J Cryptol (2018) 31:641–670
https://doi.org/10.1007/s00145-017-9266-8

Journal of
CRYPTOLOGY

Efficient Slide Attacks*

Achiya Bar-On
Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
abo1000@gmail.com

Eli Biham
Computer Science Department, Technion, Haifa, Israel
biham@cs.technion.ac.il

Orr Dunkelman
Computer Science Department, University of Haifa, Haifa, Israel
orrd@cs.haifa.ac.il

Nathan Keller
Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
nkeller@math.biu.ac.il

Communicated by Vincent Rijmen.

Received 23 December 2016 / Revised 10 July 2017
Online publication 17 August 2017

Abstract. The slide attack, presented by Biryukov and Wagner, has already become a classical tool in cryptanalysis of block ciphers. While it was used to mount practical attacks on a few cryptosystems, its practical applicability is limited, as typically, its time complexity is lower bounded by 2^n (where n is the block size). There are only a few known scenarios in which the slide attack performs better than the 2^n bound. In this paper, we concentrate on *efficient* slide attacks, whose time complexity is less than 2^n. We present a number of new attacks that apply in scenarios in which previously known slide attacks are either inapplicable, or require at least 2^n operations. In particular, we present the first known slide attack on a Feistel construction with a *3-round* self-similarity, and an attack with practical time complexity of 2^{40} on a 128-bit key variant of the GOST block cipher with *unknown* S-boxes. The best previously known attack on the same variant, with *known* S-boxes (by Courtois), has time complexity of 2^{91}.

Keywords. Slide attacks, Cycle structure, GOST, 3K-DES, 1K-AES.

*A preliminary version of the paper including only Sects. 4.1 and 5.3 appeared in FSE 2007 [4].

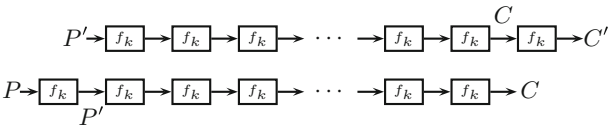


Fig. 1. A slid pair.

1. Introduction

Most modern block ciphers are constructed as a cascade of repeated keyed components, called rounds. The effectiveness of the standard cryptanalytic techniques (e.g., differential cryptanalysis [5], linear cryptanalysis [38], Square attack [16]) usually reduces drastically as the number of rounds increases. Hence, increasing the number of rounds is considered a relatively simple way to enhance the cipher's security.

Only a few cryptanalytic techniques are independent of the number of rounds. Arguably, the best known of these is the *slide attack*, introduced in 1999 by Biryukov and Wagner [10].[1] The original slide attack targets ciphers that are a cascade of ℓ *identical* functions, i.e.,

$$E_k = f_k^{\ell} = f_k \circ f_k \circ \cdots \circ f_k,$$

where f_k is a "relatively weak" keyed permutation.

The idea behind the attack is rather simple. The adversary seeks to find a *slid pair*, a pair (P, P') of plaintexts such that $P' = f_k(P)$, as demonstrated in Fig. 1. Due to the structure of E_k, the corresponding ciphertexts $C = E_k(P)$, $C' = E_k(P')$ must satisfy $C' = f_k(C)$. Hence, if a slid pair (P, P') is given, the adversary can use the "simplicity" of f_k to solve the system of equations:

$$\begin{cases} P' = f_k(P), \\ C' = f_k(C), \end{cases} \tag{1}$$

and thus to retrieve the secret key k.

As the adversary does not know which plaintext pair is a slid pair, she simply takes $O(2^n)$ random pairs (that can be constructed from $O(2^{n/2})$ known plaintexts) and tries to solve (1) for each of them. It is expected that at least one of the checked pairs is a slid pair, and then the solution of (1) yields the secret key with a high probability. (For all but a few other plaintext pairs, the equation system is expected to be inconsistent.) The data complexity of the attack is $O(2^{n/2})$ known plaintexts and the time complexity is $O(t \cdot 2^n)$, where t is the time required for solving system (1) (that is, to break f_k given two known input/output pairs).

The original slide attack can be used only when f_k is "simple," i.e., can be broken efficiently using only two known input/output pairs. Subsequent papers (e.g., [11,25]) provided methods to apply the technique in cases where f_k is more complex, usually

[1] We note that the slide attack is related to several previous techniques, including the attack of Grossman and Tucherman on Feistel constructions [30] and Biham's related-key attack [3].

at the expense of higher data and time complexities (or when the attacked cipher has a very unique structure).

The slide attack was studied in numerous papers (e.g., [1,11,18,21]) and applied to a few ciphers (most notably, the block cipher Keeloq [1]) and a few hash functions (see [28]). However, its applicability has been limited, as the time complexity of $O(t \cdot 2^n)$ is usually impractically high, and in many cases even higher than that of exhaustive key search.

Only a few general scenarios are known in which the slide attack, or its variants, have time complexity significantly smaller than 2^n. Usually, this happens when there exists an efficient way to detect the slid pairs, so that the adversary does not have to check $O(2^n)$ random plaintext pairs until a slid pair is found. These scenarios include Feistel constructions with 2-round and 4-round self-similarity (i.e., with round keys of the forms $(k_1, k_2, k_1, k_2, \ldots)$ and $(k_1, k_2, k_3, k_4, k_1, k_2, k_3, k_4, \ldots)$, respectively), possibly surrounded by key whitenings (see [11,18]), the Even-Mansour [22] construction (i.e., $E_{k_1,k_2}(P) = k_2 \oplus f(k_1 \oplus P)$, where f is a public permutation and k_1, k_2 are secret keys) and some of its variants (see [21]).

In this paper, we concentrate on *efficient* slide attacks, that is, slide attacks with complexity lower than 2^n. We present four new classes of attacks:

1. Efficient slide attacks on SP Networks (SPNs) with self-similarity. We present, for the first time, efficient slide attacks on SPNs with self-similarity (this is in contrast to Feistel constructions with self-similarity that were studied in numerous works, see below).

Consider 1K-AES, a variant of AES [17] in which all round subkeys are replaced by the same subkey (with no round constants). It seems clear that this variant is "bad," being vulnerable to the slide attack. However, the classical slide attack on it requires 2^{64} known plaintexts and has a time complexity of 2^{128} encryptions—no less than exhaustive key search! This raises the question whether the slide attack is really a threat for this variant.

We present a simple efficient slide attack on this variant with data and time complexities of 2^{64}, confirming the intuition that it is completely insecure. Our attack applies to a wide class of SP networks with 1-round self-similarity, yielding an attack with data and time complexities of $O(2^{n/2})$. We also show that in the specific case of AES, one can also break 2K-AES and 3K-AES much with an efficient slide attack, using the somewhat slow diffusion of the linear permutation of AES.

In addition, we study the more general class of SASA…SA structures [9] with 1-round self-similarity. We show that when the linear layer A is known (like in the secret S-boxes variant of AES [44]), the cipher can be broken in $O(s \cdot 2^s \cdot 2^{n/2})$ data and time, when the S-box size is s bits.

2. An efficient attack on Feistel constructions with 3-round self-similarity. We consider slide attacks on Feistel constructions with self-similarity, following [11,18,21,35,36]. While the slide-with-a-twist attack and its variants provide very efficient attacks on Feistels with 2-round and 4-round self-similarity, it appears that they do not yield an attack faster than 2^n on constructions with *3-round* self-similarity. We present the first known attack faster than 2^n on a wide class of such constructions, with data and time complexities of $O(2^{5n/6})$ and memory complexity of $O(2^{2n/3})$. Specifically, our attack is applicable for all so-called Feistel-2 structures [32], i.e., when the round function is of the form $f_k(x) = g(x \oplus k)$ for some keyless function $g(\cdot)$.

Table 1. Comparison of our attacks with the best previously known results on the same variants.

Primitive	Reference	Data	Time	Memory
AES with 1-round	[10]	$2^{n/2}$ KP	2^n	$2^{n/2}$
Self-similarity	Section 2.1	$2^{n/2}$ KP	$2^{n/2}$	$2^{n/2}$
SASA…SA with 1-round				
Self-similarity (known A)	Section 2.2	$s \cdot 2^s \cdot 2^{n/2}$ KP	$s \cdot 2^s \cdot 2^{n/2}$	$s \cdot 2^s \cdot 2^{n/2}$
DES with 3-round	[10]	$2^{n/2}$ KP	2^n	$2^{n/2}$
Self-similarity	Section 3	$2^{5n/6}$ KP	$2^{5n/6}$	$2^{2n/3}$
24-round GOST with	[25]	2^{64} KP	2^{100}	2^{64}
Unknown S-boxes	Section 5.3	2^{63} ACP	2^{63}	2^{63}
Palindromic GOST with	[14]*	2^{32} CP	2^{91}	2^{32}
Unknown S-boxes	Section 5.4	2^{40} ACP	2^{40}	2^{40}

KP known plaintexts, *CP* chosen plaintexts, *ACP* adaptively chosen plaintexts
∗ The attack of [14] assumes that the S-boxes are known

When the round function $f_k(x)$ is more structured (specifically, for all Feistel-3 structures, including a 3-round self-similar variant of DES itself), we present an improved attack that requires $O((n/s) \cdot 2^{n/2})$ adaptive chosen plaintexts and $O((n/s) \cdot 2^{n/2+s})$ time, when the S-box size is s bits.

3. A slide attack using the cycle structure of f_k; application to 24-round GOST. We present a new variant of the slide attack that finds the slid pairs directly using comparison between the cycle structures of E_k and f_k. While the data and time complexities of the attack are close to 2^n, it applies even if f_k is very complex (in which case a standard slide attack, if at all possible, would have time complexity significantly higher than 2^n). We apply the new technique to a 24-round variant of the Russian encryption standard GOST [29] with unknown S-boxes. Our attack recovers the secret S-boxes and the secret key with data and time complexity of about 2^{63}. A slide attack on the same variant can be mounted using Furuya's technique [25]; however, it would require the entire codebook and time complexity of about 2^{100} encryptions. We note that similar ideas can be used in the case where the encryption algorithm is implemented by a quantum computing device, where one can use Simon's algorithm to find the cycle efficiently (as observed in [2,34]).

4. An efficient slide attack using the cycle structure and reflection; application to a 128-bit key variant of GOST. We show that in cases where the round keys of a Feistel cipher are palindromic, our cycle-structure attack can be combined with a reflection property (see [35,36]) into a powerful attack with data and time complexity of only $O(2^{n/2})$. We apply the new attack to a 128-bit key variant of GOST in which the "full" 256-bit key has the palindromic structure $(k_1, k_2, k_3, k_4, k_4, k_3, k_2, k_1)$. While weaknesses of this variant were pointed out in previous works (e.g., [11]), the best known attack on it, presented by Courtois [14], is rather complex and has time complexity of 2^{91}, when the S-boxes are known to the adversary. Our attack breaks this variant with practical complexity of 2^{40}, *even if the S-boxes are secret*.

Thus, our attack can be used to target the original GOST by a legitimate user who wants to recover the secret S-boxes she was provided (which may be profitable, as in GOST, the same set of S-boxes is used in an entire industry, e.g., the banking industry, see [42]). If

the user is allowed to choose her secret key, she may choose it to be palindromic and then recover the secret S-boxes practically using our attack. An S-box recovery attack of this class was presented by Saarinen [41], but only for keys of the form (k, k, k, k, k, k, k, k) for which GOST is a Feistel construction with 1-round self-similarity (which is widely known to be too weak).

A comparison of our results with the best previous results on the same variants is presented in Table 1.

This paper is organized as follows. In Sect. 2, we present efficient slide attacks on SPNs with self-similarity. In Sect. 3, we present the new attack on $3K - DES$. The cycle-structure attacks are presented in Sect. 4. Finally, the applications to variants of GOST are given in Sect. 5.

