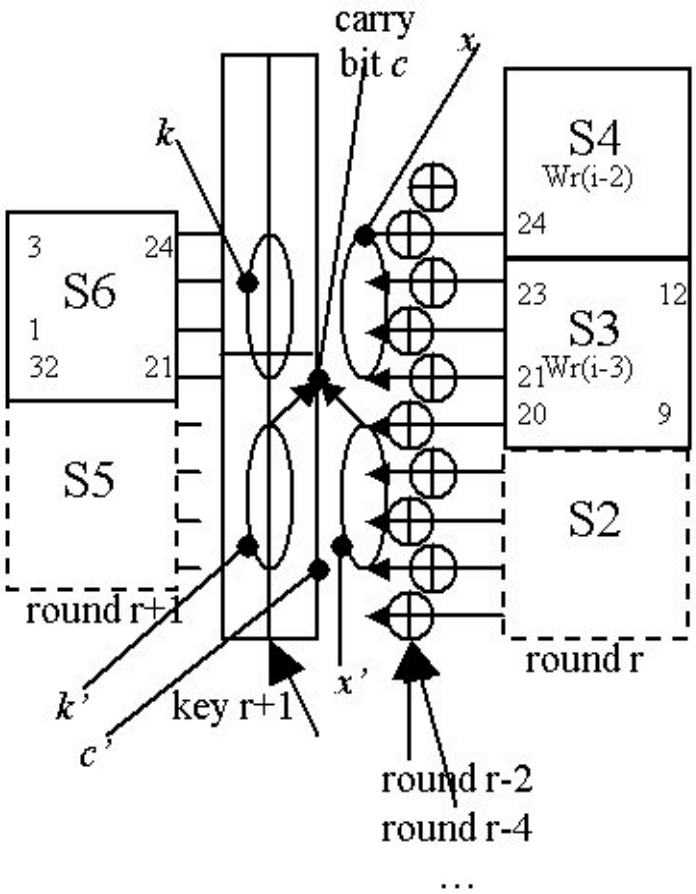

Figure 7: Computation of the Input Of One S-box With A Carry Bit

In our attack we exploit the weakness of carry propagation in the addition modulo 2^{32}. It is possible to see that carry bits such as c can be guessed with a surprisingly high accuracy. We observe that:

1. We define $Wr(i)$ by the equation $Wr(i) - 1 = (i - 1) \bmod 8$. This corresponds to the number of S-box within 1..8 with wrap-around.

2. The input of each S-box Si in round $r + 1$ is

$$a = x + k + c \bmod 16$$

and depends on (i) the 4 key bits k at the entry of this Si and (ii) x obtained from the outputs of two S-boxes in round r with numbers $Wr(i-2)$ and $Wr(i-3)$ XORred with the appropriate bits after round $r - 2$ (this part does not change in round $r - 1$), and (iii) one carry bit c.

3. The carry bit c is such that

$$c = 1 \Leftrightarrow x' + k' + c' \geq 16$$

where x' and k' are the data and the key at the previous S-box, and c' is the previous carry bit.

4. The previous carry bit influences the result with low probability which will be quantified below.

From here we easily obtain that:

Fact 5.0.3 *Let $i > 1$ (with S1 there is no carry entering as on Fig 7.*

We assume that the attacker knows the whole 64-bit output of round $r-2$, the input of one S-box Si at round $r + 1$, and the key k at the same Si in round $r + 1$, and the state of $Wr(i - 2)$ and $Wr(i - 3)$ in round r.

Let k' be the unknown key at S-box i-1 in round $r + 1$ (cf. Fig 7).

Then we have the following results:

1. *Let d be the bit he obtained from 1 lower bit from $Wr(i - 3)$ XORed with the appropriate state from round $r - 2$, which is bit 20 on our example at Fig. 7,*

   Assume that the attacker knows one most significant bit of k' which we call e.

   If $d = e = 1$, we have $c = 1$ with probability 1.

   If $d = e = 0$, we have $c = 0$ with probability 1.

