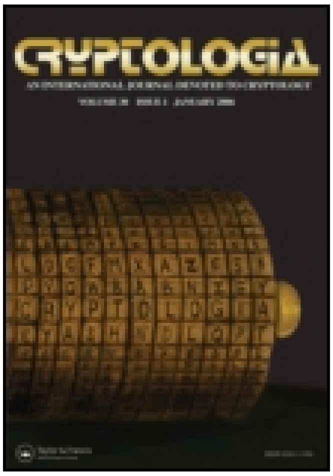

Cryptologia
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucry20

# Low-Complexity Key Recovery Attacks on GOST Block Cipher

Nicolas T. Courtois

Published online: 11 Jan 2013.

To cite this article: Nicolas T. Courtois (2013) Low-Complexity Key Recovery Attacks on GOST Block Cipher, Cryptologia, 37:1, 1-10, DOI: 10.1080/01611194.2012.739587

To link to this article: http://dx.doi.org/10.1080/01611194.2012.739587

*Cryptologia*, 37:1–10, 2013
Published with license by Taylor & Francis
ISSN: 0161-1194 print
DOI: 10.1080/01611194.2012.739587

# Low-Complexity Key Recovery Attacks on GOST Block Cipher

NICOLAS T. COURTOIS

**Abstract** GOST is a well-known Russian government block cipher. Until 2010, there was no attack on GOST used in encryption, cf. [9]. More recently, quite a few distinct key recovery attacks on full GOST have been found: [1–4, 6, 7]. Most of these attacks work by so-called "complexity reduction" [1]; they reduce the problem of breaking the full 32-round GOST to an attack with 2,3,4 KP for 8 rounds of GOST. In this article, we develop an alternative last step for these attacks. We present a new meet-in-the-middle attack for eight rounds, which is faster than any previous attack. Then we present a guess-then-determine attack with software using an SAT solver, which, for the same running time, requires much less memory. As a result we are able to improve by a factor of up to $2^{26}$ various attacks from [1, 3].

**Keywords** black-box reductions, block ciphers, key scheduling, low-data complexity attacks, meet in the middle, Russian GOST cipher, SAT solvers, self-similarity

## 1. The GOST Cipher

GOST [11] 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.

Each round of GOST looks exactly the same except for the key $k$ used:

$$(L, R) \to (R, L \oplus f_k(R)).$$

### 1.1. Key Schedule in GOST

The key structural property of GOST that makes it suitable for many cryptanalytic attacks is its key scheduling (Table 1), [1–3, 7, 8, 11].

Only 32 bits of the whole key, a fairly small proportion, is used in one round. In this article, we will concentrate on attacks on eight rounds of GOST, and for eight rounds of GOST, the key bits are never used twice.

Address correspondence to Nicolas T. Courtois, Computer Science, Univeristy College London, Gower Street, London WC1E 6BT, UK. E-mail: n.courtois@cs.ucl.ac.uk

Table 1. Key schedule in GOST

| | | | | |
|---|---|---|---|---|
| Rounds | 1 | 8 | 9 | 16 |
| Keys | $k_0k_1k_2k_3k_4k_5k_6k_7$ | | $k_0k_1k_2k_3k_4k_5k_6k_7$ | |
| Rounds | 17 | 24 | 25 | 32 |
| Keys | $k_0k_1k_2k_3k_4k_5k_6k_7$ | | $k_7k_6k_5k_4k_3k_2k_1k_0$ | |

## *1.2. Internal Connections*

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

GOST has 32 identical rounds, such as one described in Figure 1. The rounds differ only by the subsets of 32 key bits that they use.

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

Given the key schedule in Figure 1, we have a complete description of GOST.

At the left margin of Figure 1, we also show S-box numbers in the next round, which is very helpful, to see which bits are successfully determined in our attacks on GOST. A more detailed explanation of how these bits depend in the next round on the bits in the previous round will be developed below (Figure 2).

