VERSITA

TATRA MOUNTAINS
Mathematical Publications
DOI: 10.2478/tmmp-2013-0035
Tatra Mt. Math. Publ. **57** (2013), 45–63

CRYPTANALYSIS OF GOST IN THE MULTIPLE-KEY SCENARIO

Nicolas T. Courtois

ABSTRACT. GOST 28147-89 is a well-known 256-bit block cipher. In 2010 GOST was submitted to ISO, to become an international standard. Then many academic attacks which allow to break full GOST faster than brute force have been found. The fastest known single-key attack on GOST for 2^{64} of data is 2^{179} of [Courtois, N.: *An improved differential attack on full GOST,* Cryptology ePrint Archive, Report 2012/138, `http://eprint.iacr.org/2012/138`] and for 2^{32} of data it is 2^{191} of [Courtois, N.: *Algebraic complexity reduction and cryptanalysis of GOST,* Preprint, 2010–13, `http://eprint.iacr.org/2011/626`]. Other results are slower but require significantly less memory [Courtois, N.: *Algebraic complexity reduction and cryptanalysis of GOST,* Preprint, 2010–2013, `http://eprint.iacr.org/2011/626`], [Dinur, I.—Dunkelman, O.—Shamir, A.: *Improved attacks on full GOST,* in: Fast Software Encryption—FSE '12, 19th Internat. Workshop, Washington, USA, 2012, Lecture Notes in Comput. Sci., Vol. 7549, Springer, Berlin, 2012, pp. 9–28, `http://eprint.iacr.org/2011/558/`]. The common stereotype is that these will be "the best" attacks on GOST. However, ciphers are not used in practice with single keys, on the contrary. In this paper we intend to show that there exist attacks on GOST which are more versatile and even somewhat more "practical" than the best single key attack. We argument that multiple random key attacks not single key attacks, are more practical and more likely to be executed in the real life. They recover keys when other attacks recover none. One can break some (weaker) GOST keys in a population of 256-bit keys generated at random with a TOTAL computational effort as low as 2^{120}, this including the cost to examine also the cases in which the attack does not work. All our attacks are based on special non-trivial properties of data inside the cipher which however are such that keys for which the property does not hold can be rejected efficiently.

1. The GOST encryption standard and its security

The Russian encryption standard GOST 28147-89 is an important government standard [25] which is also widely used by large banks [15], [24], on the Internet with OpenSSL [15], and in entreprise security systems (e.g., RSA Labs). Its large

Keywords: block ciphers, Feistel schemes, key scheduling, self-similarity, fixed points, reflection attacks, black-box reductions.

key size of 256 bits and exceptionally low implementation cost [20] make GOST a plausible alternative for AES-256 and triple DES.

GOST is a block cipher with a simple Feistel structure, 64-bit block size, 256-bit keys and 32 rounds. Each round contains a key addition modulo 2^{32}, a set of 8 bijective S-boxes on 4 bits, and a simple rotation by 11 positions. The most complete current reference implementation of GOST in the OpenSSL library contains eight standard sets of S-boxes [15]. The attacks we consider in this paper, as well as most other recent attacks on GOST from [6], [7], [17] work with a very similar complexity whatever are the S-boxes used in GOST. Until 2011, no cryptographically significant attack on GOST used in encryption was found, which was summarized in 2010 in these words: "despite considerable cryptanalytic efforts spent in the past 20 years, GOST is still not broken", see [20]. According to Biryukov and Wagner, the structure of GOST, and in particular the reversed order of keys in the last 8 rounds, makes it secure against slide attacks [1]. However, this exact inversion of keys allows other so called "Reflection" attacks. This property marks a turning point in the security of GOST. Initially at Indocrypt 2008 only a weak-key attack with time complexity of 2^{192} is proposed, with large proportion of 2^{-32} of weak keys. Then in 2011 several attacks on regular GOST keys have been discovered, where by regular we mean typical keys generated at random, this including single key attacks. More than half of these new attacks use this reflection property, sometimes twice, three or four times [6], [17]. Interestingly many other attacks allow to break GOST without this reflection property [6], [7], [14]. Some are fixed point or involution attacks, but some are entirely new sorts of self-similarity attacks. Most of these attacks follow the framework of "Black-Box Complexity Reduction" from [6], [7] where the problem of attacking full 32-round GOST is reduced to a low-data complexity attack 8 rounds. Unhappily the quantity of data available after reduction is very small, for example 2, 3 or 4 pairs for a reduced cipher. Very few low-data complexity attacks are known. Accordingly the last step in these attacks is typically either a Meet-In-The-Middle attack [6], [14], [17] or a software algebraic attack [6], [7] and recently a combination of both [6], [8]. The best single-key attacks in this category are an attack with about 2^{192} GOST computations [6], [14] with 2^{64} of data and about 2^{224} GOST computations with 2^{32} of data [6], [14], [17].

In this paper we look at the possibility of obtaining strictly better "Complexity Reductions" for some weaker keys. However, we are not interested in weak key attacks. We are interested exclusively in attacks which can recover 256-bit GOST keys generated at random with a total effort being strictly lower than for a comparable single key attack. The main point is that weak keys and related keys **CAN** be exploited in the real life attacks **IF** they occur with sufficiently high "natural" probability for keys generated at random. The general attack scenario on block ciphers, relevant to say a government security agency, is **the multiple key scenario** as follows: given 2^X devices with distinct keys, 2^Y of data per device, and $M = 2^Z$ of memory, some keys can be recovered at the total cost of $T = 2^Z$. It is surprising that this scenario is almost never studied in cryptography. This is probably because researchers are not aware of the fact that this multiple key scenario allows to recover (at least some) keys in situations where other attacks find none (and in particular all known single key attacks). Can we demonstrate such a thing for GOST? One basic question is as follows: given a well-chosen assumption which works for 2^{-A} of GOST keys, can we obtain an attack which is more than 2^A times faster the the best known single-key attack? If so we can afford to check also all the cases in which the attack does not work and overall obtain a more realistic attack than the reference single key attack. More importantly, in many cases rejecting the cases where the attack does not work will be easier than expected. We have designed many such attacks and the security of GOST degrades surprisingly quickly as X grows. In fact we obtain a whole spectrum of attacks and will show that the cost of computing one 256-bit GOST key decreases in a very important way down to 2^{120} in this multiple key scenario. This at the price of increasing over data complexity 2^{X+Y} even though in many attacks we need only 2^{32} of data per device. Somewhere half way there will be attacks where 2^{X+Y} is not too large and the computational effort is going to be more realistic than any other attack on GOST.