   If $d + e = 1$, we have $c = 0$ or $c = 1$ which are more or less equally likely.

2. *If the attacker knows the whole 4-bit k' he can compute c correctly with probability of about $3/4 = 1 - 2^{-2}$.*

3. *If the attacker knows k' AND the state of $Wr(i-4)$ in round r, he can compute c correctly with probability of roughly about $1-2^{-4}$.*

Justification: The first result is straightforward and we will derive the second result. Each of these probabilities can be established by a discrete simulation checking all the possible cases.

The top bit b of x' is known due to $Wr(i-3)$, therefore the expected value of x' is about $x' \approx 8*b+4$, and the whole k' is known. It is easy to see that the expected approximation error (computed as average over 8x8 cases) is $|x'-8*b+4| = 1.31$.

We decide that $c=1 \Leftrightarrow 8*b+4+k'+c' \geq 16$. This will be accurate unless $x'+k'+c' < 16$ AND $8*b+4+k'+c' \geq 16$ or the vice versa with the difference between these two numbers being on average 1.31. Each of these 2 cases occurs with probability of very roughly about $1.3/16 \approx 2^{-4}$. Overall we expect that with probability $1-2^{-3}$ our computation of c is correct.

In the similar way we derived the other results.

Important Remark: The intention of this theorem is **not** that is some cases the computations done in our attack will be incorrect and therefore we might miss some cases and the attack would fail. We can handle it in a very different way. Each time we will determine if $c=1$, by checking $x'+k'+c' \geq 16$ with more or less exact approximations of x' and k' and c', we know exactly the margin of error and know exactly when there will be two possibilities for c. In all these cases we are simply going to include in our enumeration two cases, one with $c=0$ and one with $c=1$, which is going to change the 4 outputs of the affected S-box.

6 An Improved Differential Attack on GOST

For the ease of reading we split our attack in 5 stages. All the stages should be seen as a part of the same Depth First Search procedure, where we guess key bits, reject some cases, then guess more key bits, reject again, etc, and where as we advance in the attack tree the time complexity decreases, however the probability to arrive at this level for a particular set of choices also decrease with many early aborts.

6.1 Attack Stage 1

We proceed as follows:

1. We are given 2^{64} KP which are assumed to be stored in a database.
2. We have the Alpha property which holds $2^{13} + 2^{11.9}$ times for the full 32=6+20+6 rounds.
3. First we are going to reduce the total number of pairs from 2^{127} to a lower number, by a birthday-like approach which avoids the enumeration of all possible pairs.
4. Given an assumption on a certain number of key bits, we define as **inactive bit** a bit where P_i and P_j collide (the difference is 0) at a certain bit location inside the cipher, **if** our assumption about the key is correct.
5. Our attack will have many steps in which we are going progressively guess some key bits, then reduce the space of pairs considered due to our differentials, which will make it feasible to guess additional key bits at a later stage.
6. We want to write constraints which describe the following events which occur in the first 5 (and last 5) rounds in our property Alpha.
   - The output after the addition of the output of S7 and S1 after round 2 gives 8 inactive bits at 0 which are 3-6,11-14. This is implied by the set $0xFFFF8787$ which occurs after round 2.
   - The output after the addition of the output of S4,S5,S6,S7 after round 3 gives 15 (and not 16) inactive bits at 0 which are 24-31,1-7, but not 32. This is implied by the set $0x807FFF80$ which occurs after round 3.

7. We consider and try to guess the following key bits: all key bits at rounds 1,2,3 and 20 key bits for S-boxes S12345 in round 4.

8. We observe that for any guess of these $96 + 20 = 116$ key bits, we get 8+15+13=36 cancelations after rounds 1-4 as explained above, and 36 more cancelations on exactly the same S-boxes with the same keys after round 29 going backwards.

   This can be seen as a collision on 36+36=72 bits, computed as a function of type $f_k(P_i)$ where k represents 116 bits of the key and i is one of the 2^{64} KP cases.