## 2. Predicting Carry Bits in GOST

In this section, we show certain basic facts studied in more detail in [2] concerning the prediction of carry bits inside GOST, which we need in this article.

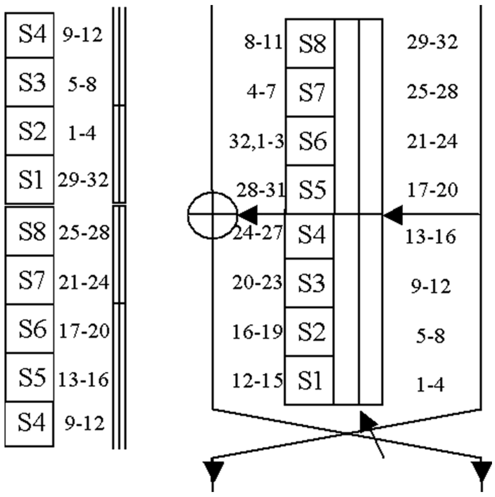

Figure 1. One round of GOST and connections in the following round.

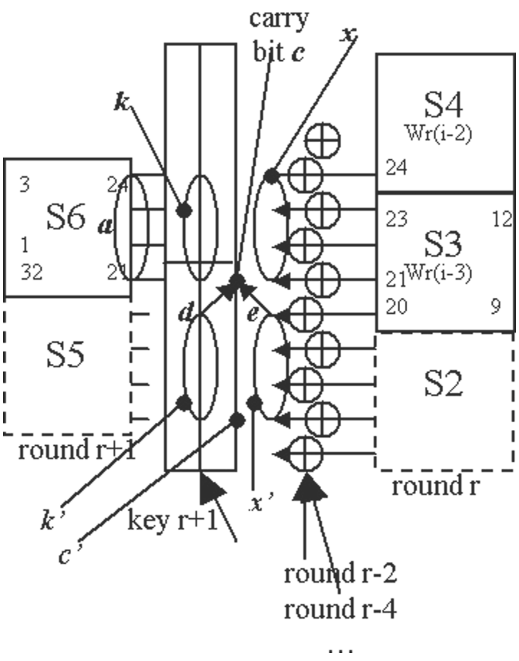

**Figure 2.** Computation of the input of one S-box with a carry bit.

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

We have the following basic fact.

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

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

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

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

Similarly, if we know the whole $x'$, there are only $1.25 = 2^{0.3}$ possibilities.

## 3. A Preliminary Question

We consider the following preliminary question.

We assume that we know the 60 key bits in the first three rounds shown in Figure 3: all key bits at S-boxes S1234567 in R1 (in round 1); All key bits at S6781 in R2, plus just one highest ranking bit of S5 in R2, which we denote by a special notation $S^1 5$; all key bits at S123 at R3; and just three lower ranking bits of S4 in R3, which we denote by $S_3 4$. To simplify the notations, we call $k_{123}^{(60)}$ this specific set of 60 key bits taken out of the 96 bits of $k_0$, $k_1$, $k_2$ in R1-3.

We define $U_{12-27}^{(i)}$ as the 16 state bits created after XORing with the outputs of exactly S1234 after R3. We want to be able to compute these 16 bits or produce a short list of possible values on 16 bits. Then it is easy to see the following fact.

**Fact 2.** Given the 60 key bits $k_{123}^{(60)}$ and the plaintext for one sigle encryption, there are on average about $2^{1.3}$ possibilities to determine the 16 bits $U_{12-27}^{(i)}$ after round 3.

*Justification.* The data bits 17-4 for S-boxes S5-8,1 entering R2 can be computed with probability 1, and we know the highest ranking bit of S5 in R1. In terms of Fact 1 (Figure 2), we are in the case where $x'$ and $d$ are known and we can enumerate 1, sometimes 2, but an average number of only $2^{0.3}$ of cases for the 4-bit state of S6 in R2. In each of these $2^{0.3}$ cases, the further carries and whole state of S781 in R2 are known with certainty.