This paper is organized as follows: In Section 2 we cover brute force and reduced-round attacks on GOST. In Section 3 we explain how the paradigm of Black Box Reduction applies to GOST and in particular we explain the reflection property as consequence of a particular decomposition of GOST. In Section 4 we study 4 basic families of weak key attacks and the detailed cost of converting them to a "real" attack on multiple random keys. Our results are summarized in Table 1 Section 5 followed by a conclusion.

2. Preliminary remarks on GOST

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

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

Fact 2.0.1. 4 P/C pairs are necessary to determine the GOST key. With 4 P/C pairs we expect to get on average about one key. We get the correct key together with a list of, sometimes a few, but on average less than 1 wrong keys.

With 5 P/C pairs we are able to discard all these wrong keys in practice.

Fact 2.0.2. A brute force attack on GOST takes 2^{255} GOST encryptions on average.

Justification: We proceed as follows: we use one P/C pair and check all the possible keys. On average half way through the right key will be found. Only for an expected number of 2^{191} keys which are confirmed with the first pair, we do further comparisons. Most of these 2^{191} are **false positives**. The key remark is that the total number of these false positives is small and the complexity of rejecting all the incorrect keys with additional P/C pairs is actually negligible.

2.1. Low data complexity attacks on reduced-round GOST

Software algebraic attacks, can be defined as attacks in which the problem of key recovery is written as a problem of solving a large system of Boolean algebraic equations which follows the geometry and structure of a particular cryptographic circuit [4]. For block ciphers these attacks typically work only for a limited number of rounds, for example to break 6 rounds of DES, but this is possible to do given only 1 known plaintext, see [4]. Our most recent results are obtained as a combination of earlier software algebraic attacks on DES [4] with dedicated optimizations for GOST described in [13] and with highly optimized guess-then-determine steps inspired by Meet-In-The-Middle attacks. Following [6]:

Fact 2.1.1 (Key Recovery for 8 Rounds and 3 KP)**.** Given 3 P/C pairs for 8 rounds of GOST we can produce 2^{64} candidates for the 256-bit key in time equivalent to 2^{110} GOST encryptions. The storage requirements are negligible and all the 2^{64} candidates can be produced in a uniform way, each of them is produced in time of 2^{56} GOST encryptions on average.

Full justification of this fact takes many pages and appears in Appendix N.3. page 94 of [6]. A faster attack with only 2^{107} GOST encryptions BUT at the expense of MUCH larger memory is given in [8]. Similarly we have:

Fact 2.1.2 (Key Recovery for 8 Rounds and 4 KP)**.** Given 4 P/C pairs for 8 rounds of GOST we can recover the full 256-bit key in time equivalent to 2^{94} GOST encryptions with negligible memory.

In [8] we find two distinct attacks with complexity of 2^{94} for 4 KP. One is an excessively technical MITM attack with large memory, another is a super simple software/algebraic/SAT-solver attack with same running time and negligible memory. The trick is to mimic the structure of a MITM attack and aim at a software/algebraic inversion attack. The attacker guesses the exact key bits 0–15,51–55,64–66,128–130,179–183,192–207,224–231,244–255 and is able to determine all the other by software.

3. Cryptanalysis with Black-Box Complexity Reduction

Following [6], [7] the work of cryptanalyst working on cryptanalysis of GOST can be split into two independent tasks. First task is how to achieve a software algebraic attack on a reduced-round version as discussed in the previous section. The second question is if and **how** the complexity major variants of full GOST with 32 rounds can ever be **reduced** to a problem of breaking a cipher with significantly less rounds. Only then we can ever hope to be able to apply results such as Fact 2.1.2. Recently it became possible to design and implement an appropriate last step for many such attacks, cf. Fact 2.1.2 and many other in [6], [8], [14].

The main idea is as follows [6], [7]. In order to reduce the attack complexity, we exploit the self-similarity of the cipher (due, e.g., to a weak key schedule) and add some well-chosen assumptions which produce interesting and sometimes quite non-trivial consequences due to the high-level structural properties of the cipher, which makes cryptanalysis problems smaller, simpler and easier to solve. In this process we need to minimise the costs (in terms of probability that our assumptions hold) and to maximise the benefits (in terms of the number and the complexity of interesting relations which hold under these assumptions).

This process is called **Algebraic Complexity Reduction**, see [6], [7]. In most cases what we get is to compute (guess or determine) many internal values inside one or several decryptions, and literally break the cipher apart into smaller pieces. In particular we have **Black-Box Algebraic Complexity Reductions** where we obtain real black-box reductions, for example, to the same cipher with strictly less rounds (and less data) again at the cost of some well-chosen assumptions. This creates new important **optimisation problems** in symmetric cryptanalysis. Reductions exploit self-similarity of different blocks and their inverses, fixed points in certain components, and reflections [6], [7]. Reductions can be compared in terms of the number of pairs obtained, the resulting reduced number of rounds, success probability, and in terms of data complexity. In this paper we focus on weak key attacks which however are valuable **only** if and because, we will be able to achieve time complexities better than any attack on regular keys from [6], [7], [10], [14].