9. For each 116 possible guesses for our selected key bits we compute 2^{64} possible strings on 72 bits for each P_i. Only a proportion of one out of 2^8 values on 72 bits are taken. For any given case i the probability that there is another j for which the 72 bits collide is 2^{-8}.

10. Then we can enumerate in time of maybe $4 \cdot 2^{64}$ CPU clocks some $2^{64-8} = 2^{56}$ possible i or j with $2^{56}/2 = 2^{55}$ distinct pairs i, j which collide on these 72 bits.

    Another way of looking at this is as follows: there are (we do NOT ever enumerate all of them) about 2^{127} pairs P_i, P_j and there are about 2^{127} differences $f_k(P_i) - f_k(P_j)$ on 72 bits. Some $2^{127-72} = 2^{55}$ of these differences will have all the 72 bits at 0.

11. These 2^{55} pairs per key assumption can be enumerated efficiently. A simplified method is as follows: We make a hash table where at address which a hash of $f_k(P_i)$ on 72 bits, and we store i as well. Each time the value is already taken we output a collision. We will output a list of 2^{55} pairs P_i, P_j.

    Memory required is roughly about 2^{70} bytes.

12. The total time spent in these steps of the attack should not exceed $2^{116 \cdot 64}$ times the cost of computing roughly speaking 1 round of GOST.

    It is not need to do as much work as computing 2^{116} times 4 first rounds of GOST and 4 last rounds of GOST. Basically the cost of computing the first three and the last three rounds of GOST can be neglected. More precisely it will be amortized in 2^{20} sub-cases of the 2^{96} cases, in which we just need to evaluate 4 S-boxes in round 3 and 4 S-boxes in round 30, which is roughly feasible to do in most an equivalent of 1 round of GOST.

Therefore we estimate that we need only about $2^{116+64} \cdot 8$ CPU clocks, which could be seen as an equivalent of roughly about 2^{174} GOST encryptions.

To summarize, we can thus in total overall time equivalent to about 2^{174} GOST encryptions and with memory of about 2^{70} bytes, enumerate $2^{171} = 2^{116+55}$ cases of type k_{116}, i, j. We get on average 2^{55} possible pairs i, j for each key assumption on 116 bits.

On Fig. 8 we summarize all the current and further steps of our attack.

guess key at S-boxes	correct	difference	new bits to cancel after outputs of	after round	new inactive bits (60 in total)		enumerate cases	per key	key bits assum.	time GOST encrypt.
								2^{127}		
all bits in R12	2^{-64}	FFFF8787	S7,S1	2,31	8	4-7, 12-15	birthday			
S3*4567R3	2^{-20}	807FFF80	S456^{3}7	3,30	15	24-31,1-7	attack			
S812R3 S12345R4	2^{-32}	F0000787	S2345^1	4,29	13	16-28	2^{55+116}	2^{55}	116	2^{174}
							2^{171}			
S6R4 S78R5	2^{-12}	80780000	S8	5,28	4	8-11	$2^{171+12-4-4}$	2^{47}	128	2^{179}
S7R4 S1R5	2^{-8}	80780000	S1	5,28	4	12-15	$2^{175+8-4-4}$	2^{39}	136	2^{174}
S8R4 S2R5	2^{-8}	80780000	S2	5,28	4	16-19	$2^{175+8-4-4}$	2^{31}	144	2^{174}
							2^{175}			
S3R5 S4^{1}5R6	2^{-9}	00000700	S5^3	6,27	3	29,30,31	$2^{175+9+1.2-3-3}$	$2^{26.2}$	153	2^{176}
S4R5 S6R6	2^{-8}	00000700	S6	6,27	4	32,1-3	$2^{179.2+8-4-4}$	$2^{18.2}$	161	$2^{178.2}$
$2^{18.2}+2^{11.5}$	is	chosen	at	$2^{2.4}$	standard		deviations	$*2^{-24}$	to survive	
except for the	right	161 bits	we	have	to remain		$2^{179.2-24}$	or 2^{-6}	per key only	
S56R5 S781R6 S23R7	2^{-28}	00000700 80000000	S8^1 S3	6,27 7,26	1+ 4	8 20-23	$2^{155.2+28-5-5}$	$2^{8.2*}$	189	2^{175}
$2^{8.2}+2^{10}$	is	chosen	at	$2^{5.9}$	standard		deviations	-	certitude	
									total	$2^{178.6}$

