An Improved Differential Attack on Full GOST

Nicolas T. Courtois

University College London, Gower Street, London, UK

Abstract. GOST 28147-89 is a well-known block cipher. Its large key size of 256 bits and incredibly low implementation cost make it a plausible alternative for AES-256 and triple DES. Until 2010 “despite considerable cryptanalytic efforts spent in the past 20 years”, GOST was not broken see [30]. Accordingly, in 2010 GOST was submitted to ISO 18033 to become a worldwide industrial encryption standard.
In paper we focus on the question of how far one can go in a dedicated Depth-First-Search approach with several stages of progressive guessing and filtering with successive distinguishers. We want to design and optimized guess-then-truncated differential attack on full 32-bit GOST and make as as efficient as we can. The main result of this paper is a single-key attack against full 32-round 256-bit GOST with time complexity of 2^{179} which is substantially faster than any other known single key attack on GOST.

Key Words: Block ciphers, GOST, differential cryptanalysis, truncated differentials, guess-then-determine, Gaussian distribution, distinguisher attacks.

1 Introduction

GOST 28147-89 is a well-known block cipher and a government standard of the Russian Federation. A 256-bit block cipher which can claim to be a serious alternative for AES-256 and triple DES, given its very low implementation cost [30]. Until 2010 there was no attack on GOST, cf. [30].

Then in 2011 it was discovered that GOST can be broken and is insecure on more than one account. There is a substantial variety of recent attacks on GOST [3–6, 19, 11, 12, 23, 16, 17, 24]. In particular there is a large variety of self-similarity and black-box reduction attacks [3, 16, 9, 11, 17, 19, 24]. There have also been quite a few papers about advanced differential attacks on GOST [33, 6, 4, 5, 13, 32, 29, 31, 8, 15]. In contrast to other recent works on this topic we do not focus on the complex question of how such attacks can be discovered, cf. [7, 14, 29], or how reliable some heuristic results are [33, 6, 4, 5, 8, 15] especially given the fact that GOST is not a Markov cipher [25, 14, 29] or how to optimize them in general for one given set of S-boxes cf. [7, 13, 29] or for major alternative sets of S-boxes cf. [13, 32, 31, 8, 29]. We don’t look at multiple key attacks [9, 11, 16, 17, 24] or at more advanced “combination” attacks which combine the complexity reduction approach based on high-level self-similarity of [16, 11, 3] with advanced differential properties with 2,3 and 4 points, [16, 17].

This paper is about developing a complex advanced differential attack on GOST block cipher which involves several steps with progressive guessing of well-chosen key bits, and several statistical distinguisher steps.

1.1 GOST and Differential Cryptanalysis

Differential cryptanalysis (DC) is based on tracking of changes in the differences between two messages as they pass through the consecutive rounds of encryption. It is one of the oldest classical attacks on modern block ciphers [18, 2], we refer to cf. [8] for a short historical survey. Differential attacks are very well known. Yet researchers have until recently failed to accurately evaluate the strength of GOST against DC. GOST was quite frequently claimed very secure against such attacks. In late 1990s Schneier writes that: "against differential and linear cryptanalysis, GOST is probably stronger than DES", cf. [34]. Later in 2000 Russian researchers claimed that breaking GOST with five or more rounds is "very hard" and claim that as few as 7 rounds out of 32 are sufficient to protect GOST against DC [20]. Needless to say later research have not confirmed at all such very optimistic claims [25, 26, 14, 15]. GOST appears to be quite secure in the very basic historical Biham-Shamir formulation of DC with single differences on the full state [2, 20, 33]. A more powerful family of attacks are "truncated differential" attacks by Knudsen [27] which have been applied to GOST as early as in 2000 by Seki and Kaneko [33] with some success. In 2011 Courtois and Misztal have found new differential sets for GOST [6] which are substantially better than previously known. The possibilities offered by such attacks remain poorly understood in the research community. For example in the a recent survey paper specifically about advanced differential cryptanalysis and specifically on block ciphers with small blocks in Section 1.1. page 3 of [1] we read: *Truncated differentials, [...] in some cases allow to push differential attacks one or two rounds further.* This paper and our other recent research on GOST [6, 4, 5, 8, 13–15] shows that we can gain not two but much closer to 20 rounds (!) compared to what we would expected to achieve with single differentials [20, 21, 33].