3.1. High-level structure and properties of GOST

GOST is a Feistel cipher with 32 rounds. In each round we have a round function $f_k(X)$ with a 32-bit key which uses a 32-bit segment of the original 256-bit key which is divided into eight 32-bit sub-keys $k = (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7)$. This pattern is repeated 3 times and the the order of keys is inverted.

Notation: Let $\mathcal{S}$ be the function which swaps two halves of a 64-bit word, i.e., $\mathcal{S}(L|R) = R|L$. We denote the swapped value by notation $\overline{A}$, i.e., $\mathcal{S}(A) = \overline{A}$. In figures we denote $\mathcal{S}$ by an explicit graphical notation $\bowtie$.

Following [18] one can write GOST as the following functional decomposition (to be read from right to left)

$$\mathrm{Enc}_k = \mathcal{D} \circ \mathcal{S} \circ \mathcal{E} \circ \mathcal{E} \circ \mathcal{E}, \tag{1}$$

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

Fact 3.1.1 (Internal Reflection Property)**.** Consider the last 16 rounds of GOST $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$ for one fixed GOST key. This function has an exceptionally large number of fixed points: applied to X gives the same value X with probability 2^{-32} over the choice of X, instead of 2^{-64} for a random permutation.

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

4. Weak key attacks on GOST

Weak keys offer a considerable degree of extra freedom to the attacker. Let d denote the density of keys for which a given attack works, defined as the probability that the attack will work for a key chosen uniformly at random. Single key attacks work for typically more than half of all keys, with $d \geq 0.63$ or better [6], [7]. In this paper we will have d between 2^{-32} and 2^{-64}. It is plausible that such weak keys would occur in the real life. For example, given that the population of our planet is about 2^{33}, and one person can use during their life many cryptographic keys, an attack with $d = 2^{-32}$ should be considered as nearly-realistic; it is plausible to assume that at some moment in the future 2^{32} different GOST keys will be used worldwide and some keys will become vulnerable to our attacks.

4.1. Weak Key Family 0

Fact 4.1.1 (Weak Keys Family 0, $d = 2^{-32}$, reduction to 1 KP for 8R)**.** We define the Weak Keys Family 0 by keys such that $\mathcal{E}$ has a fixed point A which is symmetric, i.e., $\overline{A} = A$. This occurs with density $d = 2^{-32}$.

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

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

4.1.1. Key Recovery with Family 0

In [18], this method of Fact 4.1.1 is used to recover keys with time complexity of 2^{192} and negligible memory. This is very hard to improve because the attack uses only 1 KP for 32 rounds, and there are 2^{192} keys for which this pair is correct, and all these keys must be checked against additional P/C pairs for the full 32-rounds. In the next sections we will introduce new families of weak keys able to achieve time complexities below 2^{192}.

4.2. Weak Key Family 2.1

In this section we exhibit another family of weak keys, with the same density $d = 2^{-32}$ but with more extensive possibilities. It can be seen as an extension of the original double reflection attack described at Fig. 1 in Section 11 of [6] and the resulting double reflection attack. If we require that B is also symmetric, we will be able to simultaneously improve the probability of our guess being true, from 2^{-128} to 2^{-64} to obtain 4 P/C pairs for 8 rounds, and reduce the data complexity. We obtain a triple reflection attack and it works only for weak keys but the probability of these keys is quite large $d = 2^{-32}$. We call it not Family 1 but Family 2.1 following a more extensive classification of attacks on GOST in [6].

Fact 4.2.1 (Weak Keys Family 2.1, $d = 2^{-32}$, getting 3,4 and 5 KP for 8R)**.** We define the Weak Keys Family 2.1 by keys such there exists A such that all the three values $\mathcal{E}(A)$, $\mathcal{E}^2(A)$ and $\mathcal{E}^3(A)$ are symmetric. This occurs with density $d = 2^{-32}$. For every key in Family 2.1, we have the following four reductions:

1) with 2^{32} CC we obtain 3 P/C pairs for 8 rounds of GOST correct with $P = 2^{-64}$, (where CC means Chosen Ciphertexts)
2) with 2^{32} ACC (Adaptive Chosen Ciphertexts) we obtain 4 P/C pairs for 8 rounds of GOST correct with $P = 2^{-64}$,
3) with 2^{32} CC we obtain 4 P/C pairs for 8 rounds of GOST correct with $P = 2^{-96}$,
4) with 2^{32} ACC we obtain 5 P/C pairs for 8 rounds of GOST correct with $P = 2^{-96}$.

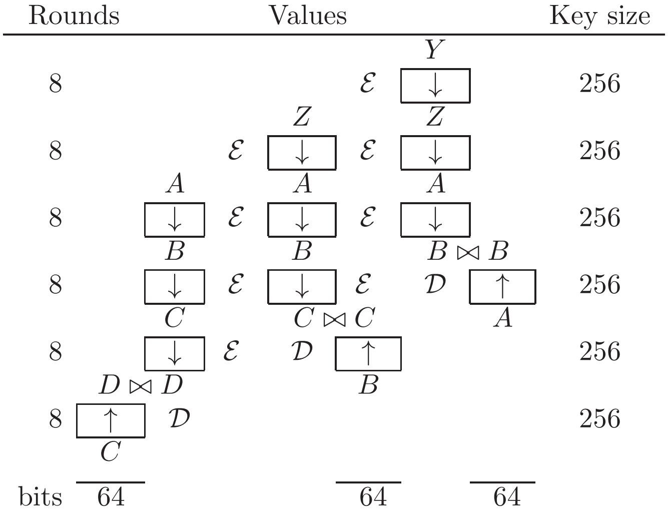

FIGURE 1. A weak-key black-box reduction which gives up to 5 pairs for 8 rounds.

Justification: It is easy to see that the event $\mathcal{E}(A)$ AND $\mathcal{E}^2(A)$ AND $\mathcal{E}^3(A)$ being symmetric leads to key density $d = 2^{64-32-32-32} = 2^{-32}$. The attack is shown on Fig. 1, where $\bowtie$ denotes $\mathcal{S}$ (swapping the two 32-bit halves).

