Algebraic Complexity Reduction and Cryptanalysis of GOST

Nicolas T. Courtois

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

Abstract. GOST 28147-89 is a well-known block cipher and the official encryption standard of the Russian Federation. Its large key size of 256 bits at a particularly low implementation cost [29] make GOST a plausible alternative for AES-256 and 3-key triple DES. The latter for the same block size of 64 bits offers keys of only 168 bits. All these algorithms are widely used, in particular in the financial industry. GOST is implemented in OpenSSL and other crypto libraries [23, 40], and is increasingly popular also outside its country of origin [22, 29]. In 2010 GOST was submitted to ISO, to become an international standard.

In theory 256-bit keys could remain secure for 200 years. GOST was analysed by Schneier, Biham, Biryukov, Dunkelman, Wagner, various Australian, Japanese, and Russian scientists, and all researchers seemed to agree that it looks quite secure. Though the internal structure of GOST seems quite weak compared to DES, and in particular the diffusion is not quite as good, it is always stipulated that this should be compensated by a large number of 32 rounds cf. [20, 38, 37, 3] and by the additional non-linearity and diffusion provided by modular additions [20, 30]. At Crypto 2008 the hash function based on this cipher was broken. Yet as far as traditional encryption applications with single random keys are concerned, and until 2011, no cryptographically significant attack on this algorithm was found. At FSE 2011, a reflection attack with very large memory requirements was presented .

In this paper we present several attacks on full 32-rounds GOST two of which are substantially faster and most of which require much less memory. Our attacks belong to the family of conditional algebraic attacks on block ciphers [12, 11]: which can be defined as attacks in which the problem of key recovery is written as a problem of solving a large system of algebraic equations, and where the attacker makes some "clever" assumptions on the cipher which lead to an important simplification in the size and algebraic complexity of the problem, which makes it solvable in practice if the assumptions hold. Our methods work by black box reduction and allow to literally break the cipher apart into smaller pieces and reduce breaking GOST to a low data complexity software algebraic attack on only 8 rounds.

In this paper we show new non-trivial applications of self-similarity, and fixed points, and new attacks with simple and double reflection. We also demonstrate that GOST can also be broken totaly independently from reflection attacks. Overall we obtain six new attacks faster than brute force on the full 32-round GOST.

Key Words: Block ciphers, Feistel schemes, key scheduling, self-similarity, fixed points, reflection attacks, MITM, advanced slide attacks, algebraic attacks.

1 Background

GOST is a block cipher with a simple Feistel structure, 64-bit block size, 256-bit keys and 32 rounds. Each round contains a key addition modulo 2^{32}, a set of 8 bijective S-boxes on 4 bits, and a simple rotation by 11 positions. A particularity of GOST is that its S-boxes can be secret. One set of S-boxes has been published in 1994 as a part of the Russian standard hash function specification GOST R 34.11-94 and according to Schneier [38] this set is used by the Central Bank of the Russian Federation. This precise version of GOST 28147-89 block cipher is the most popular one, and it is commonly called just "the GOST cipher" in the cryptographic literature. There exists a Russian reference implementation of GOST which is a part of OpenSSL library and contains eight sets of S-boxes [23] which can be used for encryption or for the GOST Hash function. Overall, it is because of the small size of the GOST S-boxes (see [8, 10, 12, 5, 6]) that the cipher is vulnerable to "algebraic attacks" [5, 8, 12, 34, 35, 11] such as described in this paper, and therefore our attacks are expected to work with a similar complexity for any choice of S-boxes. In all our attacks we assume however that the S-boxes are known. Otherwise they could be recovered, see [36, 18].

It is widely known that the structure of GOST is in itself quite weak, an in particular the diffusion is quite poor, however, this is expected to be compensated by a large number of rounds [38]. Thus, so far there was no significant attack on this algorithm from the point of view of communications confidentiality: an attack which would allow decryption or key recovery in a realistic scenario where GOST is used for encryption with various random keys. In contrast, there are already many many papers on weak keys in GOST [27, 3], attacks for some well-chosen number of rounds [27, 1, 37], attacks with modular additions removed [3], related-key attacks [28, 17, 33], reverse engineering attacks on S-boxes [36, 18], and attacks on the hash function based on this cipher [25]. In all these attacks the attacker has much more freedom than we will allow ourselves.