This gives only $2^{0.3}$ possibilities for the 15 bits 1–15, which enter S1234 in the next round R3, and there is no carry entering S1. With bit 16 being unknown, we get $2^{1.3}$ possibilities for the joint state of S1234 in R3. Overall, we obtain roughly about only $2^{1.3}$ possibilities for $U_{12-27}^{(i)}$ in one single encryption.

## 4. A Meet-in-the-Middle (MITM) Attack on 8 Rounds of GOST

Our attack has three stages, which are shown in Figure 4. We present simultaneously two versions of the attack: with $D=3/D=4$ KP, respectively. We have 3·16/4·16 bits in the middle which gives respectively 48/64 bits.

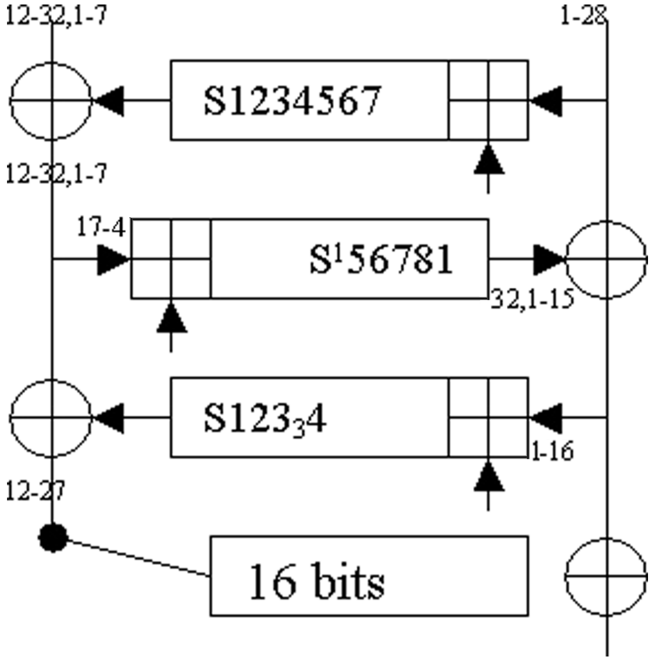

**Figure 3.** A preliminary question with 60 key bits and 16 output bits.

### 4.1. *Stage 1: An MITM Phase for Seven Rounds*

We consider the situation as on the left-most pane of Figure 4, which is perfectly symmetric with 60 key bits $k_{123}^{(60)}$ in the first three rounds, 16 bits in the middle $U_{12\text{-}27}^{(i)}$, and another 60 key bits $k_{567}^{(60)}$ in the last three rounds. We proceed as follows:

1. We apply Fact 2 three or four times for each encryption. Given $k_{123}^{(60)}$ and the three plaintexts, we can enumerate, on average, roughly but not exactly, about $2^{60+D\cdot(1.3)} \approx 2^{63.9}/2^{65.2}$ possibilities for the 48/64 bits in the middle in the two cases, $D = 3/4$, respectively.
2. This is about $2^{63.9\text{-}48} = 2^{15.9}$ or $2^{65.2\text{-}64} = 2^{1.2}$ keys per each middle value on 48/64 bits. We store this data in a hash table sorted by these 48/64 middle bits.
3. The same sort of table is computed for rounds 567 backwards.
4. This computation requires a maximum of twice $2^{65}/32$ GOST computations, as that most of the time we just need to compute about one round of GOST (other computations are amortized and require less key bits). Overall we need about $2^{61}$ GOST computations and about $2^{68}$ bytes of memory for both tables.
5. We fix 48/64 bits in the middle.
6. Now for 3 KP, in time of about $2^{48+15.9+15.9} = 2^{79.8}$ CPU clocks, which is only about $2^{71}$ GOST computations, and with memory of about $2^{68}$ bytes, we can enumerate also $2^{79.8}$ possible keys on 120 bits together with the middle values $U_{12\text{-}27}^{(1\text{-}3)}$. This enumeration is very fast because for each value in the middle, we can freely combine arbitrary pairs from two lists.
7. For 4 KP, in time of about $2^{64+1.2+1.2} = 2^{66.4}$ CPU clocks, which is only about $2^{57}$ GOST computations, and with memory of about $2^{69}$ bytes, we can enumerate also $2^{66.4}$ possible keys on 120 bits together with the middle values $U_{12\text{-}27}^{(1\text{-}4)}$.