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

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

1. We guess two symmetric values B, C. They are both correct with $P = 2^{-64}$.
2. We determine A, Z by decrypting B and C which are both symmetric.
3. We get 3 pairs $\mathcal{E}(Z) = A$, $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$ and our guess is correct with probability 2^{-64}. So far need 2^{32} Chosen Ciphertext (CC).
4. Furthermore, if we also decrypt A we get Y and obtain one additional pair $\mathcal{E}(Y) = Z$. This is obtained at the price of 2^{32} ACC (Adaptive Chosen Ciphertexts). Here though A is not symmetric we do NOT need 2^{64} KP. We can decrypt all possible A obtained from decryption of all possible symmetric values C. This requires 2^{32} ACC.
5. Going one step backwards, we don't decrypt A but also guess D which is symmetric, We get 4 pairs $\mathcal{E}(Z) = A$, $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$, $\mathcal{E}(C) = D$ and our guess is correct with probability 2^{-96}. Again we need only 2^{32} CC.

6. Now if we combine guessing D and decrypting A, we get 5 pairs given 2^{64} KP and our guess is correct with probability 2^{-96}. We need 2^{32} ACC (Adaptive Chosen Ciphertexts).

Out of four possibilities given in Fact 4.2.1, we will use two in order to obtain two new weak key attacks.

Fact 4.2.2 (Key Recovery for Family 2.1, 2^{32} CC, $d = 2^{-32}$)**.** One can recover the keys for the Weak Keys Family 2.1 with 2^{32} CC, running time of 2^{174} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the first reduction of Fact 4.2.1 and of Fact 2.1.1 which allows to enumerate a set of solutions and the time is 2^{64+110} GOST encryptions. The total number of full 256-bits keys which are false positives which need to be checked against additional P/C pairs for the full 32 rounds of the cipher is comparatively smaller, about 2^{128}, which is unlikely to influence the overall complexity of the attack which will be 2^{174} GOST encryptions.

Similarly, with just one more pair $\mathcal{E}(Y) = Z$, we obtain

Fact 4.2.3 (Faster Key Recovery for Family 2.1, 2^{32} ACC, $d = 2^{-32}$)**.** One can recover the keys for the Weak Keys Family 2.1 with 2^{32} ACC, running time of 2^{158} GOST encryptions and with negligible memory.

Justification: Here we replace 3 KP by 4 KP and Fact 2.1.1 with 2^{110} by Fact 2.1.2 with 2^{94}. We need a total time of $2^{64+94} = 2^{158}$ GOST encryptions.

4.3. An Attack with ”Regular” Keys and 2^{32} of Data per Device

In this sub-section we are going to convert our “weak key” attack into a “regular” attack with random 256-bit keys. The following attack scenario is relatively plausible 2^{32} devices with different GOST keys. In this plausible “Multiple Key” scenario we are going to show how to break GOST with total cost of only 2^{190} per key found and only 2^{32} of data per device.

Fact 4.3.1 (Key Recovery for a Diverse Population of Keys, $d = 2^{-32}$)**.** If we have a diverse population of at least 2^{32} different keys, with access to 2^{32} ACC per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{190} GOST encryptions (the total of data required is 2^{64}).

Justification: We apply Fact 4.2.3 to each of the 2^{32} devices with random keys. We recover one key out of 2^{32} in total time of 2^{32+158} including the time to check all the other devices. In one case on average, the attack will work and output a valid key which can be checked with additional pairs for that device. For the other devices the attack fails and we abort it after some 2^{158} GOST encryptions.

Quite remarkably the data complexity is only 2^{32} per device. Single key attacks on GOST with 2^{32} of data had always a cost of at least 2^{224} per key found, see [6], [14], [17]. We are the first to achieve 2^{190} GOST encryptions per key recovered, in a scenario with a realistic number of devices with different random 256-bit keys and with reasonable memory, beating all known attack on both time and memory. We have weak keys which ARE frequent enough to occur in the real life and they are also individually CHEAP enough to break, in order to worry about this attack overall, which is reflected by the total cost per key being 2^{190} GOST encryptions.

4.4. Weak Key Family 3

In this section we explore if better attacks exist, and in particular attacks with complexity less than 2^{128}, at the price of further decreasing the density of weak keys to values which are significantly less realistic but still non-negligible.

Fact 4.4.1 (Weak Keys Family 3, $d = 2^{-64}$, getting 4 KP for 8R)**.** We define the Weak Keys Family 3 by keys such there exists A such that $\mathcal{E}(A) = \overline{A}$, $\mathcal{E}^2(\overline{A}) = A$. This occurs with density $d = 2^{-64}$. For every key in Family 3, we have the following: with 2^{64} KP we obtain 4 P/C pairs for 8 rounds of GOST, correct with probability of roughly about $P = 2^{-1}$.

Justification: We proceed as follows:

1. First we observe that A is a fixed point for $\mathrm{Enc}_k(\cdot)$. Indeed

$$\mathrm{Enc}_k(A) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}^3(A)\big)\Big) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}^2(\overline{A})\big)\Big) = \mathcal{D}\big(\mathcal{S}(A)\big) = \mathcal{D}(\overline{A}) = A.$$

   Therefore given 2^{64} KP we can identify A. Due to other possible fixed points, our guess will be correct with probability roughly about $P = 2^{-1}$.

2. Moreover if we define $B = \mathcal{E}(\overline{A})$ we have $A = \mathcal{E}(B)$ and

$$\mathrm{Enc}_k(\overline{A}) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}^3(\overline{A})\big)\Big) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}(A)\big)\Big) = \mathcal{D}\big(\mathcal{S}(\overline{A})\big) = \mathcal{D}(A) = B.$$

   Therefore we can determine B from A.