In this paper we limit ourselves to the questions which pertain to the security of GOST used in encryption, with one single key chosen at random. Until 2011 no single key recovery attack on full-round GOST was proposed. A basic assessment of the security of GOST against linear and differential cryptanalysis has been conducted in 2000 by Gabidulin *et al*, see [20]. The results are quite impressive: at the prescribed security of level of 2^{256}, 5 rounds are sufficient to protect GOST against linear cryptanalysis. Moreover, even if the S-boxes are replaced by identity, and the only non-linear operation in the cipher is the addition modulo 2^{32}, the cipher is still secure against linear cryptanalysis after 6 rounds out of 32. Differential cryptanalysis of GOST seems comparatively easier and have attracted more attention. In [20] the authors also estimate that, 7 rounds should be sufficient to protect GOST against differential cryptanalysis. Moreover, two Japanese researchers [37], show that the straightforward classical differential attack with one single differential characteristic is unlikely to work **at all** for a large number of rounds. This is due to the fact that when we study reasonably "good" iterative differential characteristics for a limited number of rounds (which already propagate with probabilities not better than $2^{-11.4}$ per round, cf. [37]),

we realize that they only work for a fraction of keys smaller than half. For full 32-round GOST such an attack with a single characteristic would work only for a negligible fraction of keys of about 2^{-62} (and even for this tiny fraction if would propagate with a probability not better than 2^{-360}). The best advanced multiple differential attack proposed in [37] allows to break about 13 rounds of GOST. In 2011 better attacks of this type have been found, allowing finally to break full 32 round GOST faster than by brute force, see [14].

According to Biryukov and Wagner, the structure of GOST, and in particular the reversed order of keys in the last 8 rounds, makes it secure against sliding attacks [19, 2, 3]. However the cipher still has a lot of self-similarity and this exact inversion of keys allows other attacks in which fixed points are combined with a so called "Reflection" property [25, 27]. The latter attack allowed to break GOST only for certain keys, which are weak keys. Finally, at FSE 2011 a better reflection attack which allows to break arbitrary keys have been proposed [26]. In this paper we propose several new and better attacks with single reflection, double reflection. We also show one very good attack which does not use any reflections, better than another very recent attack from [13].

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

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

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

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

3 Algebraic Cryptanalysis and Low Data Complexity Attacks on Reduced-Round Block Ciphers

Algebraic attacks, on block and stream ciphers, can be defined as attacks in which the problem of key recovery is written as a problem of solving a large

system of Boolean algebraic equations which follows the geometry and structure of a particular cryptographic circuit [5, 6, 8, 10]. The main idea was explicitly proposed by Shannon in 1949, see [39]. For DES the idea was articulated as a method of Formal Coding [24]. The best currently known attack on DES can be found in [10]: it allows to break only 6 rounds of DES given only 1 known plaintext. The most efficient attacks nowadays are based on writing ciphers as systems of multivariate polynomial equations and manipulating these equations using either algebraic tools (elimination algorithms such as XL, Gröbner Bases [16] and ElimLin cf. [12]) or constraint satisfaction software such as SAT solvers which solve algebraic problems after conversion [9]. Many other methods have been proposed recently [34, 35] and for one problem instance many different attack techniques do usually work to some extent, see [10] and though SAT solvers do frequently solve many practical problems where Gröbner bases run out of memory, see [9], it was also shown in [9] that in a few cases where both methods worked, Gröbner bases methods were actually faster. We summarize all these methods which use "solver software" to determine unknown variables inside a complex circuit of Boolean equations under the general term of Algebraic Cryptanalysis (AC).

3.1 Application to GOST

GOST is a Feistel cipher with 32 rounds. In each round we have a round function $f_{k_i}(X)$ with a 32-bit sub-key k_i. Each round function contains a key addition modulo 2^{32}, a set of 8 bijective S-boxes on 4 bits, and a simple rotation by 11 positions. We need to to find a way to represent the cipher as an algebraic system of equations in such a way that it can efficiently be solved. It can be seen as encoding the problem of key recovery as an instance of an NP-hard problem. Both methods for encoding ciphers as such problems, and advanced heuristic algorithms for solving such problems are in constant evolution and are constantly improved. We have developed several efficient methods for formal encoding of GOST block cipher in the spirit of [10] and a lot of complex encoding, conversion and solver software for algebraic cryptanalysis. We have:

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

Justification: Our current best method for GOST is really exactly the same as the best known encoding method for DES described in [10]. We encode the S-boxes as an algebraic system of I/O relations (equations which relate Inputs and Outputs of these S-boxes), in a very similar way as for DES, see [10] for more details. Furthermore, in our fastest attacks, and also in the fastest attacks described in [10], we use about 20 additional variables per S-box, which allow equations be more more sparse. In order to encode the addition modulo 2^{32} we follow the first method described in [12]. The concatenation of all these equations describing the whole cipher or a large chunk of it is solved by various solver software, exactly

the same as in [10]. Given the fact that GOST has "weak diffusion" and that overall GOST is "not too complex" compared to any other block cipher (see [29]) we expect that to some extent our systems are solvable in practice. This is confirmed by our computer simulations. As an immediate corollary we get:

Fact 4 (Key Recovery for 8 Rounds and 2 KP). Given 2 P/C pairs for 8 rounds of GOST we can enumerate 2^{128} candidates for the full 256-bit key in time of about 2^{24} GOST encryptions each, and in total time equivalent to 2^{152} GOST encryptions on the same software platform. The memory requirements are again negligible.