In this paper we work on essentially one single highly optimized attack, the best we could find. It is a balancing act between the time complexity of different consecutive steps of a complex attack. An intermediate goal is to construct distinguishers on say 20 rounds of GOST: this question has been studied in [7, 8, 13–15, 29]. This paper is about how to transform one such distinguisher into an efficient attack on the full 32-round GOST cipher.

This paper is organized as follows: In Section we explain the high-level structure of GOST and explain the role played by GOST key scheduling. We explain the principle of splitting GOST into three sections of for example 6+20+6 rounds. Then in Section 3 we study the question of constructing a distinguisher for 20 rounds of GOST which is a central question in this paper. This is further extended into a sequence on of "concentric" distinguishers for decreasing supersets of 20 rounds cf. Section 4. In Section 5 we study the propagation inside GOST in order to be able to construct well chosen subsets of key bits to be guessed at various stages of our attack. Thus finally in Section 6 we describe a full advanced differential attack on 32 rounds of GOST which works in 5 stages in which well-chosen assumptions on the key and data (plaintext) bits are progressively refined with early rejection.

2 GOST and Key Schedule

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 circular rotation by 11 positions. Each round of GOST looks exactly the same except for the key k used:

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

GOST has 32 rounds such as the one described in Fig. 1 below. The $\boxplus$ denotes the addition modulo 2^{32}. On our picture below the $\boxplus$ denotes the addition modulo 2^{32}. 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 $32, 1, 2, 3$. At the left margin in Fig. 1 we also show S-box numbers in the next round, to see which bits are successfully determined in our attacks on GOST, cf. later Fig. 7 page 11.

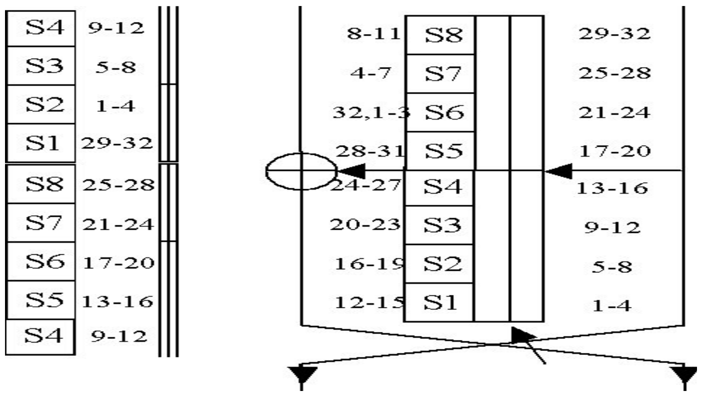

Fig. 1. One Round of GOST And Connections in The Following Round

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

Fig. 2. Key schedule in GOST

This property has a big impact on security of GOST: 32 bits of the whole key, a fairly small proportion, are 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 cf. [4]. In this paper we 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.

Given one round in Fig. 1 and key scheduling in Fig. 2 we have a complete description of GOST. In this paper we will only study the most popular set of GOST S-boxes a.k.a. the "GostR3411_94_TestParamSet" which was published by Schneier in 1994 and which claimed to be used by the Central Bank of the Russian Federation [34]. 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, we refer to [7, 13, 29, 13, 32, 31, 8, 29] for the study of similar attacks for other S-boxes.

2.1 Preliminary Remarks

In our later attack which will be described in Sections 4-6 we are going to split GOST into three pieces with 6+20+6 rounds. Early advanced differential attacks were based on a for up to 20 rounds of GOST [4, 5] and mandated guessing complete 32-bit keys for several outer rounds in order to fully reconstruct these internal differentials. This approach is refined in [13, 14, 29, 7] and in this paper. 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. 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. We are just dealing with assumptions on internal difference bits in GOST, their consequences and relations between these assumptions but for many many steps 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 some other permutation. This can only be seen at a much later stage of the attack.

Before we get there we need to study a number of preliminary technical questions.