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

$$\begin{aligned}\mathrm{Enc}_k(B) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}^3(B)\big)\Big) &= \mathcal{D}|bl(\mathcal{S}\big(\mathcal{E}^2(A)\big)\Big) \\ &= \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}(\overline{A})\big)\Big) = \mathcal{D}\big(\mathcal{S}(B)\big) = \mathcal{D}(\overline{B}) = C.\end{aligned}$$

   with the property that $\mathcal{E}(C) = \overline{B}$.

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

Fact 4.4.2 (Key Recovery for Weak Keys Family 3, $d = 2^{-64}$)**.** One can recover the keys for the Weak Keys Family 3 with 2^{64} KP, running time of 2^{95} GOST encryptions and with negligible memory.

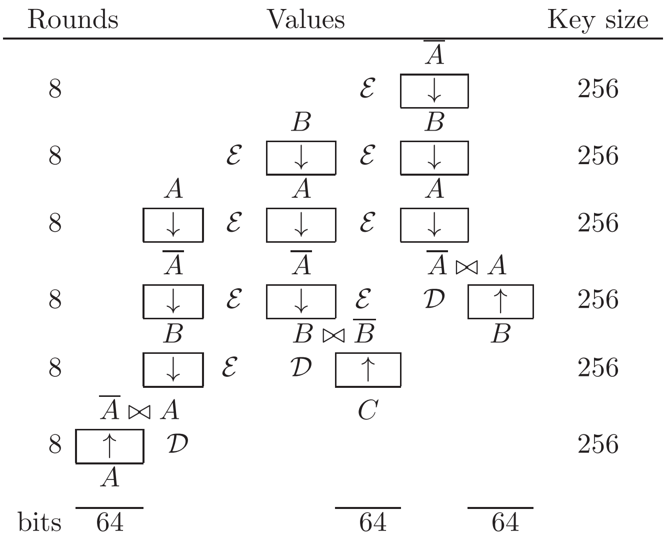

FIGURE 2. Weak Key Family 3 which gives 4 pairs for 8 rounds.

Justification: This is obtained by combination of the current reduction of Fact 4.4.1 and Fact 2.1.2 for 4 KP. We estimate that the probability that we can guess correctly which fixed point A is the correct one is about half, and accordingly on average Fact 2.1.2 needs to be applied twice. The total number of false positives which need to be checked against additional P/C pairs for the full 32 rounds is small and can be neglected.

4.5. Attack on Regular Random 256-bit Keys with Total Time 2^{159}

FACT 4.5.1 (Key Recovery for a Diverse Population of Keys)**.** If we have a diverse population of at least 2^{64} different keys, with access to 2^{64} KP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{159} GOST encryptions.

Justification: We apply the Fact 4.4.2 to 2^{64} random devices, for one of them on average in time 2^{95} GOST encryptions it will output a valid key which can be checked with additional pairs. We abort all the other cases at 2^{95} GOST encryptions.

This actually means that GOST keys can be recovered in overall total time of 2^{159} per key for multiple keys generated at random, which is substantially less than 2^{179} from [10].

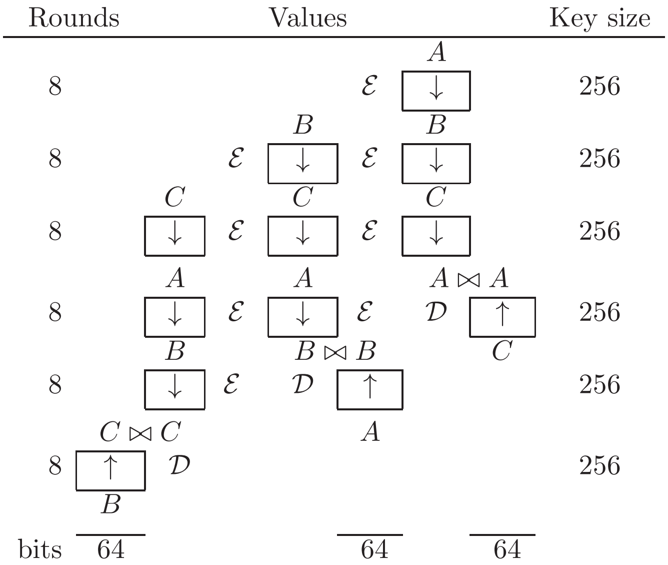

Figure 3. Weak Key Family 8.1 With Triple Reflection and 3 KP for 8R.

4.6. Weak Key Family 8.1

Here is another family of weak keys. We call Family 8.1 as it is known under this name in other papers [6], [8]. In fact the paper [8] contains a replacement component for the Family 8.1 attack of this paper with 2^3 faster running time at the expense of much more memory.

Fact 4.6.1 (Weak Keys Family 8.1, $d = 2^{-98}$, getting 3 KP for 8R)**.** We define the Weak Keys Family 8.1 by keys such there exists A such that

$$\mathcal{E}^3(A) = A,$$

and all the three values

$$A, B = \mathcal{E}^2(A), C = \mathcal{E}^3(A)$$

are symmetric (both 32-bit halves are equal). This occurs with density $d = 2^{-98}$.

For every key in Family 8.1, given 2^{32} CP we can obtain 3 P/C pairs for 4 rounds of GOST, correct with probability close to 1.

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

We proceed as follows:

1. We observe that since $\mathcal{E}^3(A) = A$, we also have $\mathcal{E}^3(B) = B$ and $\mathcal{E}^3(C) = C$.
2. We have
$$\mathrm{Enc}_k(A) = \mathcal{D}\Big(\mathcal{S}\big(\mathcal{E}^3(A)\big)\Big) = \mathcal{D}\big(\mathcal{S}(A)\big) = \mathcal{D}(A) = C = \mathcal{D}(A).$$

   In the same way, $\mathrm{Enc}_k(B) = A$ and $\mathrm{Enc}_k(C) = B$.