Justification: We check 2^{128} keys (k_0, k_1, k_2, k_3), in which case we encrypt for 4 rounds twice, (time to do this can be neglected) and obtain two pairs for 4 rounds with key (k_4, k_5, k_6, k_7) in time of in 2^{24} GOST computations, cf. Fact 3.

By the same methods we also established that:

Fact 5 (Key Recovery for 8 Rounds and 3 KP). Given 3 P/C pairs for 8 rounds of GOST we can produce 2^{64} candidates for the 256-bit key in time equivalent to 2^{120} GOST encryptions. The storage requirements are negligible and all the 2^{64} candidates can be produced in an uniform way, each of them is produced in time of 2^{56} GOST encryptions on average.

Remark: this result is particularly significant because it is close to 2^{128} which one could obtain in a Meet-In-the-Middle (MIM) attack, cf. Fact 7, and requires negligible storage.

Important Remark: The three results above are experimental facts. The main object of this paper is **NOT how to** achieve and further improve this type of software attacks, cf. [8, 10], but **how** can the complexity of GOST be **reduced in the "black box" way**, so that we can ever hope to be able to apply results such as Fact 5.

4 High-level Description of GOST and Key Observations

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

One 32-bit sub-key is used in each round, and their exact order is as follows:

rounds	1 8	9 16	17 24	25 32
keys	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$\mathbf{k_0}k_1k_2k_3k_4k_5k_6k_7$	$k_7k_6k_5k_4k_3k_2k_1\mathbf{k_0}$

Table 1. Key schedule in GOST

We write GOST as the following functional decomposition (to be read from right to left) which is the same as used at Indocrypt 2008 [27]:

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

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

4.1 The Internal Reflection Property of GOST

We start with the following observation which was already used in weak attacks on GOST from [27] and to cryptanalyse the GOST hash function Crypto 2008 [25]. In both cases the attacker had the freedom to choose the key, which made the attacks work. This property is exploited in most (but not all) of our attacks and in [26], and for arbitrary keys.

Fact 6 (Internal Reflection Property). Consider the last 16 rounds of GOST $\mathcal{D} \circ \mathcal{S} \circ \mathcal{E}$. for one fixed GOST key. This function has an exceptionally large number of fixed points: applied to X gives the same value X with probability 2^{-32} over the choice of X, instead of 2^{-64} for a random permutation.

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

Remark: with 2^{32} fixed points, one can expect that one of them will be symmetric, this is exploited in our fastest attack on GOST, see Fact 12.

5 Our Reductions: Methodology and Key Steps

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

First a certain number of assumptions on internal variables of the cipher, for one or several encryptions, are made. The probability that these assumptions hold for a random GOST key, needs to be evaluated. Then the probabilities that, when our assumptions hold, certain well chosen variables in the encryption circuit(s) can be guessed by the attacker, will be estimated. Finally the combination of the assumptions and the guessed values will allow the attacker to obtain a small number of 2,3 or 4 P/C pairs for 8 rounds of the cipher.

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

5.1 Synthetic Summary of All Our Reductions

The following Table 2 on p. 8 gives a synthetic summary of all key steps in the main attacks described in this paper.

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

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

Let X_i, Y_i be the set of known plaintexts with $i = 1, 2, 3, \ldots, M$, where $M \approx 2^{32}$. Most attacks in this paper require that the Internal Reflection Property (Fact 6) holds for (at least) one i. This requires at least 2^{32} known plaintexts on average, which means that for some keys we may need a bit less, and for some keys a bit more with $M > 2^{32}$, but rarely much more.

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

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

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

Justification: This is done as follows:

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

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

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

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

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

Fact 7. Given 2 P/C pairs for 8 rounds of GOST, and a few more additional P/C pairs for full 32-rounds of GOST for verification, the correct full GOST key on 256 bits can be determined in time of $1.25 \cdot 2^{128}$ GOST encryptions, and with 2^{132} bytes of memory.