### 4.2. *Stage 2: Extension with More Key Bits*

We consider 32 more key bits, as in the middle pane of Figure 4. We want to avoid guessing these key bits in one go, which would increase time complexity by $2^{32}$.

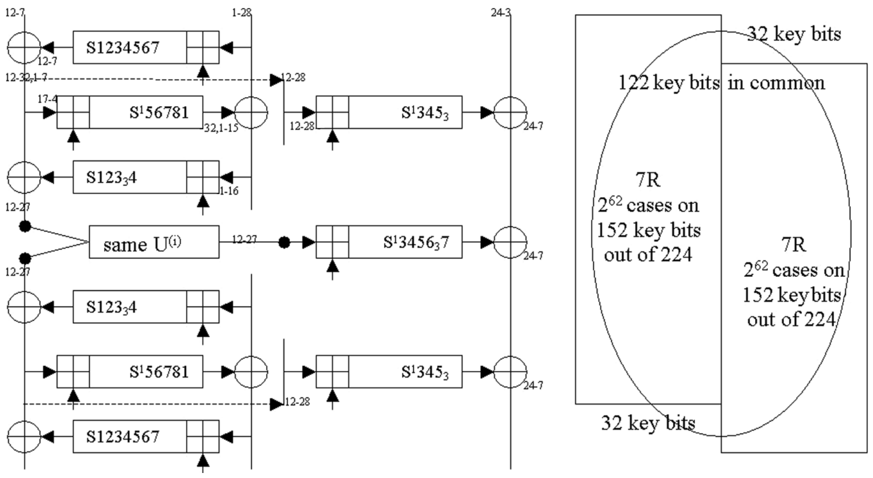

**Figure 4.** An attack with $60 + 60 + 32 + 32$ key bits for $7 + 1$ rounds.

1. We start with the enumeration of $2^{79.8}/2^{66.4}$ cases with keys on 120 bits we have obtained above, and in each case, we will do additional steps:
$$k_{123}^{(60)}, k_{567}^{(60)}, U_{12-27}^{(1..3)}.$$
2. First we look at the output of S6 (4 bits 32,1-3) in round 4. The output of S6 in R2 and R6 is already known in each of our $2^{79.8}/2^{66.4}$ cases. Therefore, the state of S6 in R4 can be determined instantly in each case, and therefore, the input of S6 in R4 can be obtained also in maybe about 4 CPU clocks per case.
3. We also know all data bits at S5 and S6 in R4. Accordingly, the key at S6 in R4 depends now on only 1 bit, the carry bit coming from S5. If for each encryption we knew 5 key bits, the key at S6 the higher ranking bit at S5, following Fact 1, we would have $2^{0.3}$ possibilities for this (already known) input of S6 in R4. Instead of checking $2^5$ cases, we can directly compute with very small precomputed table and using maybe another 4 CPU clocks, an average expected number of $2^{5-4+0.3} = 2^{1.3}$ keys on 5 bits, which are the only possible keys in each of the $2^{79.8}/2^{66.4}$ cases.
4. These propositions for the 5 key bits above must have an intersection for the three or four encryptions. For 3 KP, the probability that one fixed 5-bit key is in the intersection is about $2^{3(1.3-5)} = 2^{-11.1}$. The probability that the intersection is non-empty is about $2^{5-11.1} \approx 2^{-6.1}$. For 4 KP, the probability that the intersection is non-empty is $2^{5+4(1.3-5)} \approx 2^{-9.8}$.
5. Thus in overall time of about $(4+4)2^{79.8}$, or about $(4+4)2^{66.4}$ CPU clocks, which is only about $2^{74}/2^{61}$ GOST computations, we can obtain a list of $2^{79.8-6.1} = 2^{73.7}$/of $2^{66.4-9.8} = 2^{56.6}$ cases with keys on $120+5$ bits and 48/64 middle bits. This is the dominant step in Stage 2 of the attack. The following steps will require less time.