   If we consider the 3-fold iteration $\mathrm{Enc}_k^3(\cdot)$ we observe that all the three points A, B, C are fixed points for $\mathrm{Enc}_k^3(\cdot)$ and they are symmetric fixed points of $\mathrm{Enc}_k^3(\cdot)$.
3. Given 2^{32} CP (all values we use are symmetric) we can identify A, B, C because they form a cycle of length 3 for $\mathrm{Enc}_k(\cdot)$ with 3 elements and we do not expect that more such cycles exist in which all these elements would also be symmetric. It would be an unlikely event.
4. We get 3 P/C pairs for 8 rounds which are $\mathcal{E}(A) = B$, $\mathcal{E}(B) = C$, $\mathcal{E}(C) = A$ and these are correct with probability close to 1.

Fact 4.6.2 (Key Recovery for Weak Keys Family 8.1, $d = 2^{-98}$)**.** For the weak keys of Family 8.1 one can enumerate 2^{64} candidates given 2^{32} CP, with running time of 2^{110} GOST encryptions and with negligible memory.

Justification: This is obtained by combination of the current reduction of Fact 4.6.1 and Fact 2.1.1 for 3 KP.

4.7. Conversion of Family 8.1 into a "Regular" Attack with Multiple Keys Generated at Random

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

Fact 4.7.1 (Key Recovery for a Diverse Population of 2^{98} Keys)**.** If we have a diverse population of at least 2^{98} different 256-bit GOST keys generated at random, with access to 2^{32} CP per key, one can recover **one** of these 256-bit keys in total overall time of about 2^{120} GOST encryptions (which is less than the time to compute the data needed for the attack).

Justification: We consider $2^{97.6}$ random devices and we do NOT apply the Fact 4.6.2 for each device. Instead we examine the data, 2^{32} CP with the encryption of all symmetric points, to see if our attack is likely to be applicable. We scan for all cases where the encryption is also symmetric and list all cases where this is true. For a random permutation the probability that one symmetric point gives another symmetric point after encryption is 2^{-32} and there are 2^{32} points. Therefore we expect to find one on average but frequently also 0 or several such cases. We examine these few cases for a cycle of length 3. This is going to happen only with probability $2^{-97.6}$. For all except a proportion of about $2^{-97.6}$ GOST keys, all those where the key is not weak, we can reject them by checking 2^{32}

plaintexts as explained here in time of $2^{97.6+32}$ CPU clocks, which happily is significantly less than $2^{97.6+32}$ GOST encryptions, but rather about $2^{119.6}$ GOST encryptions.

For the remaining proportion of $2^{-97.6}$, we expect them to come from Family 8.1 with overwhelming probability. Therefore for all the cases where the key is weak, we enumerate 2^{64} candidates for the correct key in time 2^{110} GOST encryptions due to Fact 4.6.2, and check which key is correct with additional pairs for full 32 rounds. Overall we expect to recover one key in time of about $2^{110} + 2^{119.6} \approx 2^{120}$ GOST encryptions.

5. Summary of our attacks

In Table 1 we compare our new attacks with weak and regular keys generated at random to earlier attacks. With regular keys there was no attack below 2^{192} [6], [14] which is due to having 2^{192} false positives, see [6]. Finally for $d = 2^{-32}$ we obtain two attacks which are below 2^{192}. For $d = 2^{-64}$ we are below 2^{100}.

In the last three lines we compare all these attacks by the cost of 1 key if we dispose of a “sufficiently diverse” population of keys. For example, in the Fam. 3 column we see that an expected proportion of $d = 2^{-64}$ keys have a security level of only $T = 2^{95}$. Again (cf. Fact 4.5.1), if GOST is used with at least 2^{64} different keys, one can recover one of these keys in total time of not more than 2^{159} GOST encryptions indluding the time to examine all the key where the attacks does not work. This is substantially less than 2^{192} from [14] and even 2^{179} from [10]. If we dispose of 2^{159} GOST computations and not more, our attack works while no single key attack can achieve anything useful.

We can also compare all attacks with 2^{32} of data per device. Until now all such attacks required at least 2^{224} in [6], [14], [17]. In this paper for the first time ever we achieve 2^{190} GOST encryptions per key recovered, in the scenario with a realistic number of devices with different random 256-bit keys. Arguably this paper achieves both the cheapest overall attack on GOST ever found with 2^{32} of data per device and also with 2^{64} of data per device. We see that the security of GOST degrades in a very substantial way in the multiple key scenario.

Some very recent attacks on GOST achieve even less, 2^{130} per key [19] and even 2^{120} per key, this however at the price of much higher 2^{100}-ish total data requirements for all the devices combined, see [19] and Family 8.1 in this paper. The paper [8] contains a replacement last step for our Family 8.1 attacks which reduces time complexity by 2^3 at the expense of much higher memory. Other attacks on GOST achieve even substantially less but they introduce additional cryptanalytic techniques such as advanced differential properties with 2, 3 and 4 points cf. [6].

Table 1. Weak key attacks on GOST and “regular” attacks with diverse random keys.