2.2 Sets of Differentials, Aggregated and Truncated Differentials

Truncated differential attacks [27] have been studied since 1994. Some attacks on GOST are proposed in 2000 by Seki and Kaneko [33] and since 2011 better and stronger differential properties have been found, cf. [6, 4]. We consider differences with respect to the popular bitwise XOR operation. Following previous work on this topic [4, 5] 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 contained within a certain mask. This also is a special case of "Truncated Differentials" [27] which are defined as fixing the difference not on all but a subset of data bits. However we need to be careful and explicitly exclude all-zero differentials from this set. For example for

$$\varDelta = 0x80700700$$

we obtain a set of all differences on 32 bits 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 $(\varDelta, \varDelta)$ is a set of difference on 64 bits with up to 14 active bits, where any non-zero difference is allowed, including also differences where the difference is zero in one half, but not the all-zero difference on both halves. We have $|A| = 2^{14} - 1$: there are exactly $2^{14} - 1$ single 64-bit differences in this set of differentials A.

For example the following fact was established in [6, 4, 5]:

Fact 2.3 *The aggregated differential* $(\varDelta, \varDelta)$ *with uniform sampling of all differences it allows, produces an element of the same aggregated differential set* $(\varDelta, \varDelta)$ *after 4 rounds of GOST with probability about* $2^{-13.6}$ *on average over all possible keys, where* $\varDelta = 0x80700700$ *has 7 active bits.*

For 6 rounds the probability is $2^{-18.7}$ *on average over all possible keys.*
For 8 rounds the probability is $2^{-25.0}$ *on average over all possible keys.*

Remark: Recent research shows that the size of 14 bits is close to optimum, i.e. set with a different size are less likely to be as good, see [15]

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

Fact 2.4 *The set* $(\varDelta, \varDelta) = (0x80700700, 0x80700700)$ *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. We have $|A| = 2^{14} - 1$ and $|B| = 2^8 - 1$. This *an aggregated differential* A, B contains $(2^{14} - 1)(2^8 - 1)$ single differential characteristics.

Truncated differential attacks on GOST are facilitated by a strong internal structure inside GOST where GOST splits very neatly into two loosely connected parts, cf. Fig. 3 and Section 4 of [7] and different interesting truncated differential attacks can be classified in relation to this structure [7, 13, 14].

For example some of the best known attacks cf. Fact 2.3 which are also exploited in this paper are said to be of type 3+3 S-boxes in each of the loosely connected parts. These 3 S-boxes and their connections are shown in Fig. 3 below.

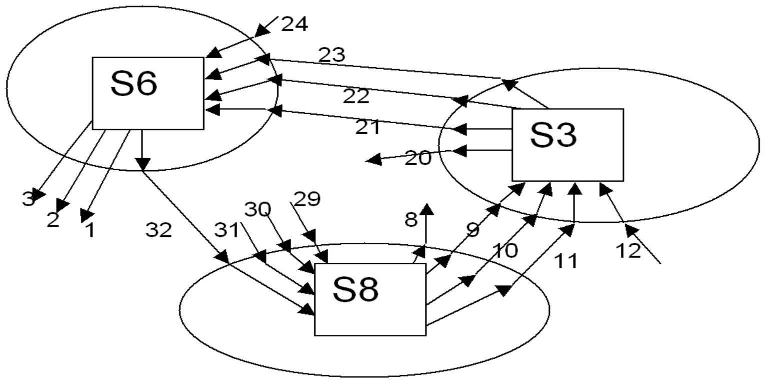

Fig. 3. Connections between S-boxes exploited in Fact 2.3.

3 Our Main Distinguisher For 20 Rounds

Our goal is to design an attack on the full 32-round GOST. As in [4, 5] we guess some key bits and use a distinguisher, however our new distinguisher is symmetric and the attack will have more stages. 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. Not only for a Random Permutation (RP) but also almost always, with overwhelming probability $1 - \varepsilon$ for ANY permutation such as several rounds of GOST block cipher. We 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.

Fact 3.1 *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 20 rounds of GOST.*

For a typical permutation on 64-bits 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* $2^{7.8}$*.*
We are at $2^{6.1}\times$ *the standard deviation.*