6. Quite importantly, in each of these $2^{73.7}/2^{56.6}$ cases, the state of S6 in R4, and therefore the carry bit entering S7 at R4, are already known. We guess 3 more key bits: the 3 lower bits for S7 in R4. We know the carry entering, and given that the data bit 28 is absent, we obtain $2^1$ possibilities for the output of S7 in R4, which is also already known in each of the $2^{73.6}/2^{56.6}$ cases. This is in each of the three/four encryptions. Accordingly, for 3 KP, we have $2^{73.7+3\cdot(1-4)} = 2^{64.7}$ cases with keys on $120+5+3$ bits. For 4 KP we get $2^{56.6+4\cdot(1-4)} = 2^{44.6}$ cases. The running time is about $2^{73.7+3}/2^{56.6+4}$ CPU clocks, which is only about $2^{63}/2^{52}$ GOST computations and can be neglected.
7. Now we guess 5 more bits in each of the rounds R246: one higher bit of S3 and four bits at S4. This allows one to compute the state of S4 in R246 and verify if the XOR of the three outputs of S4 in these rounds is confirmed, which happens with probability $2^{-4}$. For 3 KP, we enumerate about $2^{64.7+15-3\cdot4} = 2^{67.7}$ possible keys on $143 = 120+5+3+15$ bits. For 4 KP, we enumerate about $2^{44.6+15-4\cdot4} = 2^{43.6}$ possible keys on 143 bits. The running time of about $2^{64.7+15}/2^{44.6+15}$ CPU clocks can again be neglected.
8. Now we guess 3 more bits at S5 in each of the rounds R246. We can then obtain the state of S5 and check if the the XOR of the three outputs of S5 is confirmed with success probability $2^{-4}$. Moreover, we can now also compute the carry entering S6, which was previously guessed three times in each encryption in rounds 2, 4, and 6. We can then reject all the cases where the guess was incorrect, and our previous guess was true with probability $2^{0.3}$. Thus, we can gain a factor

of $2^{D \cdot 3 \cdot 0.3}$ with $D = 3/4$. Thus, in time of about $2^{67.7+9}/2^{43.6+9}$ CPU clocks, we can enumerate the following number of cases with keys on $152 = 120 + 5 + 3 + 12 + 9$ bits. For 3 KP we have $2^{67.7+9-3\cdot(4+0.9)} = 2^{62.0}$ possible keys on 152 bits. For 4 KP we have $2^{43.6+9-4\cdot(4+0.3)} = 2^{33.0}$ possible keys on 152 bits. The time can again be neglected compared to the dominating $2^{74}/2^{61}$ GOST computations. We can now summarize the whole attack, Stage 1 and Stage 2.

9. For 3 KP, in time of about $2^{74}$ GOST computations, and with $2^{68}$ bytes of memory, we can enumerate $2^{62}$ cases with keys on 152 key bits and 48 middle bits.
10. For 4 KP, overall time of about $2^{61}$ GOST computations, and with $2^{69}$ bytes of memory, we can enumerate $2^{33}$ cases with keys on 152 key bits and 64 middle bits.