Attack Ref.	[6, 14]	[6, 14]	[6]	[10]	[18]	Fam. 2.1	Fam. 2.1	Fam. 3	Fam. 8.1
Keys density d	0.63		0.63	1	2^{-32}			2^{-64}	2^{-98}
Data/key 32R	2^{32} KP	2^{64} KP	2^{64} KP	2^{64} KP	2^{32} CP	2^{32} CC	2^{32} ACC	2^{64} KP	2^{32} CP
Obtained for 8R	2 KP		3 KP	-	1 KP	3 KP	4 KP		3 KP
Valid w. prob.	2^{-96}	2^{-64}	2^{-64}	-	2^{-1}	2^{-64}	2^{-64}	2^{-1}	2^{0}
Storage bytes	2^{39}	2^{39}	2^{67}	2^{70}	small			2^{67}	small
♯ False positives	2^{128}		2^{128}		2^{192}	2^{64}	2^{-0}	2^{64}	2^{64}
Time 8 R	2^{128}	2^{128}	2^{110}	-	2^{192}	2^{110}	$\mathbf{2^{94}}$	$\mathbf{2^{94}}$	2^{110}
Time 32 R	2^{224}	2^{192}	2^{206}	$\mathbf{2^{179}}$	2^{192}	$\mathbf{2^{174}}$	$\mathbf{2^{158}}$	$\mathbf{2^{95}}$	$\mathbf{2^{110}}$
Cost of 1 key, if	2^{225}	2^{193}	2^{207}	$\mathbf{2^{179}}$	$\mathbf{2^{193}}$	2^{206}	2^{190}	$\mathbf{2^{159}}$	$\mathbf{2^{120}}$
key diversity $\geq$	single key attacks or $>$ 50% of keys				2^{32}			2^{64}	2^{98}
Data x keys	2^{33}	2^{64}	2^{65}	2^{64}	2^{64}			2^{128}	2^{130}

6. Conclusion

This paper studies several interesting weak key attack classes for the block cipher GOST. However, we are interested in such keys ONLY IF they can be transformed into a “regular” attack on GOST, when it is used with multiple 256-bit keys generated at random. Our key results are summarized in Table 1.

We show that the security of GOST degrades surprisingly quickly when the key diversity grows. With 2^{32} different keys generated at random, some keys can be recovered in time as little as 2^{158}. Moreover we already have attacks in 2^{95} for a still non-negligible proportion of 2^{-64} of full 256-bit keys. We observe that $2^{95} \cdot 2^{64} \approx 2^{159}$. Accordingly we can achieve more than just a weak key attack. For each of our weak key attacks we transform it into a “regular” attack on a diverse population of random keys which is overall less costly and more feasible to execute. For example following Fact 4.5.1, given at least 2^{64} different keys, with 2^{64} KP per key, one can recover one of these keys in total time of not more than 2^{159}. This including the time to examine all the other (stronger) keys.

Our results need to be compared to the fastest known single-key attack on GOST. For 2^{64} of data we have time complexity 2^{179} of [10] and for 2^{32} of data it is 2^{191} in [6] or 2^{192} in [14] with slightly less memory. However importantly, ciphers are hardly ever used in practice with single keys. On the contrary. If we do not dispose of a computing power of 2^{179}, none of these attacks is of any use. Is there a multiple key attack which is cheaper and more realistic than the best single key attack? If we dispose of say 2^{159} computations, is it possible to recover some GOST keys? Furthermore, can we recover any 256-bit GOST keys generated at random given only 2^{120} computations? Very surprisingly the answer is yes, we can. Even though this will require astronomical quantities of data, it requires only 2^{32} of data per device.

Our research on GOST have irreversibly changed the way in which we now think about the security of block ciphers in general. Our main point is that the popular notion of a single-key attack is in fact too restrictive and does **NOT** capture all realistic attacks on a given cipher. Single key attacks on GOST in 2^{179} remain infeasible. In contrast given as little as just 2^{120} computations, see Fact 4.7.1, or even less, about 2^{101} cf. further results in [6], one can already recover some GOST keys. Again all these are “regular” attacks in which keys are random and uniformly distributed and weak keys occur with their natural probability. It appears that the multiple random key scenario is stronger and more versatile than the single key scenario, this from a very pragmatic point of view, the cost per key recovered. and if we dispose of sufficient data. We should never again consider that a block cipher which is not broken by a single key attack is secure.

Faster attacks can be obtained if we combine the techniques of this paper with advanced differential properties [10]–[12], which is really outside the scope of this paper. A number of combined attacks with complexity reduction and advanced differential properties with 2, 3 and 4 points and additional dedicated steps allow to bring our 2^{120} per key further down to about 2^{101}, cf. [6].

REFERENCES

[1] BIRYUKOV, A.—WAGNER, D.: *Advanced slide attacks,* in: Advances in Cryptology– –EUROCRYPT '00, 19th Internat. Conf. on the Theory and Appl. of Cryptographic Techniques (B. Preneel, ed.), Bruges, Belgium, 2000, Lecture Notes in Comput. Sci., Vol. 1807, Springer, Berlin, 2000, pp. 598–606.

[2] COURTOIS, N.—PIEPRZYK, J.: *Cryptanalysis of block ciphers with overdefined systems of equations,* in: Advances in Cryptology—ASIACRYPT '02, 8th Internat. Conf. on the Theory and Appl. of Cryptology and Inform. Security (Y. Zheng, ed.), Queenstown, New Zealand, 2002, Lecture Notes in Comput. Sci., Vol. 2501, Springer, Berlin, 2002, pp. 267–287.

[3] COURTOIS, N.: *Fast algebraic attacks on stream ciphers with linear feedback,* in: Advances in Cryptology—CRYPTO '03, 23rd Annual Internat. Cryptology Conf. (D. Boneh, ed.), Santa Barbara, California, USA, 2003, Lecture Notes in Comput. Sci., Vol. 2729, Springer, Berlin, 2003, pp. 176–194.

[4] COURTOIS, N.—BARD, G. V.: *Algebraic cryptanalysis of the data encryption standard,* in: Cryptography and Coding, 11th IMA Internat. Conf. (S. D. Galbraith, ed.), Cirencester, UK, 2007

Lecture Notes in Comput. Sci., Vol. 4887, Springer, Berlin, 2007, pp. 152–169, Preprint `eprint.iacr.org/2006/402/`.

[5] COURTOIS, N.—BARD, G. V.—BOGDANOV, A.: *Periodic ciphers with small blocks and cryptanalysis of KeeLoq,* Tatra Mt. Math. Publ. **41** (2008), 167–188.

[6] COURTOIS, N.: *Algebraic complexity reduction and cryptanalysis of GOST,* Preprint, 2010–2013, `http://eprint.iacr.org/2011/626`.