2. Efficient Slide Attacks on Substitution-Permutation Networks with Self-Similarity

In this section, we study efficient slide attacks on SP networks (SPNs) with r-round self-similarity. A specific example is a variant of the encryption standard AES [17] in which the round keys are replaced by the sequence $k_1, k_2, \ldots, k_r, k_1, k_2, \ldots, k_r, \ldots$, that may be called *rK-AES*. Unlike Feistel constructions with self-similarity that have been thoroughly studied in the context of slide attacks (see Sect. 3 below), SPNs with self-similarity have not been studied specifically in previous works. Probably, the reason for that is *rK-AES* being much more immune to slide attacks than *rK-DES*.

We present several efficient slide attacks on instances of *rK-AES*. First, in the simplest case of 1-round self-similarity (i.e., 1K-AES), we present a simple efficient slide attack with data and time complexities of 2^{64} (or, more generally, $2^{n/2}$ for n-bit blocks). We then turn our attention to a more complex type of 1-round self-similarity represented by so-called SASA…SA structures (as defined in [9]), either with a known linear layer or even with a key-dependent linear layer. Finally, in the specific case of AES-like structures, we present efficient attacks on variants with 2-round and even 3-round self-similarity, exploiting the slow diffusion of the AES round function.

2.1. *A Simple Efficient Slide Attack on 1K-AES*

2.1.1. *The Structure of 1K-AES*

The AES [17] is the most widely used block cipher. We briefly recall a few details that are relevant to our attack (the interested reader can look at [17] for a complete description). The block size of AES is 128 bits, and the key sizes are 128/192/256 bits. AES is an iterated cipher composed of 10/12/14 almost identical rounds. Each round consists of four operations: SubBytes (SB), ShiftRows (SR), MixColumns (MC)—operations on the state that do not depend on the key, followed by AddRoundKey (ARK)—bitwise XOR of the state with a 128-bit subkey. In the real AES, there is an additional AddRoundKey operation before the first round, and the MC operation is omitted in the last round. We define 1K-AES to have a similar structure (with a whitening key before the first round) but the same round key in all rounds, as depicted in Fig. 2. Thus,

$$E_k(P) = f_k \circ f_k \circ \cdots \circ f_k(P \oplus k),$$

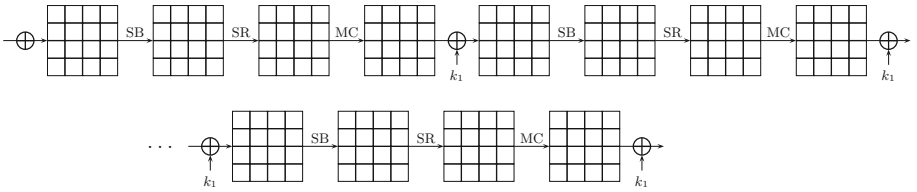


Fig. 2. Structure of 1K-AES .

where $f_k(x) = ARK_k \circ MC \circ SR \circ SB$ denotes a single round of AES. We claim that E_k can be broken with 2^{64} known plaintexts and 2^{64} time, using a simple variant of the slide attack.

2.1.2. *The Attack Algorithm*

The idea behind the attack is simple. Assume that (P, P') is a slid pair for E_k, i.e., that $f_k(P \oplus k) = P' \oplus k$. We have $P = k \oplus (SB^{-1} \circ SR^{-1} \circ MC^{-1}(P'))$. Hence, denoting $\bar{P}' = SB^{-1} \circ SR^{-1} \circ MC^{-1}(P')$, we have

$$P \oplus \bar{P}' = k. \tag{2}$$

On the other hand, by the structure of E, the corresponding ciphertexts must satisfy $C' = k \oplus (MC \circ SR \circ SB(C))$. Thus, denoting $\bar{C} = MC \circ SR \circ SB(C)$, we have

$$C' \oplus \bar{C} = k. \tag{3}$$

Combining (2) and (3), we get

$$P \oplus \bar{C} = \bar{P}' \oplus C'. \tag{4}$$

This relation allows to mount the attack described in Algorithm 1, inspired by the slide-with-a-twist attack of [11]. We note that similar observations were used in the attack of Jean et al. on the PRINCE block cipher [33] and in [21].

Algorithm 1 A slide attack on 1K-AES

Ask for the encryption of 2^{64} known plaintexts (P_i, C_i).
for Each plaintext/ciphertext pair (P_i, C_i) **do**
  Compute the value $\bar{C}_i = MC \circ SR \circ SB(C_i)$,
  Compute the value $P_i \oplus \bar{C}_i$,
  Store in a hash table the pairs $(P_i \oplus \bar{C}_i, P_i)$, indexed by the first coordinate.
end for
for Each plaintext/ciphertext pair (P_j, C_j) **do**
  Compute the value $\bar{P}_j = SB^{-1} \circ SR^{-1} \circ MC^{-1}(P_j)$,
  Compute the value $\bar{P}_j \oplus C_j$,
  Check for entries in the hash table whose first coordinate matches it.
end for
for Each collision in the table $(P_i \oplus \bar{C}_i = \bar{P}_j \oplus C_j)$ **do**
  Test the key candidate $k = P_i \oplus \bar{P}_j$.
end for

By the birthday paradox, it is expected that the data set contains a slid pair, i.e., a pair satisfying $f_k(P_i) = P_j$, with a non-negligible probability. For a random pair (P_i, P_j), the probability that $P_i \oplus \bar{C}_i = \bar{P}_j \oplus C_j$ is 2^{-128}, and thus, only a few collisions are expected in the table. These collisions include the collision induced by the slid pair, which suggests the correct value of k. The data complexity of the attack is 2^{64} known plaintexts, and the time and memory complexities are 2^{64} operations, as asserted.

We note that the collision we are looking for is of the form $P_i \oplus \bar{C}_i = \bar{P}_j \oplus C_j$. Given that each side of the equation depends on a single plaintext/ciphertext pair, one can easily transform the attack into a memoryless one, as follows:

Define $g_1(P_i) = P_i \oplus \bar{C}_i$ and $g_2(P_j) = \bar{P}_j \oplus C_j$. Use a memoryless collision finding algorithm (such as [24]) to find a collision between $g_1(\cdot)$ and $g_2(\cdot)$. As such algorithms take about $O(2^{n/2})$ queries to $g_1(\cdot)$ and $g_2(\cdot)$, the time complexity of the attack is $O(2^{n/2})$. The main change is that the data requirement becomes an adaptive chosen plaintext one (following the way memoryless collision algorithms operate), in exchange for the reduction of the memory complexity.

The attack applies to any SPN with 1-round self-similarity in which the key-dependent operation is XOR or modular addition, regardless of the non-key-dependent operations. The data and time complexities remain the same — $O(2^{n/2})$.

2.2. *An Efficient Slide Attack on SASA…SA*

We now turn our attention to the more general SPN structures known as the SASA…SA constructions. First analyzed by Biryukov and Shamir in [9], such constructions have layers of bijective nonlinear S-boxes S (the S-boxes may be different) followed by invertible affine layers A. Both types of layers may be key-dependent. General SASA…SA constructions (without self-similarity) were recently studied in several papers, for example [6,7,39]. We consider SASA…SA constructions with 1-round self-similarity, i.e., where the same combination $A \circ S$ is applied r times sequentially. The generic attack of Algorithm 1 obviously cannot be applied to this constructions, due to the complex key-dependent operations.

We first consider the case when the A layer is not key-dependent (like in the AES variant studied in [44]) and present a simple attack with data and time complexities of $O(s \cdot 2^s \cdot 2^{n/2})$ for an s-bit S-box. We then extend the attack to the case when A is key-dependent, with data complexity of $O(s \cdot 2^s \cdot 2^{n/2})$ and time complexity of $O(n^3 \cdot 2^n)$. (The latter attack falls a bit out of our scope, requiring more than 2^n operations. We chose to present it, as it uses similar techniques like the previous attacks and is much faster than the best one can get with a classical slide attack on the same variant.)

2.2.1. *Attacking SASA…SA When A is Not Key-Dependent*

Let $P' = A \circ S(P)$, i.e., (P, P') is a slid pair. As observed in [25], if (P, P') is a slid pair, then so is $(E_k(P), E_k(P'))$, as well as $(E_k^i(P), E_k^i(P'))$ for any i. Hence, using adaptive plaintext queries, one can generate as many slid pairs as needed out of a single slid pair.

Algorithm 2 A slide attack on SASA…SA

Pick $2^{n/2}$ plaintexts P_i and for each of them ask for the encryption of the chain $P_i, E_k(P_i), E_k^2(P_i), \ldots, E_k^t(P_i)$.
for Each plaintext chain $P_i, E_k(P_i), E_k^2(P_i), \ldots, E_k^t(P_i)$ **do**
Find all the locations for which the same S-box has the same value.
Store the list of locations along with the chain in a hash table indexed by the list of locations.[2]
end for
for Each plaintext chain $P_i, E_k(P_i), E_k^2(P_i), \ldots, E_k^t(P_i)$ **do**
Compute the chain $A^{-1}(P_i), A^{-1}(E_k(P_i)), A^{-2}(E_k^2(P_i)), \ldots$.
Find all the locations for which the same S-box has the same output value according to the computed chain.
Check for entries in the hash table whose list of locations matches the found locations.
end for
for Each collision in the locations (chains $P_i, E_k(P_i), \ldots$ and $A^{-1}(P_j), A^{-1}(E_k(P_j)), \ldots$) **do**
Collect for each S-box the input/output pairs defined by the chains, and deduce the S-boxes.
end for

In the data collection phase of the attack, we take $2^{n/2}$ plaintexts, and generate from each of them a sequence of plaintexts, i.e., $P, E_k(P), E_k^2(P), E_k^3(P), \ldots, E_k^t(P)$ for a parameter t we shall choose later. Then, we detect the slid pairs (P, P') (and their encryptions) efficiently using the following observation: If in the chain $P, E_k(P), E_k^2(P), E_k^3(P), \ldots, E_k^t(P)$ the same value enters one of the S-boxes twice (e.g., in $E_k^1(P)$ and $E_k^8(P)$), then the outputs of the respective S-boxes in the sequence generated by P' are equal (i.e., we have $A^{-1}(E_k^1(P')) = A^{-1}(E_k^8(P'))$). This observation allows an immediate identification of the slid pairs (without the need of comparing all sequences to all other sequences). After finding the slid pairs, the adversary obtains a large number of input/output pairs for each of the S-boxes, which is sufficient to identify the S-boxes. The attack is depicted in Algorithm 2.

The value of t depends on two factors: The length of the chains required to ensure that enough collisions exist, and the length of the chains required to ensure that we can recover the full S-box. A pair of chains of length t contains t pairs of plaintexts $(E_k^a(P_i), E_k^a(P_j))$, each with n/s S-boxes that can collide. As we have to obtain an n-bit filtering condition, and as each collision offers an s-bit filtering condition, we need $t \cdot \frac{n}{s} \cdot 2^{-s} \geq \frac{n}{s}$, i.e., $t \geq 2^s$. At the same time, obtaining 2^s different inputs to each and every S-box (where the inputs are chosen at random), requires $\ln 2 \cdot s \cdot 2^s$ inputs (due to the coupon collector's nature of the problem), i.e., we need $t = O(s \cdot 2^s)$. Therefore, the data and time complexities of the attack are $O(s \cdot 2^s \cdot 2^{n/2})$.

2.2.2. *Attacking SASA…SA When A is Key-Dependent*

In this case, we need to try all pairs of chains, as we cannot obtain direct access to the output of the S-boxes. To identify whether a candidate pair of chains is indeed slid, we use the above observation in a different way. Recall that when the inputs to the S-box

[2] Instead of storing the list of locations, one can define a lexicographic order (or any other standardized order), and apply a hash function on the list.

are the same, then so are the outputs. This fact translates into an s-bit linear relation on the chain $A^{-1}(P_i)$, $A^{-1}(E_k(P_i))$, $A^{-2}(E_k^2(P_i))$, ... Hence, for each pair of candidate chains, one can construct the relevant system of linear equations for A^{-1} and try to solve it. If the solution is consistent (by having a few more equal values, one can obtain an over-determined system of linear equations that is expected to be inconsistent for non-slid chains), one obtains the correct value of A, and then the S layer can be recovered like in the previous attack.