Figure 8: Summary of Major Steps In Our Attack on GOST

6.2 Attack Stage 2

Now we are going to work on additional key assumptions with the objective to decrease the number of pairs per key from 2^{55} to a much lower number so that we can apply the distinguisher given by Fact 4.2.1, which will be done at the Stage 4.

1. Now we look at the difference $0x80780000$ obtained after round 5, where outputs of S-boxes S8, S1 and S2 are 'newly' inactive which is a cancelation on 12 bits in round 5.

   First we will work on S8, then on S1, then on S2.

2. First we guess some additional 4+4 key bits. The situation is the same as on Fig. 7 with boxes S78 at round 5 depending mostly on boxes S456 at round 4.

   We guess 4 bits at S-box S6 in round 4, needed only to compute the bit 31 entering S8 at round 5, and the 4 key bits at S8 in round 5, and an approximation on the 4-key bits at S7 in round 5, which together with outputs of S4 and 1 bit from S5 in round 4, can be used to compute the carry entering S-box S8 at round 5 with probability of about $1 - 2^{-4}$ (cf. Fact 5.0.3).

3. More over and quite importantly we do **not** allow any errors in our computations. In rare cases where there is an ambiguity about the carry, because for example we have 15 and the carry added from S6 in round 5 could matter, we simply check both cases. This leads to a negligible increase in the total number of cases checked from about 2^{171+12} to about $(1 + 2^{-4})2^{171+12}$, see Fact 5.0.3.3. For simplicity we ignore these additional numbers which are negligible compared to other numbers in this attack.

   Later during the attack, when the key at $S6$ and early S-boxes becomes known, these additional cases will be eliminated instantly. In fact we can also leave these additional cases, everything we do later in our attack can tolerate a small proportion of additional incorrect cases.

4. With these 12 new key bits, we can enumerate 2^{171+12} cases k_{116+12}, i, j. In each cases with probability 2^{-4} the 4 bits XORed to the output of S-box S8 become inactive at round 4, and with probability 2^{-4} they also become inactive at round 29.

5. Accordingly in time of about 2^{177} computations of 2/32 full GOST, which is about 2^{179} GOST computations, (assuming one takes 2^9 CPU

clocks). We reject most cases except $2^{171+12-4-4} = 2^{175}$ cases k_{128}, i, j.

6. This, is 2^{47} cases per key.

7. Now we guess 8 more key bits. These are 4 bits at S-box S7 in round 4 which output 3 is needed to compute the input of S1 in round 5 (there is no carry entering S1). We also guess 4 key bits at S1 in round 5.

8. Now we have an enumeration of 2^{175+8} cases k_{136}, i, j, where we now have 136 key bits. In this list with probability 2^{-4} the 4 bits XORed to the output of S-box S1 become inactive at round 4, and with probability 2^{-4} they also become inactive at round 29.

9. Accordingly in time of about 2^{175+8} computations of 2/32 full GOST, which is about 2^{174} GOST computations, we have an enumeration $2^{175+8-4-4} = 2^{175}$ cases k_{136}, i, j.

10. Now we guess 8 more key bits. These are 4 bits at S-box S8 in round 4 which outputs are 8-11 and which are needed to compute the input of S-box S2 in round 5 (the carry entering S2 is already known for S1 in round 5 above). We also guess 4 key bits at S2 in round 5.

11. Thus we consider the enumeration of 2^{175+8} cases k_{144}, i, j, where we now have 144 key bits. In this list with probability 2^{-4} the 4 bits XORed to the output of S-box S2 become inactive at round 4, and with probability 2^{-4} they also become inactive at round 29.