Justification: Due to space limitations a detailed justification was moved to the Appendix. We combine this fact with Reduction 1, and we obtain immediately:

Reduction Summary					
Reduction cf.	Red. 1 §6.1	Red. 2 §7	Red. 3 §8	Red. 4 §8.1	Red 5 §9
Type	1x Internal Reflection		2x Reflection		Fixed Point
From (data 32 R)	2^{32} KP		2^{64} KP		
Obtained (for 8R)	**2** KP	**3** KP	**3** KP	**4** KP	**2** KP
Valid w. prob.	2^{-96}	2^{-128}	2^{-96}	2^{-128}	2^{-64}

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

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

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

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

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

Last step	MIM	Guess + Algebraic			
Cases $\in$ Inside	2^{128}	2^{128}	2^{64}		2^{128}
Then Fact cf.	Fact 8	Fact 4	Fact 5		Fact 4
Time to break 8R	2^{128}	2^{152}	2^{120}		2^{152}
Storage bytes	2^{132}	-	-	2^{67}	-
♯ false positives	2^{224}		2^{192}	2^{128}	2^{192}
Attack time 32 R	2^{224}	2^{248}	2^{248}	$\mathbf{2^{216}}$	2^{248}	$\mathbf{2^{216}}$

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

Fact 8. Given an average number of 2^{32} random known plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $1.25 \cdot 2^{96+128}$ GOST encryptions, which is $2^{30.7}$ times faster than brute force and with about 2^{132} bytes of memory.

Summary and Discussion. This attack requires 2^{32} known plaintexts, and the running time is $2^{224.3}$ GOST encryptions, a slight improvement compared to 2^{225} obtained in [26]. **It is incorrect to claim that both attacks are the same.** They are very different. Our attack is much simpler and slightly faster. In their attack they guess values after 4 and 12 rounds and do a MITM attack on 4+4 rounds, and use an equivalent key technique. Our attack is much simpler, we guess only one value after 8 rounds and do a MITM attack on 8+8 double rounds in parallel.

Storage Requirements: they are very important: about 2^{132} bytes of fast memory, which need basically to work at the speed of encryption with only 4 rounds of the cipher. If we consider that today the memory of 2^{30} has a cost comparable to 2^{60} computations, it is possible to believe that the cost of 2^{132} bytes of memory at some moment in the future may be similar to 2^{255} in computing power. In this case, it is possible to believe that we do **not yet** have a valid attack on GOST. In this paper we present several more convincing attacks.

6.3 A Reflection-MIM-Algebraic Attack

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

Fact 9. Given an average number of 2^{32} random known plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time 2^{96+152} GOST encryptions, which is 2^7 times faster than brute force. The memory required is negligible.

Justification: This is again straightforward:

1. Again given 2^{32} KP X_i, Y_i, we use the Reduction 1, and on average there is one index i such that $\mathcal{E}^3(X_i)$ is symmetric. Then a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ can be guessed and the guess will be correct with probability 2^{-96}.
2. In each of 2^{96} cases we apply Fact 4 which allows to enumerate 2^{128} keys in time of 2^{24} GOST computations each. The total time spent in this step is $2^{96+128+24}$ GOST computations.
3. Overall in this attack we will check 2^{96+128} full keys on 256-bits, most of them being false positives. Each is checked with on average one and at most a few additional P/C pairs X_i, Y_i. Total time spent in this step can be neglected.

Summary. This attack requires 2^{32} known plaintexts, and the running time is $2^{96+128+24}$ GOST encryptions, which is 2^7 times faster than brute force. The storage requirements are negligible.

This attack, though slower, is arguably much better than the previous one and the one of [26], which required an inacceptable amount of storage. In what follows we are going to describe better attacks, with lower complexity, and still with negligible storage.

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

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

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

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

$$B = \mathcal{E}(A) \tag{3}$$
$$C = \mathcal{E}^2(A) \tag{4}$$
$$D = \mathcal{E}^3(A) \tag{5}$$
$$D = \mathcal{S}(\mathcal{E}^3(A)) \tag{6}$$
$$C = \mathcal{D}(D) = Enc_k(A) \tag{7}$$

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

Fact 10. Given an average number of 2^{32} random knowns plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $2^{128+120}$ GOST encryptions, which is 2^7 times faster than brute force. The memory required is negligible.

1. We use the Reduction 2 and given 2^{32} KP we obtain a 5-tuple i, A, B, C, D and correct with probability 2^{-96} and 3 P/C pairs for 8 rounds of GOST: $B = \mathcal{E}(E)$ from (3), $C = \mathcal{E}(B)$ from (4), and $D = \mathcal{E}(C)$ from (5). These 8 rounds of GOST depend however on the full 256-bit key. And these 3 P/C pairs do not uniquely determine the key. Moreover, only 64 bits of information about the key are available from one single value i and the information contained in the 3 P/C pairs for 8 rounds of GOST above is largely based on attacker's guesses, and will only be confirmed after a large number of candidates for the full 256-bit GOST key will be generated, and checked against some 4 additional P/C pairs X_j, Y_j for $j \neq i$, see Fact 1.
2. Following Fact 5 (cf. page 5) in each of 2^{128} cases tried and on average, and in total time equivalent to 2^{120} GOST encryptions we obtain 2^{64} candidates for the GOST key k.
3. For each of the 2^{128} cases i, A, B, C, D we get from the program of Fact 5 a uniform enumeration of 2^{64} keys. We get an enumeration of 2^{192} 6-tuples i, A, B, C, D, k. Where k is a candidate for the full 256-bit key. These 6-tuples