As there are 2^n pairs of candidate chains, and as solving the set of n linear equations takes $O(n^3)$ operations, the total time complexity of the attack is $O(n^3 \cdot 2^n)$. The memory and data complexities are $O(s \cdot 2^s \cdot 2^{n/2})$.

2.3. *Efficient Slide Attack on Specific Instances of 2K-AES and 3K-AES*

A slight modification of the above attack presented in Sect. 2.2 can be used to attack rK-AES with $r > 1$, if the diffusion in the linear layer is not full. We demonstrate it in the case of the AES, for which we can attack variants with 2-round and 3-round self-similarity. We denote these variants below by 2K-AES and 3K-AES and alert the reader that unlike the attack on 1K-AES presented in Sect. 2.1, these attacks do exploit the specific structure of the AES.

We start with an attack on 2K-AES. Consider a slid pair (P, P') such that $P' \oplus k_1 = ARK_{k_1} \circ MC \circ SR \circ SB \circ ARK_{k_2} \circ MC \circ SR \circ SB(P \oplus k_1)$. As the ARK_{k_1} on the right-hand side cancels out with the $\oplus k_1$ on the left-hand side, the second round's SB, SR, MC operations can be easily inverted, by computing $\tilde{P}' = SB^{-1} \circ SR^{-1} \circ MC^{-1}(P')$ and obtaining $\tilde{P}' = ARK_{k_2} \circ MC \circ SR \circ SB(P \oplus k_1)$. Moreover, as MC and SR are linear operations which can be easily exchanged with the ARK operation (by applying the inverse operations to the key), $\tilde{P}'$ can be rewritten as:

$$\tilde{P}' = MC \circ SR \circ ARK_{SR^{-1} \circ MC^{-1}(k_2)} \circ SB \circ (P \oplus k_1).$$

Hence, for a given P' we compute $\bar{P}'$ defined as

$$\bar{P}' = SR^{-1} \circ MC^{-1}(\tilde{P}') = SR^{-1} \circ MC^{-1} \circ SB^{-1} \circ SR^{-1} \circ MC^{-1}(P').$$

As a result, when (P, P') is a slid pair (and thus, the chains that are generated from them are slid chains), two bytes with the same input in the chain of P must have the same output in the chain of $\bar{P}'$. This immediately allows to apply Algorithm 2. Moreover, note that there is no need to recover the full S-box, as once the slid chain is found, it is trivial to extract k_1 and $SR^{-1} \circ MC^{-1}(k_2)$. Therefore, a slightly lower data/time complexity is needed for this attack: Only 2^{69} adaptive chosen plaintext and ciphertext pairs are needed, as well as 2^{69} time.

The attack on 3K-AES is similar in nature. Due to the additional round, instead of looking at equality in inputs and outputs from a single S-box (an 8-bit condition), we consider the AES' super S-box (essentially, a column of 4 S-boxes). As a result, applying Algorithm 2 with 32-bit S-boxes results in data and time complexity of about 2^{81}.

3. Efficient Slide Attacks on Feistel Constructions with 3-round Self-Similarity

Historically, a main target of the slide attack and its generalizations have always been a Feistel construction with r-round self-similarity, i.e., whose sequence of round functions is $f_1, f_2, \ldots, f_r, f_1, f_2, \ldots, f_r, \ldots$. A specific example is a variant of the former encryption standard DES [40] in which the round keys are replaced by the sequence $k_1, k_2, \ldots, k_r, k_1, k_2, \ldots, k_r, \ldots$ (where $k_1, \ldots, k_r$ are independent of each other), called in [10] *rK-DES*.

In [10], Biryukov and Wagner showed that 2K-DES (and more generally, a Feistel construction with n-bit blocks and 2-round self-similarity) can be broken with $O(2^{n/2})$ adaptively chosen plaintexts and $O(2^{n/2})$ time using the slide attack. In [11], Biryukov and Wagner presented the *complementation slide* and *slide with a twist* techniques and used them to break 2K-DES in $O(2^{n/4})$ chosen plaintexts and $O(2^{n/4})$ time, and also to break 4K-DES with the same complexity. In [21], Dunkelman et al. introduced the *mirror slidex* attack and used it to break 2K-DES surrounded by key whitenings in $O(2^{n/2})$ known plaintexts and $O(2^{n/2})$ time. Finally, Dinur et al. [18] were able to break 4K-DES surrounded by key whitenings in $O(n \cdot 2^{n/2})$ known plaintexts and $O(n \cdot 2^{n/2})$ time using *enhanced reflection*.

Neither of these attacks seems to apply to 3K-DES, i.e., a Feistel construction with 3-round self-similarity depicted in Fig. 3. Hence, the best one can currently do is to apply the standard slide attack, which results in data complexity of $O(2^{n/2})$ known plaintexts and time complexity of $O(t \cdot 2^n)$ where t is the time required for breaking 3-round DES given two input/output pairs.

In this section, we present a new attack on this construction with a lower time complexity of $O(2^{5n/6})$, at the expense of a higher data complexity of $O(2^{5n/6})$. Our attack exploits slid pairs of a specific type and uses ideas developed in the attacks presented in Sect. 2.

In addition, for a more specific class of round functions which contains the round function of DES itself (or any Feistel-3 construction), we present a slide attack on 3K-DES with data complexity of $O((n/s) \cdot 2^{n/2})$ known plaintexts and $O((n/s) \cdot 2^{n/2+s})$ time, when the S-box size is s bits.

3.1. *The Structure of 3K-DES and Notations*

The DES [40] was the US encryption standard until 2001, and the most widely used block cipher worldwide until the end of the 1990s. Its structure is well known and can be found in [40]. We briefly recall a few details that are relevant to our attack. The block size of DES is 64 bits, and the key size is 56 bits. DES is a 16-round Feistel construction. Each round function $f_i : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$ consists of four operations: a linear expansion function E that expands the state to 48 bits, XOR with a 48-bit subkey, an S-box layer S which shrinks the state to 32 bits and a permutation P. The only key-dependent operation in f_i is the XOR with a subkey.

In our attack, we consider the following natural abstraction of 3K-DES (called "Feistel-2 construction with 3-round self-similarity," according to the terminology of [32]). Let k_1, k_2, k_3 be independent $(n/2)$-bit subkeys. Let E_{k_1,k_2,k_3} be a $3m$-round Feistel construction with n-bit blocks, in which the round function f_{k_i} has the form $f_{k_i}(x) = g(k_i \oplus x)$,

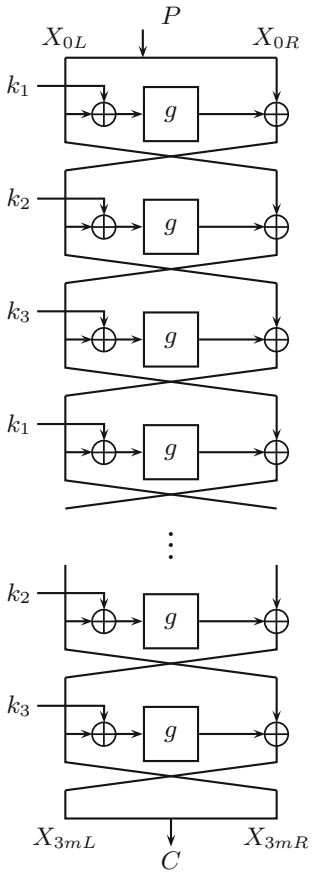


Fig. 3. Structure of 3K-DES.

where $k_i := k_{(i \bmod 3)+1}$ and g is a public keyless function. The structure of E is demonstrated in Fig. 3.

Denote by h the first three Feistel rounds of E. It is clear that $E = h^m$. For a given plaintext/ciphertext pair (P, C), we denote the input to round i in the encryption process of P by X_i (e.g., $P = X_0$), and the input to round i in the encryption process of C (assuming that it is encrypted) by Y_i (e.g., $C = Y_0$), as shown in Fig. 3. Note that if (P, P') is a slid pair for E (i.e., if $P' = h(P)$), then $P' = X_3$ and $C' = Y_3$. We also use X_L to denote the left half of X and X_R to denote the right half of X.

3.2. *The Observations Behind the Attack*

The basic idea of the attack is to search for slid pairs that satisfy the following additional property:

$$Y_{1L} = X_{1L}. \tag{5}$$

We use a few observations. Assume that (5) holds for some slid pair (P, P'). Then we can deduce the following:

1. The outputs of the second round function f_{k_2} in the encryptions of P and C are equal. By the Feistel structure, this implies $X_{0L} \oplus X_{2L} = Y_{0L} \oplus Y_{2L}$. As $X_{2L} = P'_R$ and $Y_{2L} = C'_R$, we obtain $P_L \oplus P'_R = C_L \oplus C'_R$, or equivalently,

$$P_L \oplus C_L = P'_R \oplus C'_R. \tag{6}$$

   Conversely, if the keyless function g is invertible then (6) implies (5), and even if g is a random function, (6) implies that (5) holds with a constant non-negligible probability.
2. By the Feistel construction, (5) implies $X_{0R} \oplus f_{k_1}(X_{0L}) = Y_{0R} \oplus f_{k_1}(Y_{0L})$, or equivalently, $X_{0R} \oplus Y_{0R} = f_{k_1}(X_{0L}) \oplus f_{k_1}(Y_{0L})$. Now, consider the applications of g as part of f_{k_1} in the encryptions of P and C. We know that the input difference between these two applications is $X_{0L} \oplus Y_{0L}$ (as the key addition does not change the difference), and that the output difference is $X_{0R} \oplus Y_{0R}$. For a random function g, given an input difference α and an output difference β, there exists one pair of inputs (x, x') on average such that $x \oplus x' = \alpha$ and $y \oplus y' = \beta$. Moreover, given the full Difference Distribution Table (DDT) of g and the input/output differences (α, β), one can obtain instantly all possible *actual values* of the input pairs (x, x') that correspond to those differences, whose total number is usually very small.
   Hence, if we could pre-compute the full DDT of the function g, our knowledge would be sufficient to recover the actual values $(X_0 \oplus k_1, Y_0 \oplus k_1)$, and hence, to get a suggestion for the subkey k_1. As we cannot keep in memory the full DDT (since it requires 2^n space) or even compute it (since this also takes 2^n time), we will exploit this observation in a more subtle way.
3. By the same argument, applied with P', C' instead of P, C, if (5) holds then we know the input and output differences in the applications of g as part of f_{k_3} in the encryptions of P and C. This would be sufficient for recovering k_3, and could we pre-compute and store the full DDT of g.
4. Once k_1, k_3 are known, a candidate for k_2 can be instantly found by comparing a partial encryption of P with a partial decryption of P'.

3.3. *The Attack Algorithm*

A naïve way to exploit the above observations in a slide attack is to go over many random pairs (P, P'), and for each pair assume that it is a slid pair that satisfies (5), use the above observations to get candidates for the subkeys k_1, k_2, k_3 and check them by trial encryption. However, as the probability of a random plaintext pair to be a slid pair that satisfies (5) is $2^{-n} \cdot 2^{-n/2} = 2^{-3n/2}$, such a naïve attack has time complexity of $\Omega(2^{3n/2})$.

Instead, we use relation (6) to check many candidate plaintext pairs simultaneously, and construct "slices" of the DDT that are sufficient for our purposes instead of the full DDT in order to save time and space. The attack algorithm is given in Algorithm 3.

3.4. *Analysis of the Attack*

The data complexity of the attack is $O(2^{5n/6})$ known plaintexts. Each plaintext enters tables T_1 or T_2 with probability $2^{n/3-n/2} = 2^{-n/6}$, i.e., the expected size of the tables T_1, T_2 is $2^{2n/3}$. Hence, the total number of candidate slid pairs the attack may analyze is $2^{4n/3}$. Of these pairs, the attack identifies the pairs (P, P') satisfying $P_L \oplus C_L, P'_R \oplus C'_R \in L$. Given that the size of L is $2^{n/3}$, the probability that (6) holds is $2^{-n/3}$ (since this is the probability of collision between two random elements in a set of size $2^{n/3}$). Hence, the probability that a pair (P, P') examined in the attack is a slid pair satisfying (6) is $2^{-n} \cdot 2^{-n/3} = 2^{-4n/3}$. Therefore, with a constant non-negligible probability, the data contain a slid pair that satisfies (6), and hence satisfies (5) (again, with a constant probability). As any slid pair suggests the correct subkeys (possibly, along with a few additional subkey candidates), the attack recovers the secret key with a constant probability. The success probability can be easily increased by enlarging the data complexity by a fixed constant (i.e., examining additional plaintexts).