Accordingly in time of about 2^{175+8} computations of 2/32 full GOST, which is about 2^{174} GOST computations, we enumerate about $2^{175+8-4-4} = 2^{175}$ cases k_{144}, i, j. We are left with 2^{31} pairs i, j on average for each key assumption on 144 bits which will be the cases which we will check in later steps of our attack.

For the right key assumption we will also obtain the $2^{11.9}$ cases which have the property Alpha for the correct GOST key

6.3 Attack Stage 3

We will continue the process of guessing additional key bits and decreasing the number of cases per key assumption.

1. At this stage, in each case, we know all key bits in rounds 1,4 and key bits S-boxes S1278 in round 5, for a total of 144 key bits.

2. Now in round 6 we have the difference 0xF0000787 which becomes 0x00000700. The S-box outputs which are going to become inactive are: 3 outputs of S5 with numbers 29,30,31, the whole of S6 with numbers 32,1-3, and one lower bit of S8 with number 8.

3. We will first work on S5, then on S6, and later on S8.

4. First we guess 9 key bits: for S3 at round 5, and for S5 at round 6 and just one most significant bit for S4 at round 6. We have 3 inactive bits 29-31. Following Fact 5.0.3. this allows to determine exactly the carry bit c with probability $1/2$, and the attacker knows in which case it is (when $d = e$, cf. Fact 5.0.3.1.), and otherwise we have two cases to include (when $d \neq e$, cf. Fact 5.0.3.1.).

   Overall on average we have $(1 + 2)/2 \approx 2^{0.6}$ more cases to check and we compute the output of S5 at round 6 about $2^{175+9+0.6} = 2^{177}$ times.

5. In addition we also need to compute the output of S5 at round 27 in each of these cases. In the same way sometimes this generates 1 or 2 cases to check, and overall we get another factor of $2^{0.6}$.

6. Accordingly in time of about $2^{175+9+0.6+0.6}$ computations of 2/32 full GOST, which is about $2^{176.2}$ GOST computations, we obtain a list of $2^{175+9+1.2-3-3} = 2^{179.2}$ cases k_{153}, i, j. This is $2^{26.2}$ cases per key.

7. Then we guess 8 more key bits: for S4 at round 5, and for S6 at round 6. We have 4 inactive bits 32,1-3.

8. Accordingly in time of about $2^{179.2+8}$ computations of 2/32 full GOST, which is about $2^{178.2}$ GOST computations, we obtain a list of $2^{179.2+8-4-4} = 2^{179.2}$ cases k_{161}, i, j.

9. This is only $2^{18.2}$ cases per key on 161 bits which is within reach of our distinguisher attacks.

10. The total time spent in all the above steps is is about $2^{178.5}$ GOST computations, and probably only half of this number on average is needed.

6.4 Attack Stage 4

Now we are going to be able to see if 161 key bits are right or wrong.

We recall Fact 4.2.1. For the full 32-round GOST and on average over the GOST keys, there exists two disjoint sets with $2^{13} + 2^{11.9}$ distinct pairs of plaintexts $P_i \neq P_j$ which have the Alpha property.

We have $2^{18.2}$ cases per key, which for the right key on 161 bits contains these correct $2^{13} + 2^{11.9}$ cases. All these cases come from the fact that we have independently in the first 6 and the last 6 rounds, checked if certain set of twice 55 differences are at 0, which gives 2^{17} pairs surviving. We have also produced an overhead of some $2^{1.2}$ additional cases which result from incertitude due to further unknown key bits which gives $2^{18.2}$ pairs total.

As before, It is clear that these $2^{18.2}$ pairs obtained in the specific case of the right 161-bit key, occur at random due to the random intersection between cases which may occur at the beginning of GOST, and at the end of GOST, without correlation between these events.

It is easy to see that $2^{18.2}$ such pairs on average, with an expected standard deviation of about $2^{9.1}$, are still going to occur if we explicitly exclude about $2^{64+14-1} \ll 2^{127}$ cases where a difference of type $(0x80700700, 0x80700700)$ occurs after 6+7 rounds AND at 6+7 rounds from the end, which as explained for Fact 4.0.1 occurs with very low probability of about $2^{-32.4}$ and in fact less, because in our case it is not yet certain that the difference is as expected after round 7.