contain about 2^{192} different candidates for the GOST key k. Each 6-tuple is generated in time of 2^{56} GOST encryptions on average (cf. Fact 5).
These 6-tuples are checked with 4 extra additional P/C pairs, for example the previous ones X_{i-1}, Y_{i-1} etc. With 5 P/C pairs total, only the right key will be accepted, and the probability that a wrong key is accepted in our attack is 2^{-64}, see Fact 1.

4. Thus we reject all the 2^{192} 6-tuples i, A, B, C, D, k except the correct one which contains the full 256-bit key of the cipher.

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

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

Now we are going now to describe a better attack where we are still going to reduce the security of GOST to the problem of breaking 8 rounds of GOST with 3 known P/C pairs where our guess will be valid with a higher probability. This however will be obtained at a price of 2^{64} known plaintexts (instead of 2^{32}KP). This larger quantity of data is required if order to find cases where the internal reflection (cf. Fact 6) occurs twice, in order to be able to analyse several encryptions at the same time, which will reduce the number of false positives.

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

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

Fact 11 (Key Property). There is on average one value A which satisfies Assumption 1 above. For 63% of all GOST keys at least one such A exists.
Justification: We have 2^{64} possibilities, each time the probability is 2^{-64}. Such a value A exists for $1-(1-1/N)^N \approx 63\%$ of all GOST keys where $N = 2^{64}$.
Remark: For 37 % of keys this attack fails but our earlier attacks requiring only 2^{32} KP still work.

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

Justification: will be provided below.

1. Let X_i, Y_i be the set of all the 2^{64} known plaintexts.
2. On average there exists one index such that both $C = \mathcal{E}^2(X_i)$ $D = \mathcal{E}^3(X_i)$ are symmetric values on 64 bits. Then since $D = \mathcal{E}^3(A)$ is symmetric we have
$$Enc_k(A) = C = \mathcal{E}^2(A) \tag{8}$$
So far we don't know neither i nor A, C, D. However since from our Key Assumption on i the value of $C = \mathcal{E}^2(X_i)$ must be a symmetric value on 64-bits, we can limit ourselves to select C among all symmetric ciphertexts, guess $C = Y_i$ and our guess is true with probability 2^{-32}. Let $A = X_i$ be the corresponding plaintext. We have a triple i, A, C which is correct with probability 2^{-32}.

3. Then we guess B and get a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ and our guess will be correct with probability 2^{-96}.
   As in our first two attacks we don't try to guess D.
4. This gives exactly 2 P/C pairs for 8 rounds $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$.
5. One extra pair will be obtained by decrypting B as follows. We define Z as $Z = Dec_k(B) = \mathcal{E}^{-3}(\mathcal{S}(\mathcal{E}(B)))$. We have

$$Z = Dec_k(B) = \mathcal{E}^{-3}(\mathcal{S}(C)) = \mathcal{E}^{-3}(C) = \mathcal{E}^{-2}(B) = \mathcal{E}^{-1}(A) = \mathcal{D}(A).$$

   Where we used our assumption that $C = \mathcal{E}^2(A)$ is symmetric, and we get that $Z = Dec_k(B) = \mathcal{D}(A)$. Thus we get our 3-rd pair $A = \mathcal{E}(Z)$. This decryption is done in constant time if we assume that all the pairs X_i, Y_i are stored using a hash table.
6. Thus we determine i, Z, A, B, C and we get 3 known P/C pairs for 8 rounds of GOST, and our guess is valid with probability 2^{-96}.

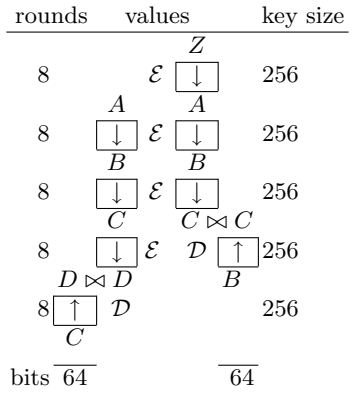

Fig. 1. Our best attack on GOST

Thus we obtain the following result:

Fact 12. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{216} GOST encryptions. The storage required is 2^{64} times 8 bytes.