The time complexity of the first external loop is $2^{5n/6}$, dominated by encrypting the plaintexts. The second external loop is repeated for $2^{n/3}$ values of α and its complexity

Algorithm 3 A slide attack on 3K-DES

Choose an arbitrary list L of $2^{n/3}$ $n/2$-bit values.
Collect $2^{5n/6}$ known plaintext/ciphertext pairs (P_i, C_i).
Initialize three empty hash tables T_1, T_2, and T_3.
for All plaintext/ciphertext pairs (P_i, C_i) **do**
  if $P_{iL} \oplus C_{iL} \in L$ **then**
    Add $(P_{iL} \oplus C_{iL}, P_{iL})$ to T_1 (which is indexed by $P_{iL} \oplus C_{iL}$)
  end if
  if $P_{iR} \oplus C_{iR} \in L$ **then**
    Add $(P_{iR} \oplus C_{iR}, P_{iR})$ to T_2 (which is indexed by $P_{iR} \oplus C_{iR}$)
  end if
end for
for All $\alpha \in L$ **do**
  Compute the row of the DDT of g that corresponds to the input difference α.
  for All entries of T_1 satisfying $P_{iL} \oplus C_{iL} = \alpha$ **do**
    Assume that (5) holds and use the above observations to instantly obtain a suggestion for the subkey k_1.
    Using k_1, partially encrypt P through the first round, and store the pair (X_{i1L}, X_{i1R}) in the hash table T_3.
  end for
  for All entries of T_2 satisfying $P'_{jL} \oplus C'_{jL} = \alpha$ **do**
    Assume that (5) holds and use the above observations to instantly obtain a suggestion for the subkey k_3.
    Using k_3, partially decrypt P'_j through the third round to compute (X'_{j2L}, X'_{j2R}).
    Search for entries in the table T_3 whose first coordinate (i.e., X_{i1L}) matches X'_{j2R}.
    for All matching entries (X_{i1L}, X_{i1R}) and (X'_{2jL}, X'_{2jR}) **do**
      Use the values X_{i1R} and X_{i2L} to extract k_2 from f_{k_2}.
      Check the resulting suggestion (k_1, k_2, k_3) by a trial encryption.
    end for
  end for
end for

is dominated by computing a row in the difference distribution table that requires $2^{n/2}$ operations for each value of α. (The first internal loop takes $2^{n/3}$ operations for each α, and the second internal loop takes $2^{n/6}$ encryptions, since the expected number of collisions with the table T_3 is $2^{n/3} \cdot 2^{n/3} \cdot 2^{-n/2} = 2^{n/6}$.) Therefore, the total time complexity of the attack is $O(2^{5n/6})$ encryptions.

The memory complexity is $O(2^{2n/3})$, dominated by the storage required for the tables T_1, T_2. (Note that the plaintext/ciphertext pairs themselves that do not enter tables T_1 and T_2 can be discarded.) As T_3 is built for each value of α independently, and as each α value is expected to correspond to $2^{n/3}$ pairs in T_1, the size of T_3 is just $O(2^{n/3})$.

3.5. *Attack on 3K-DES Using the Structure of the Round Function*

Let us assume now that the round function F in the Feistel construction has the following specific form (sometimes referred to as Feistel-3, as opposed to the more general Feistel-2 structure we considered above): first, the input of F is divided into s-bit chunks that are independently processed through an S-box layer, and then a linear transformation is applied to the entire state. This structure is satisfied by DES, with $|s| = 6$ (up to the "linear expansion" operation, which changes the analysis only slightly and thus is omitted here), and by many other Feistel ciphers. In this case, we present an improved slide attack that takes only $O((n/s) \cdot 2^{n/2})$ adaptive plaintexts (i.e., relies on finding a single slid pair) and $O((n/s) \cdot 2^{n/2+s})$ time.

Consider a candidate pair (P, P') and their corresponding ciphertexts (C, C'), respectively. For such a pair, we know that $P' = f_{k_3}(f_{k_2}(f_{k_1}(P)))$ and $C' = f_{k_3}(f_{k_2}(f_{k_1}(C)))$, which allows for a simple meet in the middle attack.

Namely, it is possible to guess the part of k_1 that allows partial encryption of a single S-box for P and for C. Similarly, one can guess the part of k_3 that allows partial decryption of a single S-box for P' and for C'. Hence, by guessing s key bits from k_1 and s from k_3, we obtain a $2s$-bit filter condition (due to the existence of two "input/output" pairs to 3 rounds of DES). Obviously, one can extend the attack to find more key bits by targeting the other S-boxes, resulting in a simple attack on 3 rounds of DES.

This simple meet in the middle attack can be exploited in a slide attack as well. First of all, we generate from each plaintext a chain of a few encryptions $P, E_k(P), E_k(E_k(P)), \ldots$. It is easy to see that each of the 3-round DES instances that need to be attacked uses the same order of subkeys, and thus, guessing s bits of k_1 allows for the partial encryption of all the values in the chain by one round.

The attack is thus very simple: For each chain $P, E_k(P), E_k(E_k(P)), \ldots$, we guess s bits of subkey material for k_1 and store in the table the resulting partial encryptions of the chain values (under all k_1 guesses). Similarly, for each chain $P', E_k(P'), E_k(E_k(P')), \ldots$ we try s bits of k_3, and perform partial decryption to search in the table for collisions. Each such collision suggests a candidate slid chain as well as candidate keying material. Once a slid chain is identified, it is easy to extract the remaining bits of k_1 and k_3, and to use them to retrieve k_2.

For a chain of length t, we obtain $t \cdot s$ filtering bits for the meet in the middle parts. There are $2^{n/2+s}$ candidate chains from each side, resulting in 2^{n+2s} possible pairs of

chains. By having a $n + 2s$-bit filtering, we can immediately identify the slid chains, suggesting that $t = n/s + 2$ is sufficient.

The resulting data complexity is $(n/s + 2) \cdot 2^{n/2}$ adaptive chosen plaintexts and ciphertexts. The memory and time complexities are both $O((n/s) \cdot 2^{n/2+s})$.

4. Efficient Slide Attacks using the Cycle Structure of Permutations

In this section, we present a new variant of the slide attack that exploits the relation between the cycle structures of the entire cipher $E_k = f_k^\ell$ and the round function f_k to detect slid pairs efficiently. First, we present a generic attack with data and time complexities of 2^{n-1} that applies even when f_k is very complex. Then, we show that for Feistel constructions with a palindromic key schedule the complexity of the generic attack drops to $O(2^{n/2})$, using a reflection property.

4.1. *A Generic Attack Using the Cycle Structure*

4.1.1. *Facts and observations*

First we recall several facts regarding the cycle structure of permutations that are used in our attack. Let $g : GF(2^n) \to GF(2^n)$ be a random permutation. For every $x \in GF(2^n)$ we denote $CycleLength(x) = \min\{k > 0 | g^k(x) = x\}$. Following the classical results of [26,27] on the distribution of cycle sizes in random permutations (see also [23, p. 596]), we expect that the largest cycle $Cycle_{max}$ has an expected size of $(1 - 1/e) \cdot 2^n$, and for every $x \in GF(2^n)$, $\Pr[x \in Cycle_{max}] \approx 1 - 1/e$. Here, $e = 2.7182\ldots$ is the basis of the natural logarithm. Moreover, we expect that the second largest cycle has a size of $1/e \cdot (1-1/e) \cdot 2^n$, and generally, the size of the ith largest cycle is $(1/e)^{i-1} \cdot (1-1/e) \cdot 2^n$.

The main observation used in our attack is the following: Let P be an arbitrary plaintext. We consider the cycles of P with respect to f_k and E_k, denoted by $Cycle_{f_k}(P)$ and $Cycle_{E_k}(P)$, respectively. Denote the length of $Cycle_{f_k}(P)$ by m_1, i.e., $m_1 = \min\{t > 0 | f_k^t(P) = P\}$. Similarly, denote the length of $Cycle_{E_k}(P)$ by m_2. As $E_k = f_k^\ell$, the relation $E_k^{m_2}(P) = P$ can be written as $f_k^{l \cdot m_2}(P) = P$. It clearly follows that $m_1 | \ell \cdot m_2$. On the other hand, $m_2 = \min\{t > 0 | E_k^t(P) = P\} = \min\{t > 0 | f_k^{(t \cdot \ell) \bmod m_1}(P) = P\}$. Combining these two statements, we get $m_2 = \min\{t > 0 | t\ell = 0 \bmod m_1\}$, or equivalently,

$$m_2 = m_1 / \gcd(m_1, \ell). \tag{7}$$

In particular, if $\gcd(m_1, \ell)$=1, then $m_1 = m_2$.

If $\gcd(m_1, \ell) = 1$, we can use Euclid's extended algorithm to find $1 \le d_1 \le \ell - 1$ and d_2 such that $d_1 \cdot m_1 = \ell - 1 \pmod{\ell} = d_2 \cdot \ell - 1$. For those d_1, d_2, we have

$$f_k^{d_1 \cdot m_1 + 1}(P) = f_k(P), \qquad \text{and} \qquad f_k^{d_1 \cdot m_1 + 1}(P) = f_k^{d_2 \cdot \ell}(P) = E_k^{d_2}(P).$$

Hence, the pair $(P, E_k^{d_2}(P))$ is a slid pair for E_k.

It is important to note that the adversary does not have access to f_k and thus cannot compute m_1. Only m_2 can be computed, by sequential encryption using E_k. However, the adversary may assume that $\gcd(m_1, \ell) = 1$ and then use the value $m_1 = m_2$ to compute d_1, d_2 and find the candidate slid pair $(P, E_k^{d_2}(P))$. If the assumption $\gcd(m_1, \ell) = 1$ is correct, the found pair must be a slid pair. Furthermore, as observed in [11], this implies that for every t, the pairs $(E_k^t(P), E_k^{d_2+t}(P))$ must also be slid pairs.

4.1.2. *The Attack Algorithm and Its Analysis*

Based on the above observations, we can mount the following attack.

1. Fix an arbitrary plaintext P_0.
2. Ask for a sequential encryption of P_0 through E_k, i.e., for the sequence $E_k(P_0)$, $E_k^2(P_0) = E_k(E_k(P_0))$, $E_k^3(P_0), \ldots$, until $E_k^{m_2}(P_0) = P_0$ is encountered for the first time.
3. Use the extended Euclid algorithm to find d_1, d_2 such that $d_1 \cdot m_2 = d_2 \cdot \ell - 1$.
4. Deduce that $(P, E_k^{d_2}(P))$ is a candidate slid pair and check the guess by attacking f_k using $\{(E_k^t(P), E_k^{d_2+t}(P))\}_{t \geq 0}$ as input/output pairs.

For a random value of x, $\mathbb{E}[CycleLength(x)] = (1 - 1/e) \cdot 2^n$, and thus, we expect to get about $(1 - 1/e) \cdot 2^n$ pairs of the form $\{(E_k^t(P), E_k^{d_2+t}(P))\}_{t \geq 0}$. This amount is sufficient for most possible attacks on f_k, including adaptive chosen plaintext and ciphertext attacks. Hence, Step 4 is expected to succeed even for fairly complex functions f_k (e.g., f_k being the full 16-round DES).

The attack succeeds if $gcd(m_1, \ell) = 1$. For a random permutation f_k, this occurs with probability $\varphi(\ell)/\ell$, where $\varphi(\ell) = |\{1 \leq d \leq (\ell - 1) : \gcd(d, \ell) = 1\}|$ is the Euler function. In case of failure, we can apply the attack again with another plaintext P_1 that is not included in the sequence $E_k^i(P_0)$. If the attack fails again, we can take a new plaintext P_2 etc. After k applications of the attack, the total success probability is $1 - (1 - \varphi(\ell)/\ell)^k$, which approaches 1 quickly.