However because the $2^{11.9}$ cases do ALL have differences of type $(0x80700700, 0x80700700)$ after 6+7 rounds AND at 6+7 rounds from the end, the two sets are disjoint. To summarize we obtain the following result:

Fact 6.4.1 *After Stage 3 of our attack, if the 161 bits are wrong, most of the time (this will be quantified below) we get about $2^{18.2}$ cases per key.*

We assume that the attacker will decide that the key on 161 bits is correct if he sees at least $2^{18.2} + 2^{11.5}$ cases for this key. Otherwise he will reject it.

The correct 161-bits key will be accepted with probability of 95%.

Incorrect 161 bits will be accepted with probability of about 2^{-39}.

Justification: A correct 161 bits should give about $2^{18.2} + 2^{11.9}$ cases with standard deviation of $2^{9.1}$ and will be rejected only if we are below $2^{18.2}+2^{11.5}$ cases which is on one side of and outside of $(2^{11.9-11.5})/2^{9.1} = 2$ standard deviations. By applying the Gauss error function [20] we see that a correct key will be accepted with probability of about 95%.

If the 161 bits are wrong, we are outside of and on one side of, $2^{11.5-9.1} = 2^{2.4}$ standard deviations. Here the Gauss error function [20] gives a probability only about 2^{-24}.

6.5 Attack Stage 5

We need to do some additional guessing and filtering.

Up till now, with total time of about $2^{178.5}$ GOST computations, we are able to enumerate $2^{179.2-24} = 2^{155.2}$ cases k_{161}, i, j.

Our 161 bits of the key are all the bits for the first 4 rounds, and 24 bits at round 5 for S781234, and 9 bits at round 6 for S5,S6 and one bit at S4.

1. We guess the remaining 8 bits to complete round 5 with boxes S56. Then we guess the key at boxes S7181 at round 6 and at S213 in round 7. This is a total of 28 bits. For simplicity we guess all these bits (a more refined approach is NOT needed because the total time spent in this step is small).

2. The output after S8 in round 6 needs to cancel on 1 bit which is number 8, and the output of S3 in round 7 needs to cancel on 4 bits which are 20-23.

   This is implied by the sets $0x00000700$ and $0x80000000$ in the Alpha property obtained after round 6 and 7.

3. Accordingly in time of about $2^{155.2+28}$ computations of 2/32 full GOST, which is about 2^{175} GOST computations, we reject most cases except $2^{155.2+28-5-5} = 2^{173.2}$ cases k_{189}, i, j.

   This seems to be about 2^{-16} per key on average, which comes from the fact that only some 161-bit sub-keys are present in the keys on 189 bits. However if we look only at 2^{165} keys on 189 bits which are actually present, we have $2^{8.2}$ cases per key.

4. We assume that the attacker will reject all cases where the count is less than $2^{8.2} + 2^{10}$.

5. Then it is easy to see that if the key is correct, it will be accepted with probability very close to 1.

6. If the key is wrong, we observe that that $2^{8.2} + 2^{10}$ is outside $2^{5.9}$ standard deviations. Here the Gauss error function [20] gives a figure much smaller than 2^{-256}.

Summary: Thus given 2^{64} KP and in an average time of about $2^{177.6}$ GOST computations, we are able to determine with certitude 189 bits of GOST key. The remaining 66 bits can then be found by brute force.

The attack was designed to work for 95% of GOST keys.

7 Conclusion

Bruce Schneier has written on page 334 of his famous book (second edition 1996, cf. [19]) that "Against differential and linear cryptanalysis, GOST is probably stronger than DES". In 2000 Russian researchers claimed that as few as 7 rounds out of 32 are sufficient to protect GOST against differential cryptanalysis, see [11, 10]. In the same year Japanese researchers [18] show that more powerful differential attacks exist, exploiting sets of differentials [18], which allows to break about 13 rounds of GOST out of 32.