Justification: As above we get 3 known P/C pairs for 8 rounds of GOST, and our guess is valid with probability 2^{-96}. For each of the 2^{96} cases i, A, B, C we get from the program of Fact 5 a uniform enumeration of 2^{64} keys. Thus we get an enumeration of 2^{160} 5-tuples i, A, B, C, k. Where k is a candidate for the full 256-bit key. These 5-tuples contain about 2^{160} different candidates for the GOST key k. Each 5-tuple i, A, B, C, k is generated in time of 2^{56} GOST encryptions on average (cf. Fact 5). These 4-tuples are checked with 4 extra additional P/C pairs. We reject all the 2^{160} 4-tuples i, D, B, k except the correct one. Total cost is about 2^{160+56} GOST encryptions.

8.1 Alternative Attacks with Reduction to 4 Pairs

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

Reduction 4. [From 2^{64} KP for 32 Rounds to 4 KP for 8 Rounds]
Given 2^{64} known plaintexts for GOST, it is possible to obtain four P/C pairs for 8 rounds of GOST and our guess will be correct with probability 2^{-128}.

In Appendix we present three other and different methods to obtain 4 pairs given 2^{64} KP with the same and even slightly better success probability.

Unhappily, it appears that these alternative reductions currently do not lead to attacks which are better than those described in this paper. This is due to the fact, that currently we are not able to improve the timing of Fact 5, if we dispose of 4 P/C pairs for 8 rounds. Currently, we have no result better than Fact 5 and the necessity to guess D makes the attack simply 2^{32} times slower. This attack is summarized in the next to next to last column in Table 2.

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

Until now and in [26] all single-key attacks on GOST are based on reflections [26, 27] and in our best attack on GOST a reflection occurs twice. However there is another very simple attack on GOST with a similar complexity of about 2^{216} which does not use any reflection and where no symmetric 64-bit values appear. This shows that GOST is broken independently of reflection attacks [26, 27]

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

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

Fact 13. Given 2^{64} known plaintexts, it is possible to determine the full 256-bit key of GOST cipher in time of 2^{216} GOST encryptions. The storage required is 2^{64} times 8 bytes.

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

Important Remark This attack will work for about 63 % of all GOST keys for which $\mathcal{E}$ has a fixed point. For the remaining 37 % of Family B keys this attack fails, but the attack described in Section 8 is expected to still work with roughly the same complexity. Both results should be improved in the near future, and the exact best complexities obtained do not need to be exactly equal. This provides strong motivation for doing more research also on the software solver aspects of algebraic attacks which are now shown to be instrumental in breaking important real-life ciphers (also in [11]).

10 Conclusion

The Russian encryption standard GOST is very widely used and it has a substantially lower implementation cost than any comparable cipher, see [29]. Accordingly, in 2010 GOST was submitted to ISO to become an international encryption standard. Just about 7 rounds out of 32 are needed against linear and differential cryptanalysis, [20], cf. also [37]. Until 2011, no single-key attack on GOST faster than brute force was found.

The general idea of Algebraic Cryptanalysis has been around for more than 60 years [39, 24]. Yet only in the last 10 years several efficient software tools for solving various NP-hard problems involved have been developed, while numerous specific vulnerabilities leading to efficient attacks of this type have been found. A number of stream ciphers are indeed broken [8, 6, 7]. For block ciphers, algebraic cryptanalysis is generally known to be able to break any block cipher... if the number of rounds is small enough see [10]. The main contribution here is to be able to reduce the security of a cipher with 32 rounds to the security of the cipher with 8 rounds. In 2008 an important industrial cipher KeeLoq was in this way broken by an algebraic attack [11]. In this paper we break another important real-life block cipher.

All these attacks have a lot in common. We call it Algebraic Complexity Reduction. It applies to ciphers, which have "a lot of" self-similarity. In order to achieve our complexity reduction we need to solve a certain combinatorial puzzle. With well-chosen assumptions on internal values, and other values determined, we are able to literally break the cipher apart into smaller pieces. This greatly reduces the complexity of the cipher as a circuit and makes algebraic cryptanalysis suddenly feasible. We proposed five non-trivial black-box reductions of this type on GOST, and main results are summarized in Table 2 on page 8. We considerably enlarge the spectrum of self-similarity attacks on block ciphers: our single and double reflection attacks are able to exploit similarities of individual sub-blocks and their inverses, and are now quite far away from any known variant of slide attack (cf. [19, 2, 3, 1, 18]), or a known fixed point attack (cf. [11, 27]). We also exhibit (another one in [13] and two more in the extended version of this paper) different single-key attacks on GOST which don't use any reflections [26, 27] whatsoever, which proves that we are able to break GOST independently from previous works [26, 27].

Our two fastest attacks on GOST require 2^{64} known plaintexts and are 2^{39} times faster than brute force. We also presented three attacks requiring only 2^{32} known plaintexts and still faster than brute force. Both very recently published attacks on GOST in [26, 13] can be seen as the starting point on which we improve. Our best attacks on GOST are particularly interesting because they are significantly better than meet-in-the middle attacks [26] on both time and memory complexity.