As the expected length of $Cycle_{E_k}(P_0)$ is $(1 - 1/e) \cdot 2^n$), the data complexity of the attack is very large — almost the entire codebook.

The advantage of the attack over a standard slide attack is the time complexity — $O(2^{n-1} + t)$, where t is the time required for breaking f_k with $O(2^{n-1})$ known input/output pairs (compared to $O(2^n \cdot t)$ for the standard slide attack). Hence, the attack is advantageous in cases where f_k is very complex, so that t is large. In Sect. 5, we exemplify this technique with an attack on 24-round GOST with unknown S-boxes, where the function f_k is 8-round GOST with unknown S-boxes. Due to the complexity of f_k, our attack is 2^{36} times faster than the best previously known slide attack on the same variant.

We note that when the encryption algorithm is implemented by a quantum computing device capable of encrypting a quantum state, one can use the quantum cycle finding algorithm by Simon [43]. In such a case, the detection of the cycle itself can be done in time $O(n)$ using $O(1)$ quantum queries to the device (by generating an entangled state composed of all possible inputs, which is then encrypted to an entangled state composed of all possible outputs). After finding the cycle length, one can either exploit

a standard cryptanalytic attack or use some advanced quantum attack algorithms, e.g., those presented in [2,34].

4.2. *An Efficient Attack on Feistel Constructions with a Palindromic Key Schedule*

The attack algorithm described above does not depend on the exact cycle structures of E_k and f_k, apart from its success probability (that depends on the probability $\Pr[\gcd(m_1, \ell) = 1]$). Hence, if for some cipher E_k the cycle structure deviates from random, such that shorter cycles are more likely to occur, the same attack holds with a lower complexity (as the complexity is dominated by the computation of $Cycle_{E_k}(P_0)$).

A notable case in which such a reduction occurs is Feistel ciphers with a palindromic key schedule, i.e., round subkeys of the form $k_1, k_2, \ldots, k_r, k_r, k_{r-1}, \ldots, k_1$ for some r. Let f_k be such a construction, and let $E_k = f_k^\ell$ as before (so that E_k consists of $2r\ell$ Feistel rounds). Consider a sequential encryption of a plaintext P_0, as performed in the attack described above, and denote the input of the j'th round in the encryption process of $E_k^i(P_0)$ by $X[2r\ell \cdot i + j]$ (for $j = 0, 1, \ldots, 2r\ell - 1$).

We use the following observation, introduced in [35,36] and called *reflection property*. Assume that for some $i \in \mathbb{N}$ and for some $0 \leq m \leq \ell - 1$, the intermediate value $X = X[2r\ell \cdot i + 2rm + r]$ satisfies $X_L = X_R$. In such a case, by the palindromic Feistel structure, we have $X[2r\ell \cdot i + 2rm] = X[2r\ell \cdot i + 2r(m+1)]$. Furthermore, the reflection property can be extended all the way in both directions to obtain

$$P_0 = X_0 = X[2r\ell \cdot 2i + 2r(2m+1)]. \tag{8}$$

This intermediate value is not seen by the adversary who sees only values of the form $E_k^j(P_0) = X[2r\ell \cdot j]$. However, if we denote $c = \frac{\ell}{\gcd(2m+1,\ell)}$, then (8) implies

$$\begin{aligned} P_0 &= X\left[c(2r\ell \cdot 2i + 2r(2m+1))\right] = X\left[2r\ell \cdot \left(2ic + \frac{2m+1}{\gcd(2m+1,\ell)}\right)\right] \\ &= E_k^{2ic + \frac{2m+1}{\gcd(2m+1,\ell)}}(P_0). \end{aligned}$$

It follows that in such a case,

$$CycleLength_{E_k}(P_0) \leq 2ic + \frac{2m+1}{\gcd(2m+1,\ell)}. \tag{9}$$

Furthermore, the same argument holds if some $X' = X[2r\ell \cdot i + 2rm]$ satisfies $X'_L = X'_R$, and in such a case,

$$CycleLength_{E_k}(P_0) \leq 2ic' + \frac{2m}{\gcd(2m,\ell)}, \tag{10}$$

where $c' = \ell / \gcd(2m, \ell)$.

We now analyze the expectation of the right-hand sides of (9) and (10). By standard randomness assumptions, for each $i \in \mathbb{N}$ and $0 \leq m \leq \ell - 1$, the probability that

$X = X[2r\ell \cdot i + 2rm + r]$ satisfies $X_L = X_R$ is $2^{-n/2}$. The same holds for the probability that $X' = X[2r\ell \cdot i + 2rm]$ satisfies $X'_L = X'_R$. Hence, it is expected that the smallest index for which (8) or the respective statement for X' holds is $X[j]$, for $j = O(r \cdot 2^{n/2})$. Therefore, in expectation (9) and (10) yield

$$E\left[CycleLength_{E_k}(P_0)\right] \leq O\left(\frac{r \cdot 2^{n/2} \cdot 2 \cdot \frac{\ell}{\gcd(2m+1,\ell)}}{2r\ell}\right) \leq O(2^{n/2}). \qquad (11)$$

It follows that when we apply the attack algorithm described above to E_k, the expected data complexity is as small as $O(2^{n/2})$. In Sect. 5, we apply this attack to a 128-bit key variant of the block cipher GOST, to get an attack which is 2^{51} times faster than the best previously known attack of [14] on an even weaker variant.

It should be noted that as $CycleLength_{E_k}(P_0)$ becomes smaller, the task of breaking f_k using the candidate slid pairs becomes more complex, since the cycle contains at most $O(2^{n/2})$ known input/output pairs for f_k. Combining many cycles together is problematic since each candidate slid pair is correct only with probability $\varphi(\ell)/\ell$, so the probability that all examined cycles suggest correct slid pairs may be too low. A possible way to overcome this obstacle is to devise an efficient *distinguishing* attack on f_k that requires at most $O(2^{n/2})$ known input/output pairs. Using the distinguishing attack, the adversary can check which of the suggested slid pairs are correct ones and then combine verified slid pairs suggested by different cycles to break f_k with a more data-consuming attack.

5. Application to the GOST Block Cipher

In this section we apply the new slide attacks presented in Sect. 4 to variants of the block cipher GOST—the Russian encryption standard.[3] First we present the structure of GOST and a brief account of previous results on the cipher. Then we present new efficient attacks on 8-round GOST with unknown S-boxes that will be used as subroutines in the slide attacks. Finally, we present the new slide attacks on 24-round GOST and on a 128-bit key variant of the cipher considered in [11,14].

5.1. *The Structure of GOST and Previous Results*

The Russian encryption standard GOST [29], designed in 1989, is a 64-bit block and 256-bit key cipher with a Feistel structure of 32 rounds. The round function accepts an input and a subkey of 32 bits each. The input and the subkey are added (modulo 2^{32}), and the outcome is divided into eight sets of four bits each. Each such set enters a different S-box, where the least significant set enters $S1$ and the most significant set enters $S8$. A central feature of GOST is that the actual S-boxes that are used are kept secret and are

[3] We note that GOST has been the Russian encryption standard since 1989. In 2015, the Russian Federation standardization body issued another standard called Kuznyechik, to be effective in parallel with GOST (which is now called MAGMA in the official documents). Hence, currently GOST is one of the two Russian standards for block ciphers.

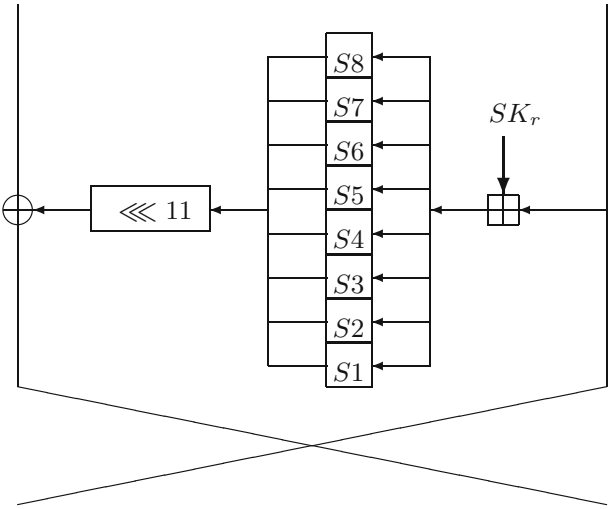


Fig. 4. Round function of GOST.

assigned by the government to a given set of users in each industry.[4] The outputs of all S-boxes are combined to 32 bits, which are then rotated to the left by 11 bits. Figure 4 describes one round of GOST.

The key schedule algorithm takes the 256-bit key and treats it as eight 32-bit words, i.e., $K = K_1, \ldots, K_8$. The subkey k_r of round r is

$$k_r = \begin{cases} K_{(r-1) \bmod 8+1} & r \in \{1, \ldots, 24\}; \\ K_{33-r} & r \in \{25, \ldots, 32\}. \end{cases}$$

Most of the previous results on GOST consider a variant with *known* S-boxes, using as a specific example the set of S-boxes used in the Russian banking industry that has leaked and appears in [42]. In this mode, Isobe [31] was the first to publish an attack on the full cipher in 2010. Later on, several improved attack were presented by Dinur et al. [19], and a multitude of attacks using various techniques were presented by Courtois [12,13,15].

The results obtained so far on GOST with unknown S-boxes are by far less impressive. A related-key distinguisher for the entire cipher that requires only two related-key chosen plaintexts is presented in [37]. In [11,21] a slide-with-a-twist attack on a 20-round variant of GOST is described, and in [18] an efficient enhanced reflection attack on 18-round GOST is given. Finally, a weak-key class consisting of 2^{32} keys of the form (K, K, K, K, K, K, K, K) identified using a slide attack is presented in [25], and in [41] it is shown that for a key in this weak-key class, an adversary can recover the secret S-boxes with a chosen-key attack. To summarize, the only known attacks on more than 20 rounds of GOST with secret S-boxes are either related-key attacks or attacks that apply for only 2^{32} of the 2^{256} keys.

[4] We note that according to the published documentation [29], the S-boxes are not necessarily permutations. Hence, an S-box can be modeled as an unknown 64-bit key.

In our attacks on variants of GOST presented below, we use the following notations. The plaintext and the ciphertext are denoted, as usual, by P, C, respectively. The input to round i is denoted by X_i, so $P = X_1$. Note that following the official GOST specifications, we start the rounds at round 1. For any state X, the left and right halves of X are denoted by X_L and X_R, respectively. Bits $j, j+1, \ldots, \ell$ of a state X are denoted by $X[j–\ell]$. Finally, a difference between two encryptions in the intermediate state X is denoted by X'.

5.2. *Efficient Known-Plaintext Attacks on 8-Round GOST with Unknown S-Boxes*

We now present two new attacks on 8-round GOST with unknown S-boxes. The first is a simple distinguishing attack with data complexity of 2^{36} known plaintexts. The second attack is more complex, but allows to recover the unknown S-boxes and the secret key with data complexity of $2^{36.5}$ known plaintexts and time complexity of $2^{36.5}$. Both attacks will be used as subroutines in the slide attacks presented in Sects. 5.3 and 5.4. (For reasons described below, both are needed.)

5.2.1. *A Distinguishing Attack on 8-Round GOST with Unknown S-Boxes*

The distinguishing attack is a simple truncated differential attack which uses the 8-round differential characteristic depicted in Fig. 5. The characteristic exploits the weak diffusion of the GOST round function to predict the difference in 17 bits of the state after 8 rounds with probability $(1/2)^8 \cdot (3/4)^8 = 2^{-11.32}$. As can be seen, for each S-box whose input difference is $L_i \in \{0, 1, \ldots, 7\}$, with probability of at least $3/4$, the key addition operation does not result in carry to the next S-box (and so, the next S-box remains inactive[5]), and with probability of $1/2$, the LSB of the output difference is zero, so that after the rotation, only one S-box is affected and not two. In the last three rounds of the characteristic, we allow some of the active S-boxes to affect two S-boxes in the next round in order to increase the probability of the characteristic.