GOST has a substantially lower implementation cost than any other comparable cipher at a similar security level, see [17] and until 2011, no cryptographically significant attack on GOST used in encryption was found [17]. Consequently, in 2010 GOST was submitted to ISO to become an international encryption standard, see [3]. 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. [17].

Many new attacks on GOST have been proposed since 2011 [3, 8, 14, 6, 4, 5, 9] including new better attacks which exploit multiple differentials. In 2011 Courtois and Misztal have found new differential sets which allow for much faster differential attacks on GOST [6] than expected [18]. Then if one exploits the key scheduling of GOST, one can break full GOST within time of $2^{254.6}$ GOST computations which is very slightly faster than brute force [4] and which attack can be further refined by a more careful guessing procedure in [5] to achieve about 2^{224}.

In this paper we present a further improved and refined advanced differential attack on full 32-round GOST. Given 2^{64} KP we can recover the full 256-bit key for GOST within only about $\mathbf{2^{178}}$ **GOST computations** on average for a success probability of 95 %. The memory required is about 2^{70} bytes.

This is the fastest single-key attack on GOST found so far. We can compare it to the most recent result by Shamir *et al.* with time complexity of 2^{192} which is going to be presented at FSE 2012 in Washington DC, on 19 March 2012. Our attack is several thousands of times faster.

References

[1] Eli Biham, Adi Shamir, *Differential Cryptanalysis of DES-like Cryptosystems,* Journal of Cryptology, vol. 4, pp. 3-72, IACR, 1991.

[2] Nicolas Courtois: *The Best Differential Characteristics and Subtleties of the Biham-Shamir Attacks on DES,* On `eprint.iacr.org/2005/202`.

[3] 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/`.

[4] Nicolas Courtois, Michał Misztal: *First Differential Attack On Full 32-Round GOST,* in ICICS'11, pp. 216-227, Springer LNCS 7043, 2011.

[5] Nicolas Courtois, Michał Misztal: *Differential Cryptanalysis of GOST,* In Cryptology ePrint Archive, Report 2011/312. 14 June 2011, `http://eprint.iacr.org/2011/312`.

[6] Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* In 11th Central European Conference on Cryptology, post-proceedings in preparation.

[7] Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007. Preprint available at `eprint.iacr.org/2006/402/`. Also presented at ECRYPT workshop Tools for Cryptanalysis, Krakow, 24-25 September 2007.

[8] Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* Preprint, 12 November 2011, available at `http://eprint.iacr.org/2011/626`

[9] Itai Dinur, Orr Dunkelman and Adi Shamir: *Improved Attacks on Full GOST, 11 October 2011,* At `http://eprint.iacr.org/2011/558/`. A version of this paper will be presented at FSE 2012 in Washington DC, on 19 March 2012.

[10] 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

[11] 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.

[12] 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. An English translation can be found at `http://www.autochthonous.org/crypto/gosthash.tar.gz`

[13] 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/`

[14] Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011.

[15] Orhun Kara: *Reflection Cryptanalysis of Some Ciphers,* In Indocrypt 2008, LNCS 5365, pp. 294-307, 2008.

[16] Gregor Leander, Axel Poschmann: *On the Classification of 4 Bit S-Boxes, In Proceedings of WAIFI'07, 1st international workshop on Arithmetic of Finite Fields.*

[17] Axel Poschmann, San Ling, and Huaxiong Wang: 256 Bit Standardized Crypto for 650 GE GOST Revisited, In CHES 2010, LNCS 6225, pp. 219-233, 2010.

[18] Haruki Seki and Toshinobu Kaneko: *Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, Selected Areas in Cryptography, Douglas R. Stinson and Stafford E. Tavares, editors, LNCS 2012, pp. 315323, Springer, 2000.*

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

[20] Standard Deviation – wikipedia article, 13 June 2011, available at `http://en.wikipedia.org/wiki/Standard_deviation`.