### *4.3. Stage 3: An Extension to Eight Rounds*

In order to extend this attack to eight rounds, we proceed as follows:

1. We work on eight rounds of GOST and 4 KP. It will be viewed as alternatively $1+7$ and $7+1$ rounds (Figure 4).
2. First we guess the 32 key bits in the last round.
3. Then we apply the whole enumeration of Stage $1+2$ above for the seven rounds. We obtain, with the total overall running time of about $2^{32+74}/2^{32+61}$ GOST computations, and with $2^{68}/2^{69}$ bytes of memory, an enumeration of about $2^{32+62}/2^{32+33}$ cases with $32+152=184$ bits of the key.
4. It is easy to see that if we do the same in the decomposition $1+7$ rounds, we also obtain about $2^{32+62}/2^{32+33}$ cases on 184 key bits.
5. The two sets of $2^{94}/2^{65}$ cases have 122 key bits in common, which can be used to join the two distributions.
6. Very few key bits are not used in our attack; these are the 5 most significant bits in rounds 4 and 5.
7. For 3 KP, each enumeration contains only $2^{94}$ keys on 122 bits. Given any 122 common key bits chosen at random, it is present in one enumeration only with probability $2^{94-122} = 2^{-28}$, and in the intersection, we will find only $2^{94-28} = 2^{66}$ keys on 246 bits. We expect to get $2^{66}$ cases with keys on 246 bits, and the remaining 10 key bits can be determined instantly in each case, because most bits inside the cipher are known. We check every solution with the 3 KP available. And we expect to obtain an enumeration of exactly $2^{64}$ cases with 256 key bits.
8. For 4 KP, each enumeration contains only $2^{65}$ keys on 122 bits. In the intersection, we expect find at most one key on 246 bits, the right key.

We summarize our two results:

**Fact 3**. Given 3 KP for eight rounds of GOST, the full 256-bit key can be found in time of about $2^{107}$ GOST computations and $2^{68}$ bytes of memory.

**Fact 4**. Given 4 KP for eight rounds of GOST, the full 256-bit key can be found in time of about $2^{94}$ GOST computations and $2^{69}$ bytes of memory.

We can compare this to the results claimed in the original 2011 version of [1]: $2^{120}$ for 3 KP and also for 4 KP. The second result is particularly significant, $2^{26}$ times faster than in [1]. However, the software attack mentioned in [1] requires only negligible memory. It is possible to believe that $2^{120}$ and negligible memory can be as costly as $2^{94}$ time and $2^{69}$ memory.

Can we get the best of both worlds, $2^{94}$ time and negligible memory? Very surprisingly, we are now going to show how to achieve just that by another attack, a pure software attack with an SAT solver.

## 5. An SAT Solver Attack for 4 KP with the Same Complexity and Much Less Memory

We have the following result.

**Fact 5**. Given 4 KP for eight rounds of GOST, the full 256-bit key can be found in time of about $2^{94}$ GOST computations and negligible memory.

*Justification*. We proceed as follows.

1. We use the encoding of GOST described in [5].
2. We use a guess-then-determine approach with a highly non-trivial set of bits that are guessed; the other bits are determined by software.
3. We use the following specific set of 68 bits, which is built following the same pattern as exploited in our MITM attacks. The bits used are: 0-15,51-55,64-66,128-130,179-183,192-207,224-231,244-255.
4. We convert our problem to a native SAT problem with native XORs such as accepted by CryptoMiniSat 2.92. software [10].
5. We run the software $2^{68}$ times for all possible assignments of the 68 bits with a timeout of 400 seconds.
6. Computer simulations show that if the 68 bits are correct, the SAT solver will output the correct key with probability about 50%.
7. Overall, it is NOT necessary to run all the $2^{68}$ cases for 400 seconds. This is because a proportion of $1\text{-}2^{-5}$ of cases terminates automatically with UNSAT within 2 seconds average time, which is $2^{23}$ GOST encryptions.
8. Assuming that all other cases run for 400 seconds (some still terminate earlier), our conservative estimate of the attack time is $2^{68+23} + 2^{68+31-5} \approx 2^{94}$ GOST computations.