[7] COURTOIS, N.: *Security evaluation of GOST 28147-89 in view of international standardisation,* Cryptologia **36** (2012), 2–13.

[8] COURTOIS, N.: *Low complexity key recovery attacks on GOST block cipher,* Cryptologia **37** (2013), 1–10.

[9] COURTOIS, N.—MISZTAL, M.: *First differential attack on full 32-round GOST,* in: 13th Internat. Conf.—ICICS '11 (S. Qing et al., eds.), Beijing, China, 2011 Lecture Notes in Comput. Sci., Vol. 7043, Springer, Berlin, 2011, pp. 216–227.

[10] COURTOIS, N.: *An improved differential attack on full GOST,* Cryptology ePrint Archive, Report 2012/138, `http://eprint.iacr.org/2012/138`.

[11] SAMARATI, P.—MOUROUZIS, TH.: *Enhanced truncated differential cryptanalysis of GOST,* in: 10th Internat. Conf. on Security and Cryptography—SECRYPT '13 (P. Samarati, ed.), Reykjavik, Iceland, 2013, Lecture Notes in Comput. Sci., Vol. 7783, Springer, Berlin, 2013, pp. 411–418.

[12] COURTOIS, N. T.—MOUROUZIS, TH.: *Propagation of truncated differentials in GOST,* in: SECURWARE '13, The 17th Internat. Conf. on Emerging Security Inform., Systems and Technol., 2013, Barcelona, Spain (accepted).

[13] COURTOIS, N. T.—HULME, D.—MOUROUZIS, TH.: *Solving circuit optimisation problems in cryptography and cryptanalysis,* in: (informal) Proc. of SHARCS '12, Workshop, Washington, USA, pp. 179–191, `http://2012.sharcs.org/record.pdf`. An abridged version appears in the electronic proceedings of the 2nd IMA Conf. Mathematics in Defence 2011, UK.

[14] DINUR,I.—DUNKELMAN,O.—SHAMIR,A.: *Improved attacks on full GOST,* in: Fast Software Encryption—FSE '12, 19th Internat. Workshop, Washington, USA, 2012, Lecture Notes in Comput. Sci., Vol. 7549, Springer, Berlin, 2012, pp. 9–28, `http://eprint.iacr.org/2011/558/`.

[15] 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/`

[16] MENDEL, F.–PRAMSTALLER, N.–RECHBERGER, CH.–KONTAK, M.–SZMIDT, J.: *Cryptanalysis of the GOST hash function,* in: Advances in Cryptology–CRYPTO '08, 28th Annual Internat. Cryptology Conf. (D. Wagner, ed.), Santa Barbara, CA, USA, 2008, Lecture Notes in Comput. Sci., Vol. 5157, Springer, Berlin, 2008, pp. 162–178.

[17] ISOBE, T.: *A single-key attack on the full GOST block cipher,* in: Fast Software Encryption—FSE '11, 18th Internat. Workshop (A. Joux, ed.), Lyngby, Denmark, 2011, Lecture Notes in Comput. Sci., Vol. 6733, Springer, Berlin, 2011, pp. 290–305.

[18] KARA, O.: *Reflection cryptanalysis of some ciphers,* in: Progress in Cryptology–INDOCRYPT 08, 9th Internat. Conf. on Cryptology in India (R. Chowdhury et al., eds.), Kharagpur, India, 2008, Lecture Notes in Comput. Sci., Vol. 5365, Springer, Berlin, 2008, pp. 294–307.

[19] KARA, O.—KARAKOÇ, F.: *Fixed points of special type and cryptanalysis of full GOST,* in: The 11th Internat. Conf. on Cryptology and Network Security—CANS '12 (J. Pieprzyk et al., eds), Darmstadt, Germany, 2012, Lecture Notes in Comput. Sci., Vol. 7712, Springer, Berlin, 2012, pp. 86–97.

[20] POSCHMANN, A.—LING, S.—WANG, H.: *256 bit standardized crypto for 650 GE–GOST revisited,* in: 12th Internat. Workshop—CHES '10 (S. Mangard et al., eds.), Santa Barbara, USA, 2010, Lecture Notes in Comput. Sci., Vol. 6225, Springer, Berlin, 2010, pp. 219–233.

[21] Random Permutation Statistics—Wikipedia article, November 2012, `http://en.wikipedia.org/wiki/Random~permutation~statistics`.

[22] RUDSKOY, V.—DMUKH, A.: *Algebraic and differential cryptanalysis of GOST: fact or fiction,* in: Workshop on Current Trends in Cryptology—CTCrypt '12, affiliated with 7th Internat. Comput. Sci. Symposium in Russia (CSR '12), 2012, Nizhny Novgorod, Russia, 2012.

[23] SEKI, H.—KANEKO, T.: *Differential cryptanalysis of reduced rounds of GOST.* in: Selected Areas in Cryptography—SAC '00, (D. R. Stinson and S. E. Tavares, eds.), 7th Annual Internat. Workshop, 2000, Waterloo, Ontario, Canada, Lecture Notes in Comput. Sci., Vol. 2012, Springer, Berlin, 2000, pp. 315–323.

[24] SCHNEIER, B.: *Section 14.1 GOST (2nd ed.)*, in: Applied Cryptography, John Wiley and Sons, New York, 1996.

[25] ZABOTIN, I. A.—GLAZKOV, G. P.—ISAEVA, V. B.: *Cryptographic protection for information processing systems,* Government Standard of the USSR, GOST 28147-89, Government Committee of the USSR for Standards, 1989. (In Russian), translated to English in `ftp.funet.fi/pub/crypt/cryptography/papers/gost/russian-des-preface.ps.gz`

Received September 3, 2013

University College London
Gower Street
London WC1E 6BT
UNITED KINGDOM
E-mail: n.courtois@cs.ucl.ac.uk