The differential characteristic allows to mount a distinguishing attack on 8-round GOST with data complexity of 2^{37} known plaintexts. Indeed, the plaintexts contain $2^{2 \cdot 37-1} = 2^{73}$ pairs, out of which $2^{73} \cdot 7 \cdot 2^{-64} \approx 2^{11.8}$ pairs satisfy the input difference of the characteristic. Hence, it is expected that for 8-round GOST, at least one of them satisfies also the output difference of the characteristic. On the other hand, for a random permutation, the probability that the output difference of the characteristic is obtained for a given pair is 2^{-17}, and thus with a very high probability there are no pairs that satisfy the characteristic's input and output differences. This allows to distinguish 8-round GOST from a random permutation with a high probability.

The data complexity of the attack can be slightly reduced to 2^{36} known plaintexts, by using variants of the differential characteristic in which the input difference is rotated cyclically by k nibbles (for $k = 1, 2, \ldots, 7$). It can be checked that each of these characteristics holds with probability of at least $2^{-12.57}$, and so, 2^{71} pairs (formed from 2^{36} plaintexts) are sufficient to have a right pair with respect to one of the 8 characteristics

[5] Obviously when we discuss the most significant S-box, there is no such carry with probability 1.

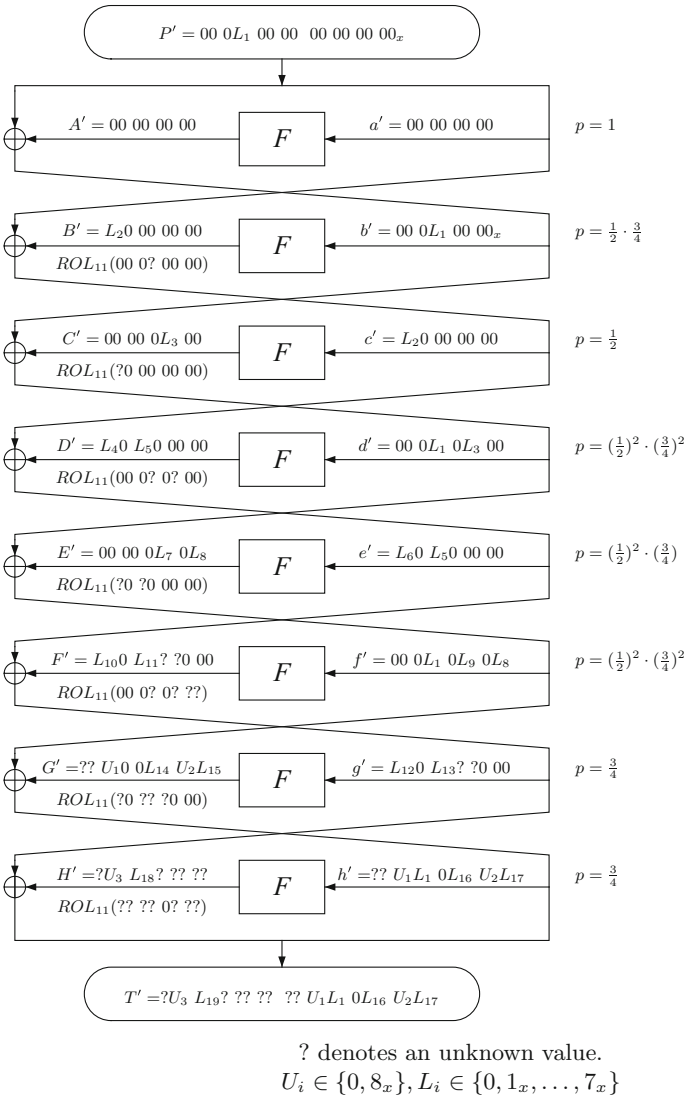
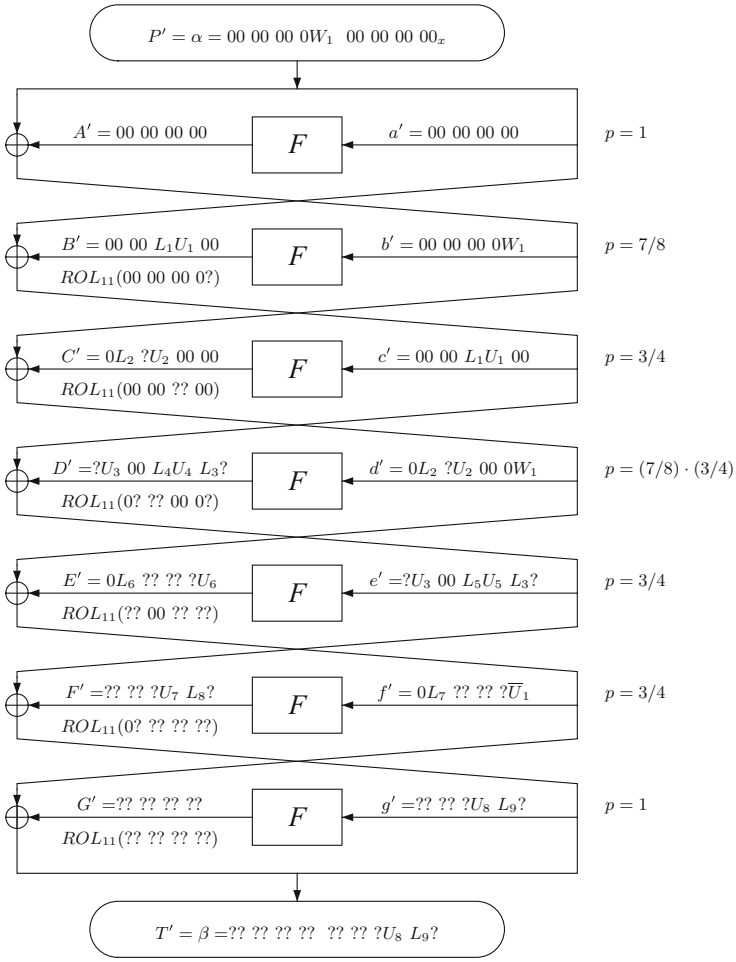


Fig. 5. An 8-round truncated differential of GOST with probability $(\frac{1}{2})^8 \cdot (\frac{3}{4})^8 = 2^{-11.32}$.

with a high probability.[6] Finally, the attack can be performed efficiently by inserting all plaintexts into a hash table indexed by the right half of the input and the 8 MSBs of the nibbles in the left half of the input (i.e., a total of 40 bits), and checking only colliding pairs in the table as candidates for satisfying the characteristic. In this way, the time complexity is dominated by encrypting the plaintexts.

[6] Using standard independence assumption and the Poisson distribution of right pairs, the success probability is 71.8%.



? denotes an unknown value.

$U_i \in \{0, 8_x\}, L_i \in \{0, 1_x, \ldots, 7_x\}, W_1 \in \{1_x, 2_x, 3_x\}, \overline{U}_1 \in \{1_x, 2_x, 3_x, 9_x, A_x, B_x\}$

Fig. 6. A 7-round differential of GOST with probability $(\frac{7}{8})^2 \cdot (\frac{3}{4})^4 = 0.242$.

5.2.2. *A 7-Round Truncated Differential and a Key Recovery Attack on 8-Round GOST with Unknown S-Boxes*

The basic 7-round truncated differential we use is depicted in Fig. 6. It exploits the weak diffusion of the GOST round function to predict the difference in 4 bits of the intermediate state after 7 rounds with probability 0.242.

As in the 8-round differential presented above, we require that in all "active" S-boxes (i.e., S-boxes with a nonzero input difference), the key addition does not result in carry into the next S-box. The probability of this event is either $7/8$ (if the difference in the S-box is W_1) or $3/4$ (if the difference is L_i). This time, we always allow the output difference to affect two S-boxes in order to maximize the probability of the characteristic.

We use the differential in rounds 2–8 of the encryption process and analyze the first round. Denote its input difference by α, i.e., we look for pairs with difference $X'_2 = \alpha$. For such pairs, the input difference of the round function in round 1 is $P'_{1L} = 00\ 00\ 00\ 0W_1$. With probability $7/8$, this implies that only one S-box in round 1 is active and the output difference is of the form $00\ 00\ L_1U_1\ 00$. If this is the case, then there are only 16 possible plaintext differences (which are all P' such that $P'_R[0\text{–}10,15\text{–}31] = 0$) that lead to difference α in the input of round 2. Denote the set of these 16 possible differences of P'_R by S.

In the attack, we examine $2^{36.5}$ plaintexts, and out of the 2^{72} possible plaintext pairs that they compose we keep only those that satisfy three conditions:

1. $P'_L = 00\ 00\ 00\ 0W_1$, for $W_1 \in \{1_x, 2_x, 3_x\}$.
2. $P'_R \in S$ (which means that $P'_R[0\text{–}10,15\text{–}31] = 0$).
3. $C' \in \beta$ (which means that $C'[7\text{–}10] = 0000$).

Out of the $2^{36.5}$ known plaintexts, it is possible to compose 2^{72} pairs. The probability that a random pair satisfies the above three conditions is $3/2^{32} \cdot 2^{-28} \cdot 2^{-4} = 3 \cdot 2^{-64}$, and we expect $2^{72} \cdot 3 \cdot 2^{-64} = 768$ random pairs to satisfy all three conditions. Independently, out of the 2^{72} pairs, a fraction of $3 \cdot 2^{-64}$ satisfy the input difference α at the entrance to the second round. Each of these $2^{72} \cdot 3 \cdot 2^{-64} = 768$ pairs has probability 0.242 to satisfy the differential characteristic (i.e., to have the output difference $C' \in \beta$). This suggests that about $768 \cdot 0.242 = 184$ right pairs exist in the data. We note that with probability $7/8$ a pair with input difference α to the second round has only one active S-box in the first round (Condition 2), i.e., we expect about $184 \cdot 7/8 = 161$ right pairs to exist in the data set (along 768 random pairs which satisfy the input/output differences).

Therefore, after the initial filtering we are left with about 160 "right" pairs that satisfy the truncated differential and 768 "random" pairs. We now need to identify the right pairs and use them appropriately.

Observation 1. *Let* (P_1, P_2) *and* (P_1^*, P_2^*) *be two right pairs satisfying Condition 2. If* $P_{1L}[0\text{–}3] = P_{1L}^*[0\text{–}3]$ *(the input bits to the single active S-box in round 1) and* $P_{2L}[0\text{–}3] = P_{2L}^*[0\text{–}3]$ *(which imply that the input differences* P'_L, P'^*_L *are equal), then the output differences of the first round function are equal in both pairs, and thus* $P'_R[11\text{–}14] = P'^*_R[11\text{–}14]$.

This observation can be used to distinguish 8-round GOST from a random permutation, as follows: We divide the ≈ 900 pairs (out of which about 160 are right pairs) that satisfy the three conditions into 48 sets, according to the value of $(P_L[0\text{–}3], P'_L[0, 1])$. (Note that $P'_L[0, 1]$ has only three possible values.) In each set, check how many times each value of $P'_R[11\text{–}14]$ is encountered. For a random cipher (or when the pairs are random), the values of $P'_R[11\text{–}14]$ in each set are distributed uniformly among the 16 possible values. In particular, the probability that in more than a few sets a value of $P'_R[11\text{–}14]$ appears at least four times is fairly low. On the other hand, for 8-round GOST, in each of the 48 sets, all right pairs have the same value of $P'_R[11\text{–}14]$. As each set contains 3.33 right pairs on average, it is expected that at least 10 sets have a value of $P'_R[11\text{–}14]$ appearing at least four times, or at least two sets have a value of $P'_R[11\text{–}14]$ appearing at least five times.