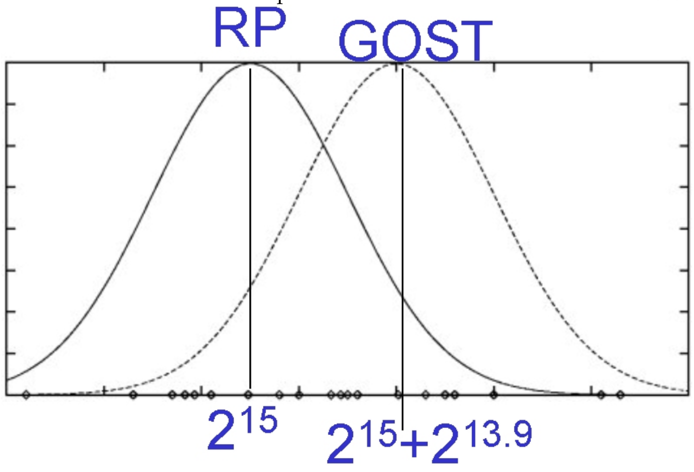

Fig. 4. Signal vs. Noise Differential Distinguisher for 20 Rounds of GOST

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 of type $(0x80780000, 0x00000700) \rightarrow$ (after some permutation OR 20 inner rounds of GOST) $\rightarrow (0x00000700, 0x80780000)$. 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 the actual 20 rounds of GOST 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
```

Fig. 5. "Propagation" For 20 rounds With Specific Middle Differentials

Following Fact 2.3 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 2.4 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 in 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 to 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 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.

4 Concentric Distinguishers

We have constructed one very good distinguisher for 20 rounds of GOST. Now the question is as follows. In the similar way as in [5] we want to avoid the necessity to examine all possibilities for the key in the first 6 rounds, and just apply the distinguisher. We want to progressively reduce the key size and the data space on the way, and for this to build a sequence of **concentric distinguishers** for 22, 24 and more rounds which allow early rejection of many cases, so that we are going to examine 20 rounds of GOST with some assumptions on the key and some subset of data much less frequently. This is expected to lead to really efficient attacks on full 32-round of GOST. One very simple example of such attack was already described in [5]. In this paper we are going to study much more complex distinguishers.

4.1 Extending with Additional Weakly Constrained Rounds

We start with our distinguisher property of Fact 3.1. This property is going to be extended with a "weakly constrained" differential propagation which occurs with quite a high probability for 6 more rounds on each side.

We also need a model to account for what is going to happen when our assumptions are wrong. Therefore we are going to compare what happens with GOST split as 6+20+6 rounds to a situation which involves a random permutation (RP) as follows. We 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 is illustrated in Fig. 6 which accounts for both sort of situations. It can represent the full 32-round GOST with 20 rounds in the middle, and it could also be a situation which we wrongly assumed to be the full

32-round GOST and the middle permutation is not exactly a random permutation however it is not at all what we assumed in our attack and we can expect that it might behave as a random permutation for the properties we study.

This leads to the following property which is the core property in our later attack on full 32-round GOST.

Definition 4.2 (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:*