Algebraic description of S-boxes is quite similar for any S-box and all our attacks are expected to work with a similar complexity for any choice of GOST S-boxes. We conclude that ISO should not standardize GOST, as this algorithm is deeply flawed and does not provide the security level required by ISO.

References

1. Eli Biham, Orr Dunkelman, Nathan Keller: *Improved Slide Attacks,* In FSE 2007, LNCS 4593 Springer 2007, pp. 153-166.
2. A. Biryukov, D.Wagner: *Slide Attacks,* In proceedings of FSE'99, LNCS 1636, pp. 245-259, Springer, 1999.
3. Alex Biryukov, David Wagner: *Advanced Slide Attacks,* In Eurocrypt 2000, LNCS 1807, pp. 589-606, Springer 2000.
4. Nicolas Courtois, Michał Misztal: *Aggregated Differentials and Cryptanalysis of PP-1 and GOST,* To appear in 11th Central European Conference on Cryptology, will be held in Debrecen, Hungary, on June 30 - July 2, 2011.
5. Nicolas Courtois and Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations,* Asiacrypt 2002, LNCS 2501, pp.267-287, Springer.
6. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, LNCS 2656, pp. 345-359, Springer. An extended version is available at `http://www.minrank.org/toyolili.pdf`
7. Nicolas Courtois: *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback,* Crypto 2003, LNCS 2729, pp: 177-194, Springer.
8. Nicolas Courtois: *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers,* in AES 4, LNCS 3373, pp. 67-83, Springer, 2005.
9. Gregory V. Bard, Nicolas T. Courtois and Chris Jefferson: *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers,* `http://eprint.iacr.org/2007/024/`.
10. Nicolas Courtois, Gregory V. Bard: *Algebraic Cryptanalysis of the Data Encryption Standard,* In Cryptography and Coding, 11-th IMA Conference, pp. 152-169, LNCS 4887, Springer, 2007. Preprint available at `eprint.iacr.org/2006/402/`.
11. Nicolas Courtois, Gregory V. Bard, David Wagner: *Algebraic and Slide Attacks on KeeLoq,* In FSE 2008, pp. 97-115, LNCS 5086, Springer, 2008.
12. Nicolas Courtois and Blandine Debraize: *Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0.,* In ICICS 2008, 10th International Conference on Information and Communications Security, 20 - 22 October, 2008, Birmingham, UK. In LNCS 5308, pp. 328-344, Springer, 2008.
13. Nicolas Courtois: *Security Evaluation of GOST 28147-89 In View Of International Standardisation,* document officially submitted to ISO in May 2011, In Cryptology ePrint Archive, Report 2011/211. 1 May 2011, `http://eprint.iacr.org/2011/211/`.
14. Nicolas Courtois, Michał Misztal: *Differential Cryptanalysis of GOST,* In Cryptology ePrint Archive, Report 2011/312. 14 June 2011, `http://eprint.iacr.org/2011/312`.
15. Charles Bouilleguet, Patrick Derbez, Orr Dunkelman, Nathan Keller, Pierre-Alain Fouque: *Low Data Complexity Attacks on AES,* Cryptology ePrint Archive, Report 2010/633. `http://eprint.iacr.org/2010/633/`.
16. Jean-Charles Faugère: *A new efficient algorithm for computing Gröbner bases without reduction to zero (F5),* Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002, ACM Press.
17. Fleischmann Ewan, Gorski Michael, Huehne Jan-Hendrik, Lucks Stefan: *Key recovery attack on full GOST block cipher with zero time and memory,* Published as ISO/IEC JTC 1/SC 27 N8229. 2009.