Formally, we define the distinguisher based on whether at least 10 sets have a value of $P'_R[11–14]$ appearing at least four times or at least two sets have a value of $P'_R[11–14]$ appearing at least five times. Otherwise, we decide that the examined cipher is a random permutation. A straightforward computation shows that the success probability of this distinguisher is very high. We experimentally verified this by running 10,000 experiments with right pairs (all of which were identified correctly). We also run 100,000 experiments with a random permutation (all of which were identified correctly as well). The algorithm of the distinguisher is presented below.

1. Ask for the encryption of $2^{36.5}$ known plaintexts.
2. Insert the plaintext/ciphertext pairs into a hash table T_1 indexed by bits $P_L[2–31]$, $P_R[0–10,15–31]$, $C[7–10]$.
3. Collect all pairs colliding in T_1 (all satisfying Conditions (1)–(3)).
4. Insert the colliding pairs into a second hash table, T_2 indexed by the value of $(P_L[0–3], P'_L[0, 1])$.
5. In each entry of T_2, find the maximal number of times that a $P'_R[11–14]$ value appears. Count how many times this maximum is 4 or at least 5. If the number of times 4 appears as the maximum is 10 or more, or if the number of times 5 (or more) appears as the maximum is 2 or more, output "8-round GOST." Otherwise, output "random permutation."

The data and time complexities of the above distinguisher are $2^{36.5}$.

5.2.3. *Recovering the Subkeys and the S-Boxes*

The algorithm described above allows not only to distinguish 8-round GOST from a random permutation, but also to detect the right pairs, and using them, to recover the subkeys and the secret S-boxes. For each of the 48 sets considered above, with a very high probability the right pairs are those which correspond to the most frequently proposed value of $P'_R[11–14]$.

We start the analysis by using the right pairs from each set to determine the least significant nibble of k_1. Right pairs that passed our analysis do not have the second S-box active, and thus, subkey suggestions that cause overflow and activate the second S-box can be discarded. After analyzing all 48 sets, we expect only the correct value of k_1 to remain.

We also note that one can apply the attack from the ciphertext side (by reversing the order of rounds in the differential characteristic and using it in rounds 1–7). This allows finding the least significant nibble of k_8 using the same technique.

At this point, we know various input pairs to $S1$ and their corresponding output differences. While it allows using techniques such as those discussed in [8], a more subtle analysis is needed. The S-box recovery as well as the related subtleties are given in "Appendix".

After recovering the first S-box $S1$, as well as the least significant nibbles of k_1 and k_8, we can repeat the attack targeting $S2$ and the second least significant nibbles. The attack is then repeated until the full k_1 and k_8 as well as the 8 S-boxes are recovered. Extracting the remaining six subkeys $(k_2, k_3, \ldots, k_7)$ can be easily done by truncating the differential and re-running the attack on less rounds using the truncated differential.

Hence, we can fully recover the secret S-boxes and the secret key with data, memory, and time complexity of $2^{36.5}$.

5.3. *A Slide Attack on 24-Round GOST with Unknown S-Boxes*

An efficient slide attack on a reduced round variant of GOST which consists of the first 24 rounds of the cipher can be mounted by combining the generic technique of Sect. 4.1 with the attack on 8-round GOST of Sect. 5.2.2.

Indeed, by the structure of GOST, its first 24 rounds can be written as $E_k = (f_k)^3$, where f_k is 8-round GOST with subkeys $K_1, K_2, \ldots, K_8$. This allows to apply the attack of Sect. 4.1 to retrieve a set of $\approx 2^{63}$ slid pairs, which are input/output pairs to f_k. Given these input/output pairs (and actually, $2^{36.5}$ of them are sufficient), the secret S-boxes and the secret key $K_1, K_2, \ldots, K_8$ can be recovered using the attack of Sect. 5.2.2.

The attack complexity is dominated by the complexity of the generic attack of Sect. 4.1, which requires 2^{63} adaptively chosen plaintexts (or almost 2^{64} known plaintexts) and has time complexity of 2^{63}. The success probability is close to 1 as if the attack fails for one cycle of E_k (which happens with probability $1 - \varphi(3)/3 = 1/3$, as $\ell = 3$ in our case), we can repeat it for other cycles.

5.4. *An Efficient Slide Attack on an 128-Bit Key Variant of GOST*

Consider a weak-key class of GOST of size 2^{128} in which the 256-bit key is *palindromic*, i.e., has the form $K_1, K_2, K_3, K_4, K_4, K_3, K_2, K_1$ where K_1, K_2, K_3, K_4 are 32-bit words. This weak-key class was mentioned in [11], where it was remarked that under such keys GOST is an involution, and thus, has 2^{32} fixed points. However, this is clearly not sufficient for attacking it efficiently. In [14], Courtois analyzed two natural 128-bit key variants of GOST. One of these variants is our "palindromic key schedule" variant, and for this variant Courtois presents a fixed-point attack which requires 2^{32} chosen plaintexts and 2^{91} encryptions, assuming that the S-boxes are known. We present an efficient slide attack on this variant, which has a practical complexity of about 2^{40}, even if the S-boxes are secret.

For keys of the form $K_1, K_2, K_3, K_4, K_4, K_3, K_2, K_1$, the full 32-round GOST can be written as $E_k = f_k^4$, where f_k is 8-round GOST with unknown S-boxes. Furthermore, f_k is a Feistel construction with a palindromic key schedule, and thus, we can apply to E_k the efficient slide attack of Sect. 4.2. In this case, we have $n = 64$, $\ell = 4$, and $r = 4$, and thus, the expected length of $Cycle_{E_k}(P_0)$ is 2^{31}. By the attack of Sect. 4.2, this would be sufficient to break E_k with 2^{31} plaintexts and time, could we break 8-round GOST with 2^{31} known plaintexts. As our best attack on 8-round GOST requires $2^{36.5}$ known plaintexts, we have to work a little more.

In order to collect $2^{36.5}$ plaintexts, we must use plaintexts from many different cycles together. However, for each of the cycles, the probability that the candidate pairs yielded by it are correct slid pairs is $\varphi(4)/4 = 1/2$. Hence, the probability that all pairs we collect are *simultaneously* correct is extremely low. To overcome this problem, we first use the distinguishing attack of Sect. 5.2.1 to detect the cycles that yield correct slid pairs and then apply the key recovery attack of Sect. 5.2.2 to the "correct" pairs.

Specifically, we start with 2^{40} adaptive plaintexts (collected from ≈ 512 different cycles) and detect all plaintext pairs that satisfy the truncated differential presented in Sect. 5.2.1. As described above, the pairs can be detected efficiently (i.e., in time that is negligible compared to the encryption of the plaintexts) using a hash table. Among the 2^{79} plaintext pairs, about 2^{77} result from correct slid pairs (as this happens exactly when both plaintexts belong to cycles that yield correct slid pairs). Hence, about 64 pairs are expected to satisfy the truncated differential characteristic . In addition, it is expected that $2^{79} \cdot 2^{-64} \cdot 2^{-17} \cdot 7 \cdot 8 \approx 16$ pairs satisfy the input and output differences of the characteristics by chance. Despite the "false alarms," we treat all found pairs as "right" pairs, and for each of them, we assume that both cycles from which the elements of the cycle were taken contain "correct" slid pairs. Thus, we obtain about 2^7 "correct" cycles, which are used in the subsequent key recovery attack.

In the key recovery attack, we are given about $2^{31} \cdot 2^7 = 2^{38}$ plaintexts, that form 2^{75} pairs. However, out of these pairs only $2^{75} \cdot (4/5)^2 \approx 2^{74.31}$ are real plaintext/ciphertext pairs, while the rest are "random" pairs taken from the "false alarm" cycles. As a result, the signal/noise ratio of the truncated differential attack presented in Sect. 5.2.2 is reduced by a factor of $(4/5)^2$. However, it can be easily seen that as the total number of plaintexts is increased by a factor of 2, the attack still succeeds to recover the key and the secret S-boxes with a high probability.

To summarize, we obtain an attack on the "palindromic key schedule" variant of GOST with data, memory, and time complexity of 2^{40}. It seems plausible that the complexity of the attack can be reduced even further (say, to about 2^{35}), by devising an attack on 8-round GOST with a smaller data complexity.

6. Summary and Conclusions

In this paper, we presented four types of new efficient slide attacks, that perform better than the "standard" 2^n time complexity of the slide attack, due to efficient detection of the slid pairs. We applied our attacks to two variants of the Russian encryption standard GOST, improving the total complexity over the best previously known attacks on the same (or weaker) variants by a factor of more than 2^{35}. One of our attacks can be used practically by a legitimate user of GOST who wants to recover her secret S-boxes (that are supposed to be used in other branches of the same industry).

It will be interesting to find further scenarios in which slide attacks can be performed with time complexity of less than 2^n, and thus, make slide attacks more applicable in practice.

Acknowledgements

The authors are grateful to Itai Dinur and Adi Shamir for numerous fruitful discussions. The authors would also like to thank Nicolas Courtois for his useful comments on a draft of the paper. We also would like to thank Erkan Uslu for his comment on this manuscript. The help of the anonymous referees in improving the attacks of this paper is highly appreciated. The first author was partially supported by the Israeli Ministry

of Science and Technology and by the Check Point Institute for Information Security, and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. The third author was supported in part by the Israel Science Foundation through grants No. 827/12 and No. 1910/12 and in part by the European Commission under Contract ICT-645622 PQCrypto. The fourth author was supported by the Alon Fellowship.

Appendix: S-Box Recovery

In this appendix, we describe the S-box recovery part of our attack on 8-round GOST with unknown S-boxes. To the best of our knowledge, the problem of recovering an S-box from its difference distribution table (or parts of it) was not fully addressed in the literature. In a related manuscript [20], we discuss the problem and analyze various strategies. Here we present a simple algorithm which is sufficient for the purposes of our current attack.

In the settings discussed in this paper, the adversary has access to only several rows of the difference distribution table. Recall that the slide attack allows obtaining inputs to some S-box, e.g., $S1$, followed by their output difference. One thing which affects our analysis is the fact that though we obtain all different inputs, the pairs of which we know the output difference have input difference of either 1,2, or 3, up to the effects of the modular key addition. When the key nibble corresponding to the S-box we analyze is nonzero, other input differences to the S-box may occur (due to the carry chain). However, even in these cases, we obtain only a partial view of the difference distribution table (we later discuss methods to obtain the missing bits and pieces).

We shall analyze the "worst" case from our perspective, namely when the respective key nibble is all zero. In such a case, all input pairs are indeed with difference 1,2, or 3. This means that each four values of $S[4i]$, $S[4i+1]$, $S[4i+2]$, and $S[4i+3]$ can be determined independently for $i = 0, \ldots, 3$. Moreover, due to the nature of the information we have, one can only find the candidate values for such a quartet, but cannot determine which quartet of values obtains which solution.

Our algorithm is quite straightforward. We first determine $S[0]$ to be x (any value will be acceptable, as the output differences are preserved under XORing the same value to all outputs of $S[0]$), and try all possible $S[1]$ that are plausible (namely, for which the output difference $S[0] \oplus S[1]$ is possible given an input difference 1_x). For each such possibility, we try all possible values of $S[2]$ that are plausible with respect to $S[0]$ and then continue to verify that the difference $S[1] \oplus S[2]$ may occur for an input difference $1 \oplus 2 = 3$. Similar considerations are applied on $S[3]$. Also, we can check that the determined values do not offer entries that contradict the known values of the difference distribution table (i.e., offer values higher than those in the actual difference distribution table).

After computing the first four values, we can continue. In case the two least significant bits of the key nibble are zero, we must try all possible remaining values for some entry (otherwise, we can use the difference distribution table to determine some other S-box entry). We continue this process, until the full S-box is recovered. We then verify the correctness of the S-box by checking the rows of the difference distribution table which are available to the adversary.