```
     <plaintext>             ------------------->
0xFFFFFFFF 0xFFFFFFFF         |         0x00000700 0x80780000
      (1 Round)               |               (1 Round)
0xFFFFFFFF 0xFFFFFFFF         |         0x80780000 0xF0000787
      (1 Round)               |               (1 Round)
0xFFFFFFFF 0xFFFF8787    (20 Rounds)    0xF0000787 0x807FFF80
      (1 Round)            (or RP)            (1 Round)
0xFFFF8787 0x807FFF80     (or other)    0x807FFF80 0xFFFF8787
      (1 Round)               |               (1 Round)
0x807FFF80 0xF0000787        / \        0xFFFF8787 0xFFFFFFFF
      (1 Round)               |               (1 Round)
0xF0000787 0x80780000         |         0xFFFFFFFF 0xFFFFFFFF
      (1 Round)               |               (1 Round)
0x80780000 0x00000700         |         0xFFFFFFFF 0xFFFFFFFF
          |___________________|              <ciphertext>
```

Fig. 6. The Alpha Property

We note that this property is perfectly symmetric (encryption/decryption).

4.3 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, similar as in Fig. 6.

We have the following result:

Fact 4.4 *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 3.1 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 3.1) 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 [35], 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 (cf. [7, 12]):

Fact 5.1 *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. This is illustrated in Fig. 7 below.*

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.

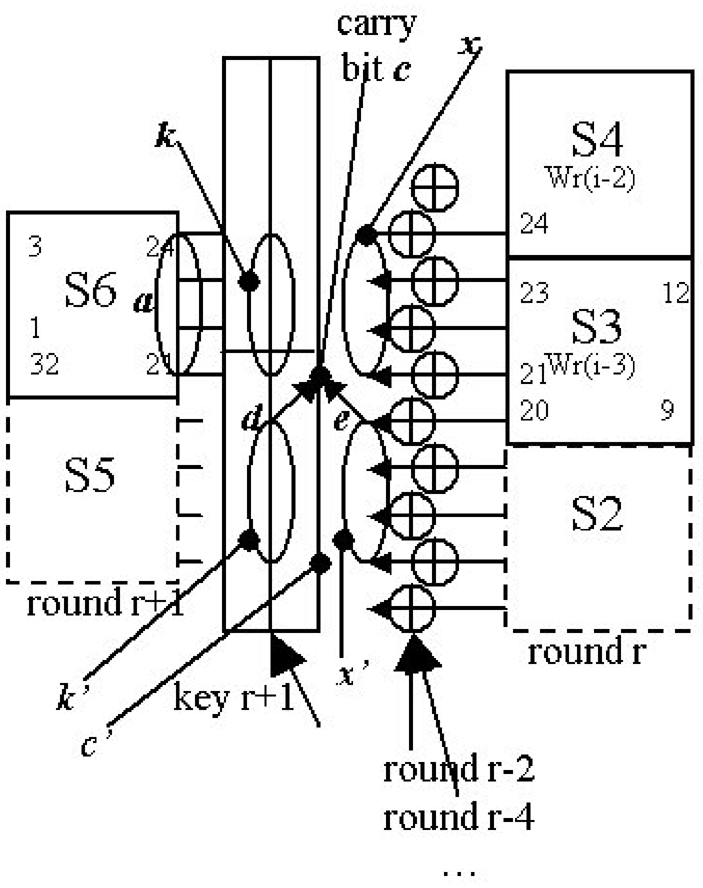

Fig. 7. Computation of the Input Of One S-box With A Carry Bit

From here we easily obtain that:

Fact 5.2 *Let $i > 1$ (with S1 there is no carry entering as in 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, e be respectively the most significant bits of k' and x'. The bit d is 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,*
   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 $c = 1$ which are more or less equally likely. Here we get exactly 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 and the state of only 2 S-boxes in the previous round.

2. *If the attacker knows the whole 4-bit x' he can compute $k'+x'$ with an interval of incertitude of 8 instead of 16 previously. Thus only with probability 1/4 there will be two answers. Thus on average we obtain $1/4 \times 2 + 3/4 \times 1 = 1.25 = 2^{0.3}$ possibilities for a.*
3. *The same happens if the attacker knows the whole 4-bit k' but not x'.*
4. *If the attacker knows k' AND the whole 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 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. Other results are obtained in the same way.

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 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 $c = 1$, with different values for 4 outputs of the current 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. As we advance in the attack tree the time complexity may increase or decrease, and the probability to arrive at this level for a particular set of choices decreases with many early aborts: tree branches which do need to be explored only with low probability.

6.1 Attack Stage 1 - First 4 and Last 4 Rounds

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 cf. Fig. 6 which holds for $2^{13} + 2^{11.9}$ distinct pairs i, j of encryptions for the full 32=6+20+6 rounds, cf. Fact 4.4.
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 reduce the number of pairs under attack and 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 3 then 4 and the last 3,4 rounds in our property Alpha, cf.Fig. 8.

```
     <plaintext>              ------------------->
0xFFFFFFFF 0xFFFFFFFF         |        0xF0000787 0x807FFF80
      (2 Rounds)              |               (1 Round)
0xFFFFFFFF 0xFFFF8787    (24 inner)    0x807FFF80 0xFFFF8787
      (1 Round)           (rounds)            (1 Round)
0xFFFF8787 0x807FFF80     (of GOST)    0xFFFF8787 0xFFFFFFFF
      (1 Round)               |               (2 Rounds)
0x807FFF80 0xF0000787        / \       0xFFFFFFFF 0xFFFFFFFF
          |__________________|              <ciphertext>
```

Fig. 8. First 4 and last 4 rounds in the Alpha Property of Fig. 6

- 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 our Alpha property imposes after round 2.
- The output after the addition of the output of S4,S5,S6,S7 after round 3 gives 15 inactive difference bits at 0 which are 24-31,1-7 (excluding bit 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 being 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+64} times the cost of computing roughly speaking 1 round of GOST.
    It is not needed 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 3+ and the last 3+ 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.

In Fig. 9 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^{179}

Fig. 9. Summary of Major Steps In Our Attack on GOST

6.2 Attack Stage 2 - Working on Rounds 5 and 28

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 will be able later at Stage 4 apply the distinguisher given by Fact 4.4.

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 after round 5, cf. earlier Fig. 6 page 9 or Fig. 10 page 17.
   First we will work on S8, then on S1, then on S2.
2. First we guess additional 4+4 key bits. The situation is the same as in Fig. 7 with boxes S78 at round 5 depending mostly on boxes S456 in 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 5.2).
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 5.2.3. 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.