18. Soichi Furuya: *Slide Attacks with a Known-Plaintext Cryptanalysis,* In ICISC 2001, LNCS 2288, 2002, pp. 11-50.
19. E. K. Grossman, B. Tuckerman: *Analysis of a Weakened Feistel-like Cipher,* 1978 International Conference on Communications, pp.46.3.1-46.3.5, Alger Press Limited, 1978.
20. Vitaly V. Shorin, Vadim V. Jelezniakov and Ernst M. Gabidulin: *Linear and Differential Cryptanalysis of Russian GOST,* Preprint submitted to Elsevier Preprint, 4 April 2001
21. I. A. Zabotin, G. P. Glazkov, V. B. Isaeva: *Cryptographic Protection for Information Processing Systems,* Government Standard of the USSR, GOST 28147-89, Government Committee of the USSR for Standards, 1989. In Russian, translated to English in `http://www.autochthonous.org/crypto/gosthash.tar.gz`
22. Vasily Dolmatov, Editor, RFC 5830: *GOST 28147-89 encryption, decryption and MAC algorithms,* IETF. ISSN: 2070-1721. March 2010. `http://tools.ietf.org/html/rfc5830`
23. 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/`
24. J. Hulsbosch: *Analyse van de zwakheden van het DES-algoritme door middel van formele codering,* Master thesis, K. U. Leuven, Belgium, 1982.
25. Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak and Janusz Szmidt: *Cryptanalysis of the GOST Hash Function,* In Crypto 2008, LNCS 5157, pp. 162 - 178, Springer, 2008.
26. Takanori Isobe: *A Single-Key Attack on the Full GOST Block Cipher,* In FSE 2011, Fast Software Ecnryption, Springer LNCS, 2011.
27. Orhun Kara: *Reflection Cryptanalysis of Some Ciphers,* In Indocrypt 2008, LNCS 5365, pp. 294-307, 2008.
28. John Kelsey, Bruce Schneier, David Wagner: *Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES,* In Crypto'96, LLNCS 1109, Springer, 1996.
29. 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.
30. C. Charnes, L. O'Connor, J. Pieprzyk, R. Savafi-Naini, Y. Zheng: *Comments on Soviet encryption algorithm,* In Advances in Cryptology - Eurocrypt'94 Proceedings, LNCS 950, A. De Santis, ed., pp. 433-438, Springer, 1995.
31. Random Permutation Statistics – wikipedia article, 22 January 2008, available at `http://en.wikipedia.org/wiki/Random~permutation~statistics`.
32. J.-J. Quisquater and Y. Desmedt and M. Davio: *The Importance of 'good' Key Scheduling Schemes (How to make a secure DES scheme with* $\leq$ *48 bit keys?,* In Crypto'85, LNCS 218, pp. 537–542, Springer, 1985.
33. Vladimir Rudskoy: *On zero practical significance of Key recovery attack on full GOST block cipher with zero time and memory,*
34. Igor Semaev: *Sparse Algebraic Equations over Finite Fields,* SIAM J. Comput. 39(2): 388-409 (2009).
35. Haavard Raddum and Igor Semaev: New Technique for Solving Sparse Equation Systems, ECRYPT STVL website, January 16th 2006, available also at `eprint.iacr.org/2006/475/`
36. Markku-Juhani Saarinen: *A chosen key attack against the secret S-boxes of GOST,* unpublished manuscript, 1998.

37. Haruki Seki and Toshinobu Kaneko: *Differential Cryptanalysis of Reduced Rounds of GOST. In SAC 2000, Selected Areas in Cryptography, Douglas R. Stinson and Stafford E. Tavares, editors, LNCS 2012, pp. 315323, Springer, 2000.*
38. *Bruce Schneier: Section 14.1 GOST,* in *Applied Cryptography,* Second Edition, John Wiley and Sons, 1996. ISBN 0-471-11709-9.
39. Claude Elwood Shannon: *Communication theory of secrecy systems,* Bell System Technical Journal 28 (1949), see in particular page 704.
40. Wei Dai: Crypto++, a public domain library containing a reference C++ implementation of GOST and test vectors, `http://www.cryptopp.com`

A Additional Explanations for The Reflection-Meet-In-The-Middle Attack with Memory

We recall our claim and justify it fully here:

Fact 7: Given 2 P/C pairs for 8 rounds of GOST, and a few more additional P/C pairs for full 32-rounds of GOST for verification, the correct full GOST key on 256 bits can be determined in time of $1.25 \cdot 2^{128}$ GOST encryptions, and with 2^{132} bytes of memory.

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

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

Fact 8 Given an average number of 2^{32} random known plaintexts for the full 256-bit GOST cipher, it is possible to determine the secret key in time $1.25 \cdot 2^{96+128}$ GOST encryptions, which is $2^{30.7}$ times faster than brute force and with about 2^{132} bytes of memory.

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

1. Let X_i, Y_i be the set of 2^{32} known plaintexts.
2. As in Reduction 1, on average there is one index i such that $\mathcal{E}^3(X_i)$ is symmetric. Then a 4-tuple i, A, B, C with $A = X_i$, $B = \mathcal{E}(A)$ and $C = \mathcal{E}(B)$ can be guessed and the guess will be correct with probability 2^{-96}.
3. In each of 2^{96} cases we apply our MIM attack from Fact 7. Thus we check 2^{96+128} cases and need to perform an equivalent of $1 + 8/32$ full encryptions per case, where the cost of each pre-computation of 2^{128} cases is amortized over each interval containing 2^{128} cases. In each of 2^{96} cases i, B we we obtain exactly one key, which is checked with on average one and at most a few additional P/C pairs X_i, Y_i. Overall only one correct key is obtained in this attack.

Remark: Again, this Reflection-MITM proposed here is different, simpler and very slightly faster than than one proposed in [26].