Our experiments with 32 different S-boxes (the 8 S-boxes of GOST used in the Russian banking industry, the 8 S-boxes suggested in the revised GOST R 34.12-2015 standard, the 8 S-boxes of the block cipher Serpent, and their 8 inverses) show that given the correct value of $S[0]$, the information from the difference distribution table corresponding to input differences 1, 2, and 3 is sufficient to reduce the number of candidates to between 147456 and 2359296 (which is equivalent to a total of between $2359296 = 2^{21.2}$ and $37748736 = 2^{25.2}$ possible S-boxes). The process of reconstructing all possible S-boxes given a single difference distribution table takes about 1.31 seconds on average, using a C program compiled with gcc-4.8.4 with the optimization flag "-O2" on an Intel Quad-Core i7-5500U CPU running at 2.40 GHz.

To further reduce the number of possible S-boxes, one needs to obtain more rows of the difference distribution table. When the two least significant bits of the key nibble are nonzero, one can obtain additional rows using carries (and use them to filter wrong S-box candidates). In the other cases, one can consider pairs that satisfy rotated versions of the 8-round differential characteristic depicted in Fig. 5 such that the eighth round has a nonzero input to the S-box, with a known output difference. This allows reconstructing the missing rows in the difference distribution table.

Using this additional knowledge, we were able to reduce the number of candidate S-boxes to between 144 and 352 (including the uncertainty concerning the value of $S[0]$). The average time complexity of identifying these candidates is less than 1.35 seconds on the same machine (i.e., it takes a fraction of a second more to reconstruct the difference distribution tables and check them).

Once the first S-box is recovered, one can continue and recover other S-boxes, or analyze other rounds. For example, after recovering the S-boxes $S1$, $S6$, and $S7$, it is possible to target the value of the first nibble of k_2 and/or of k_7. If a guess of these S-boxes does not offer a consistent suggestion to these key nibbles, then it can be discarded. This way we can determine step by step more S-boxes and key nibbles, until only a single consistent solution remains.

The time complexity is mostly determined by the analysis of each S-box independently using the difference distribution table (or its known parts) which takes at most the equivalent of 2^{32} GOST encryptions.

References

[1] W. Aerts, E. Biham, D.D. Moitie, E.D. Mulder, O. Dunkelman, S. Indesteege, N. Keller, B. Preneel, G.A.E. Vandenbosch, I. Verbauwhede, A practical attack on KeeLoq. *J. Cryptol.* **25**(1), 136–157 (2012)

[2] M.V. Anand, E.E. Targhi, G.N. Tabia, D. Unruh, Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation, in T. Takagi (ed.) *Post-Quantum Cryptography—7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24–26, 2016, Proceedings*. Lecture Notes in Computer Science, vol. 9606 (Springer, Berlin, 2016), pp. 44–63

[3] E. Biham, New types of cryptanalytic attacks using related keys. *J. Cryptol.* **7**(4), 229–246 (1994)

[4] E. Biham, O. Dunkelman, N. Keller, Improved slide attacks, in A. Biryukov (ed.) *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26–28, 2007, Revised Selected Papers*. Lecture Notes in Computer Science, vol. 4593 (Springer, Berlin, 2007), pp. 153–166

[5] E. Biham, A. Shamir, Differential Cryptanalysis of the Data Encryption Standard (Springer, Berlin, 1993)

[6] A. Biryukov, C. Bouillaguet, D. Khovratovich, Cryptographic schemes based on the ASASA structure: black-box, white-box, and public-key (extended abstract), in P. Sarkar, T. Iwata (eds.) *Advances*

in Cryptology—ASIACRYPT 2014—20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873 (Springer, Berlin, 2014), pp. 63–84

[7] A. Biryukov, D. Khovratovich, L. Perrin, Multiset-Algebraic Cryptanalysis of Reduced Kuznyechik, Khazad, and secret SPNs. IACR Trans. *Symmetric Cryptol.* **2016**(2), 226–247 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/572

[8] A. Biryukov, L. Perrin, On reverse-engineering S-boxes with hidden design criteria or structure, in R. Gennaro, M. Robshaw, (eds.) *Advances in Cryptology—CRYPTO 2015—35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2015, Proceedings, Part I*. Lecture Notes in Computer Science, vol. 9215 (Springer, Berlin, 2015), pp. 116–140

[9] A. Biryukov, A. Shamir, Structural Cryptanalysis of SASAS. *J. Cryptol.* **23**(4), 505–518 (2010)

[10] A. Biryukov, D. Wagner, Slide attacks. in L.R. Knudsen, (ed.) *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24–26, 1999, Proceedings*. Lecture Notes in Computer Science, vol. 1636 (Springer, Berlin, 1999), pp. 245–259

[11] A. Biryukov, D. Wagner, Advanced slide attacks, in B. Preneel, (ed.)*Advances in Cryptology—EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14–18, 2000, Proceeding*. Lecture Notes in Computer Science, vol. 1807 (Springer, Berlin, 2000), pp. 589–606

[12] N. Courtois, Algebraic complexity reduction and cryptanalysis of GOST. IACR Cryptology ePrint Archive 2011, 626 (2011). http://eprint.iacr.org/2011/626

[13] N.T. Courtois, An improved differential attack on full GOST. IACR Cryptology ePrint Archive 2012, 138 (2012). http://eprint.iacr.org/2012/138

[14] N.T. Courtois, Cryptanalysis of two GOST variants with 128-bit keys. *Cryptologia* **38**(4), 348–361 (2014)

[15] N.T. Courtois, An improved differential attack on full GOST, in P.Y.A. Ryan, D. Naccache, J. Quisquater, (eds.) *The New Codebreakers—Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. Lecture Notes in Computer Science, vol. 9100 (Springer, Berlin, 2016), pp. 282–303

[16] J. Daemen, L.R. Knudsen, V. Rijmen, The block cipher square, in E. Biham, (ed.) *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20–22, 1997, Proceedings*. Lecture Notes in Computer Science, vol. 1267 (Springer, Berlin, 1997), pp. 149–165

[17] J. Daemen, V. Rijmen, The design of Rijndael: AES—the advanced encryption standard. Information Security and Cryptography (Springer, Berlin, 2002)

[18] I. Dinur, O. Dunkelman, N. Keller, A. Shamir, Reflections on slide with a twist attacks. Des. *Codes Cryptogr.* **77**(2–3), 633–651 (2015)

[19] I. Dinur, O. Dunkelman, A. Shamir, Improved attacks on full GOST, in A. Canteaut, (ed.) *Fast Software Encryption—19th International Workshop, FSE 2012, Washington, DC, USA, March 19–21, 2012. Revised Selected Papers*. Lecture Notes in Computer Science, vol. 7549 (Springer, Berlin, 2012), pp. 9–28

[20] O. Dunkelman, N. Keller, Reverse Engineering the Difference Distribution Table (2016), work in progress

[21] O. Dunkelman, N. Keller, A. Shamir, Slidex attacks on the Even–Mansour encryption scheme. *J. Cryptol.* **28**(1), 1–28 (2015)

[22] S. Even, Y. Mansour, A construction of a cipher from a single pseudorandom permutation. *J. Cryptol.* **10**(3), 151–162 (1997)

[23] P. Flajolet, R. Sedgewick, Analytic Combinatorics (Cambridge University Press, Cambridge, 2009). http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521898065

[24] R.W. Floyd, Nondeterministic Algorithms. *J. ACM* **14**(4), 636–644 (1967)

[25] S. Furuya, Slide attacks with a known-plaintext cryptanalysis, in K. Kim, (ed.) *Information Security and Cryptology—ICISC 2001, 4th International Conference Seoul, Korea, December 6–7, 2001, Proceedings*. Lecture Notes in Computer Science, vol. 2288 (Springer, Berlin, 2001), pp. 214–225

[26] V. Goncharov, On the distribution of cycles in permutations. *Doklady Akedmii Nauk SSSR* **35**, 299–301 (1942)

[27] V. Goncharov, Some facts from combinatorics. *Izvestia Academii Nauk SSSR* **8** , 3–48 (1944), ser. Mat.

[28] M. Gorski, S. Lucks, T. Peyrin, Slide attacks on a class of hash functions, in J. Pieprzyk, (ed.) *Advances in Cryptology—ASIACRYPT 2008, 14th International Conference on the Theory and Application of*

Cryptology and Information Security, Melbourne, Australia, December 7–11, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5350 (Springer, Berlin, 2008), pp. 143–160
[29] Government Committee of the USSR for Standards, Gosudarstvennei Standard 28147-89: Cryptographic Protection for Data Processing Systems. Tech. rep. (1989)
[30] E.K. Grossman, B. Tucherman, Analysis of a weakened Feistel-like Cipher, in *Proceedings of International Conference on Communications* (1978), pp. 46.3.1–46.3.5
[31] T. Isobe, A single-key attack on the full GOST block cipher. *J. Cryptol.* **26**(1), 172–189 (2013)
[32] T. Isobe, K. Shibutani, Generic key recovery attack on Feistel scheme, in K. Sako, P. Sarkar, (eds.) *Advances in Cryptology—ASIACRYPT 2013—19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1–5, 2013, Proceedings, Part I*. Lecture Notes in Computer Science, vol. 8269 (Springer, Berlin, 2013), pp. 464–485
[33] J. Jean, I. Nikolic, T. Peyrin, L. Wang, S. Wu, Security analysis of PRINCE, in S. Moriai (ed.) *Fast Software Encryption—20th International Workshop, FSE 2013, Singapore, March 11–13, 2013. Revised Selected Papers*. Lecture Notes in Computer Science, vol. 8424 (Springer, Berlin, 2013), pp. 92–111
[34] M. Kaplan, G. Leurent, A. Leverrier, M. Naya-Plasencia, Breaking symmetric cryptosystems using quantum period finding, in M. Robshaw, J. Katz, (eds.) *Advances in Cryptology—CRYPTO 2016—36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14–18, 2016, Proceedings, Part II*. Lecture Notes in Computer Science, vol. 9815 (Springer, Berlin, 2016), pp. 207–237
[35] O. Kara, Reflection attacks on product ciphers. IACR Cryptology ePrint Archive 2007, 43 (2007). http://eprint.iacr.org/2007/043
[36] O. Kara, Reflection cryptanalysis of some ciphers, in D.R. Chowdhury, V. Rijmen, A. Das, (eds.) *Progress in Cryptology—INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14–17, 2008. Proceedings*. Lecture Notes in Computer Science, vol. 5365 (Springer, Berlin, 2008), pp. 294–307
[37] Y. Ko, S. Hong, W. Lee, S. Lee, J. Kang, Related key differential attacks on 27 rounds of XTEA and full-round GOST, in B.K. Roy, W. Meier, (eds.) *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5–7, 2004, Revised Papers*. Lecture Notes in Computer Science, vol. 3017 (Springer, Berlin, 2004), pp. 299–316
[38] M. Matsui, Linear cryptanalysis method for DES cipher, in T. Helleseth, (ed.) *Advances in Cryptology—EUROCRYPT '93, Workshop on the Theory and Application of Cryptographic Techniques, Lofthus, Norway, May 23–27, 1993, Proceedings*. Lecture Notes in Computer Science, vol. 765 (Springer, Berlin, 1993), pp. 386–397
[39] B. Minaud, P. Derbez, P. Fouque, P. Karpman, Key-recovery attacks on ASASA, in T. Iwata, J.H Cheon, (eds.) *Advances in Cryptology—ASIACRYPT 2015—21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II*. Lecture Notes in Computer Science, vol. 9453 (Springer, Berlin, 2015), pp. 3–27
[40] National Bureau of Standards: Data Encryption Standard. NBS Federal Information Processing Standard (FIPS) 46 (1977)
[41] M.J Saarinen, A chosen key attack against the secret S-boxes of GOST (1998). http://citeseer.ist.psu.edu/saarinen98chosen.html
[42] B. Schneier, *Applied Cryptography* 2nd edn (Wiley, New York, 1996)
[43] D.R. Simon, On the power of quantum computation. *SIAM J. Comput.* **26**(5), 1474–1483 (1997)
[44] T. Tiessen, L.R. Knudsen, S. Kölbl, M.M. Lauridsen, Security of the AES with a secret S-Box. in G. Leander, (ed.)*Fast Software Encryption—22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8–11, 2015, Revised Selected Papers*. Lecture Notes in Computer Science, vol. 9054 (Springer, Berlin, 2015), pp. 175–189