```
     <plaintext>            ------------------->
0xFFFFFFFF 0xFFFFFFFF         |        0x00000700 0x80780000
      (2 Rounds)              |             (2 Rounds)
0xFFFFFFFF 0xFFFF8787    (20 Rounds)   0xF0000787 0x807FFF80
      (1 Round)            (or RP)          (1 Round)
0xFFFF8787 0x807FFF80     (or other)   0x807FFF80 0xFFFF8787
      (1 Round)               |             (1 Round)
0x807FFF80 0xF0000787        / \       0xFFFF8787 0xFFFFFFFF
      (2 Rounds)              |             (2 Rounds)
0x80780000 0x00000700         |        0xFFFFFFFF 0xFFFFFFFF
          |___________________|            <ciphertext>
```

Fig. 10. First 6 and last 6 rounds in the Alpha Property of Fig. 6

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 , cf. Fig. 10. 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 5.2. 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 5.2.1.), and otherwise we have two cases to include (when $d \neq e$, cf. Fact 5.2.1.).
   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 4.4. 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 3.1 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 6.5 *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 [35] 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 [35] gives a probability only about 2^{-24}.

6.6 Attack Stage 5

We need to do some additional guessing and filtering. Up till now, with total time of up to 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 some $2^{155.2+28-5-5} = 2^{173.2}$ 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 $2^{8.2} + 2^{10}$ is outside $2^{5.9}$ standard deviations. Here the Gauss error function [35] gives a figure much smaller than 2^{-256}.

Summary: Thus given 2^{64} KP and in an average time of about 2^{179} 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.

Applicability: Current attack was optimized for just one set of GOST S-boxes. The space of possible variants ot this attack is very large It is very much premature to claim [32] that it would not work for a certain well-designed set of S-boxes [32, 31]. On the contrary. Similar results exist for any set of S-boxes [13, 8, 29]. We conjecture that for any set of bijective S-boxes in GOST (the worst case) there is a differential attack substantially faster than brute force and very similar to the one presented in this paper.

7 Conclusion

GOST 28147-89 is a well-known block cipher and a Russian government standard. In his 1994 book [34] Schneier has written 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 [21, 20]. In the same year Japanese researchers [33] show that more powerful differential attacks exist, exploiting sets of differentials [33]which allow to break about 13 rounds of GOST out of 32. Many new attacks on GOST have been proposed since 2011 [3, 16, 23, 6, 4, 5, 19, 12, 11, 17] including new combined attacks which also exploit multiple differentials. In 2011 Courtois and Misztal have found new differential sets for GOST [6] most of which can also be seen as "truncated" differential attacks [27]. If one exploits the key scheduling one can break full GOST faster than brute force [4]. This attack was further improved in [5] to achieve about 2^{224}. In a recent paper about advanced differential cryptanalysis, we read: *Truncated differentials, [...] in some cases allow to push differential attacks one or two rounds further.* In this paper we gain not 1 or 2 but much closer to 20 rounds, compared to previous more basic differential attacks [20, 21, 33].

The main result of this paper is a multi-stage 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 2^{179} GOST computations on average for a success probability of 95 %. The memory is about 2^{70} bytes. This is the fastest **single-key** attack on GOST found so far. The best previous single-key attack on GOST was 2^{192} of [19] which could be improved to 2^{191} in [16]. Our 2^{179} is an inexact result assuming independence of certain events. At this moment the attack was optimized only for one set of S-boxes.

In practice ciphers are NOT used with single keys. Faster and more realistic attacks exist when we are dealing with **multiple keys generated at random**, cf. [16, 11, 24, 17]. Numerous such attacks use very similar truncated differential properties as in this paper or more advanced properties with 3 or 4 points cf. [16, 17]. Many such attacks also require only 2^{32} of data per key instead of 2^{64} in this paper, cf. [16, 17]. One such attack allows to recover a full GOST key at a total cost as low as 2^{101} GOST computations total, cf. [16, 17].