## 6. Application to Full 32-Round GOST

Following [1, 3], the problem of breaking the full 32-round GOST can be reduced to a low-data complexity attack on eight rounds. In this article we propose a plug-in replacement last step for these numerous recent attacks [1, 3]. It improves the running time of [3], and in Table 2, we give six other attacks from [1], which are substantially improved in this article. The first two are single key attacks, and the other can be applied to a diverse population of keys generated at random.

All these attacks on GOST can be compared in a meaningful way as follows. We consider the cost of recovering one full 32-round GOST key with 256-bit keys generated at random, including the time to examine all the other (nonweak) keys. The fastest known single key attack on GOST requires $2^{179}$ GOST encryptions per key with $2^{64}$ KP [2]. In comparison, our attacks with multiple keys can achieve less. We see that as the population of different GOST keys and overall data requirements grow in the multiple key setting, the cost of one key decreases in a spectacular way, down to 159 and even 120-bit security level, surprisingly low for a 256-bit military-grade cipher and government standard.

**Table 2.** Improved attacks on GOST compared to previous results in [1]; in data complexity figures, [A]CC/P means [Adaptive] Chosen Ciphertext/Plaintext attacks

| | Various attacks with complexity reduction from 2011 [1] | | | | | New cf. [1] |
|---|---|---|---|---|---|---|
| Reduction steps | Reduction 2 | Reduction 3 | Family 2 | Family 2 | Family 3 | Family 8.1 |
| Reference in [1]: | | | | | | |
| Key size | 256 | 256 | 256 | 256 | 256 | 256 |
| Key density $d$ | $2^{0}$ | $2^{-0.7}$ | $2^{-32}$ | $2^{-32}$ | $2^{-64}$ | $2^{-98}$ |
| From (data 32 R) | $2^{32}$ KP | $2^{64}$ KP | $2^{32}$ CC | $2^{32}$ ACC | $2^{64}$ KP | $2^{32}$ CP |
| Obtained (for 8 R) | **3 KP** | **3 KP** | **3 KP** | **4 KP** | **4 KP** | **3 KP** |
| Valid w. prob. | $2^{-128}$ | $2^{-96}$ | $2^{-64}$ | $2^{-64}$ | $2^{-1}$ | $2^{0}$ |
| Previous 2011 results in [1] | | | | | | |
| Time to break 8 R | $2^{120}$ | $2^{120}$ | $2^{120}$ | $2^{120}$ | $2^{120}$ | |
| Attack time 32 R | $2^{248}$ | $2^{216}$ | $2^{184}$ | $2^{184}$ | $2^{121}$ | |
| Memory bytes | small | $2^{67}$ | small | $2^{67}$ | $2^{67}$ | |
| Cost of 1 key, if | $2^{248}$ | $2^{217}$ | $2^{226}$ | $2^{226}$ | $\mathbf{2^{185}}$ | |
| key diversity $\geq$ | $2^{0}$ | $2^{0.7}$ | $2^{32}$ | $2^{64}$ | $2^{64}$ | |
| Improvement to [1] (this article) | | | | | | |
| Time to break 8 R | $2^{107}$ | $2^{107}$ | $2^{107}$ | $2^{94}$ | $2^{94}$ | $2^{107}$ |
| Attack time 32 R | $2^{235}$ | $2^{203}$ | $2^{171}$ | $2^{158}$ | $2^{95}$ | $2^{107}$ |
| Memory bytes | $2^{68}$ | $2^{68}$ | $2^{68}$ | $2^{67}$ | $2^{67}$ | $2^{68}$ |
| Cost of 1 key, if | $2^{235}$ | $2^{204}$ | $2^{203}$ | $\mathbf{2^{190}}$ | $\mathbf{2^{159}}$ | $\mathbf{2^{120}}$ |
| key diversity $\geq$ | $2^{0}$ | $2^{0.7}$ | $2^{32}$ | $2^{64}$ | $2^{64}$ | $2^{98}$ |
| Data x keys | $2^{32}$ | $2^{65}$ | $2^{64}$ | $2^{96}$ | $2^{128}$ | $2^{130}$ |