References

1. 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/`.
2. Eli Biham, Adi Shamir, *Differential cryptanalysis of the full 16-round DES,* In Crypto'92, pp. 487-496, LNCS 740, Springer-Verlag, 1992.
3. Nicolas Courtois: *Security Evaluation of GOST 28147-89 In View Of International Standardisation,* in Cryptologia, Volume 36, Issue 1, pp. 2-13, 2012.
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,* 14 June 2011, `http://eprint.iacr.org/2011/312` .
6. 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. 11–26, Springer.
7. Nicolas Courtois: *An Improved Differential Attack on Full GOST,* 15 March 2012, `http://eprint.iacr.org/2012/138` .
8. 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.*
9. *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` .
10. 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.
11. 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` .
12. Nicolas T. Courtois: *Low-Complexity Key Recovery Attacks on GOST Block Cipher,* In Cryptologia, Volume 37, Issue 1, pp. 1-10, 2013.
13. Nicolas T. Courtois, Theodosis Mourouzis: *Enhanced Truncated Differential Cryptanalysis of GOST,* in SECRYPT 2013, Reykjavik, July 2013, `http://www.nicolascourtois.com/papers/sec13.pdf` .
14. 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` .
15. Nicolas Courtois, Theodosis Mourouzis, Anna Grocholewska-Czurylo and Jean-Jacques Quisquater: *On Optimal Size in Truncated Differential Attacks,* In post-proceeding of CECC 2014 conference, Stud. Scient. Math. Hungarica, 2015.
16. Nicolas Courtois: *Algebraic Complexity Reduction and Cryptanalysis of GOST,* Monograph study on GOST cipher, 2010-2014, 214 pages, available at `http://eprint.iacr.org/2011/626`.
17. Nicolas Courtois: *On Multiple Symmetric Fixed Points in GOST,* in Cryptologia, Iss. 4, vol 39, 2015, pp. 322-334, `http://www.tandfonline.com/doi/full/10.1080/01611194.2014.988362`
18. Don Coppersmith, *The development of DES, Invited Talk, Crypto'2000, 8'2000.*

19. *Itai Dinur, Orr Dunkelman and Adi Shamir: Improved Attacks on Full GOST,* FSE 2012, LNCS 7549, pp. 9-28, 2012.
20. V.V. Shorin, V.V. Jelezniakov, E.M. Gabidulin: *Linear and Differential Cryptanalysis of Russian GOST,* submitted Elsevier preprint, 4 April 2001.
21. V.V. Shorin, V.V. Jelezniakov, E.M. Gabidulin *Security of algorithm GOST 28147-89,* (in Russian), In Abstracts XLIII MIPT Science Conf. 8-9 December 2000.
22. 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.
23. Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, pp. 290-305, Springer LNCS 6733, 2011.
24. Orhun Kara and Ferhat Karakoç: *Fixed Points of Special Type and Cryptanalysis of Full GOST.* In CANS 2012, LNCS 7712, pp 86-97, 2012.
25. L.V. Kovalchuk: *Generalized Markov ciphers: evaluation of practical security against differential cryptanalysis,* in: Proc. 5th All-Russian Sci. Conf. "Mathematics and Safety of Information Technologies" (MaBIT-06), 25-27 Oct. 2006, MGU, Moscow, pp. 595-599, 2006 [in Russian].
26. 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` .
27. Lars R. Knudsen: *Truncated and Higher Order Differentials,* In FSE 1994, pp. 196-211, LNCS 1008, Springer.
28. 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.*
29. *Theodosis Mourouzis: 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` .
30. 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.
31. 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*
32. *Vladimir Rudskoy, Andrey Dmukh: Algebraic and Differential Cryptanalysis of GOST: Fact or Fiction,* In CTCrypt 2012, Workshop on Current Trends in Cryptology, 2 July 2012, Nizhny Novgorod, Russia. 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`
33. Haruki Seki and Toshinobu Kaneko: *Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, LNCS 2012, pp. 315-323, Springer, 2000.*
34. *Bruce Schneier: Section 14.1 GOST,* in *Applied Cryptography,* Second Edition, John Wiley and Sons, 1996. ISBN 0-471-11709-9.
35. Standard Deviation – wikipedia article, 13 June 2011, available at `http://en.wikipedia.org/wiki/Standard_deviation`.