## About the Author

Nicolas T. Courtois is a cryptologist and a Senior Lecturer at University College London. He was born in Poland, received his PhD from the Paris 6 University, and then he worked as a cryptographic engineer for the French smart card industry. He is a highly influential code-breaker with more than 50 regular publications and more than 4000 citations. He has pioneered or/and achieved significant results in all of the following areas of cryptography: design and analysis of new public key cryptosystems (Sflash, Quartz, HFE), generalized linear cryptanalysis of block ciphers (Crypto 2004), cryptanalysis of LFSR-based stream ciphers with and without additional memory (Eurocrypt 2003, Crypto 2004, ICISC 2004), efficient algorithms for solving systems of multivariate equations (Eurocrypt 2000), innovative attacks on block ciphers (Asiacrypt 2001, AES'4), alternatives to Gröbner bases algorithms (Asiacrypt 2001, FSE 2012), low-data complexity cryptanalysis of block ciphers with SAT Solvers (IMA 2007), self-similarity attacks on block ciphers with black-box reductions (FSE 2008,Cryptologia 2012), advanced differential attacks (SECRYPT 2009), and importantly, in security analysis of major industrial standards and real-life cryptographic algorithms used by hundreds of hundreds of millions of people every day (E0 cipher in Bluetooth, automobile cipher KeeLoq, MiFare Classic Crypto-1 in contactless smart cards).

## References

1. Courtois, N. 2010–2012. Algebraic Complexity Reduction and Cryptanalysis of GOST. Preprint. http://eprint.iacr.org/2011/626 (accessed 1 December 2012).
2. Courtois, N. 2012. An Improved Differential Attack on Full GOST. Cryptology ePrint Archive, Report 2012/138. 15 March. http://eprint.iacr.org/2012/138 (accessed 11 December 2012).
3. Courtois, N. 2012. "Security Evaluation of GOST 28147–89 In View of International Standardisation," *Cryptologia*, 36(1):2–13.
4. Courtois, N. and M. Misztal. 2011. First Differential Attack On Full 32-Round GOST. In: *ICICS 11, LNCS 7043*, Springer, pp. 216–227.
5. Courtois, N. T., D. Hulme, and T. Mourouzis. 2012. Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis. In (informal) *Proceedings of SHARCS 2012 Workshop*, pp. 179–191, http://2012.sharcs.org/record.pdf (accessed 18 March 2012).
6. Dinur, I., O. Dunkelman, and A. Shamir. 2012. *Improved Attacks on Full GOST. FSE, LNCS 7549*, pp. 9–28.
7. Isobe, T. 2011. A Single-Key Attack on the Full GOST Block Cipher. In: *FSE 2011, LNCS 6733*, Springer, pp. 290–305.
8. Kara, O. 2008. Reflection Cryptanalysis of Some Ciphers. In: *Indocrypt 2008, LNCS 5365*, pp. 294–307.
9. Poschmann, A., S. Ling, and H. Wang. 2010. 256 Bit Standardized Crypto for 650 GE GOST Revisited. In: *CHES 2010, LNCS 6225*, pp. 219–233.
10. Sörensson, N., N. Eén, and M. Soos. CryptoMiniSat 2.92, an open-source SAT solver package based on earlier MiniSat software. http://www.msoos.org/cryptominisat2/ (accessed 22 January 2012).
11. Zabotin, I. A., G. P. Glazkov, and V. B. Isaeva. 1989. Cryptographic Protection for Information Processing Systems. Government Standard of the USSR, GOST 28147-89, Government Committee of the USSR for Standards.