# Analysis of Involutional Ciphers: Khazad and Anubis*

Alex Biryukov

Katholieke Universiteit Leuven, Dept. ESAT/COSIC, Leuven, Belgium
abiryuko@esat.kuleuven.ac.be

**Abstract.** In this paper we study structural properties of SPN ciphers in which both the S-boxes and the affine layers are involutions. We apply our observations to the recently designed Rijndael-like ciphers Khazad and Anubis, and show several interesting properties of these ciphers. We also show that 5-round Khazad has $2^{64}$ weak keys under a "slide-with-a-twist" attack distinguisher. This is the first cryptanalytic result which is better than exhaustive search for 5-round Khazad. Analysis presented in this paper is generic and applies to a large class of ciphers built from involutional components.

## 1 Introduction

Ciphers constructed from a product of involutions are not new in cryptography. In fact one of the most popular constructions – the Feistel construction is a product of two involutions: XOR of the two halves of the block and swap of the two halves of the block. However SPNs resulting from a product of involutions (i.e. both the non-linear S-box layer and the affine layer are involutions) were not intensively studied. Recently two ciphers with this new property have been designed by Barreto and Rijmen and submitted to the European prestandardization project NESSIE [6]. These ciphers are the 64-bit 8-round SPN cipher Khazad [1] and the 128-bit 12-18-round cipher Anubis [2]. Both ciphers have Rijndael-like structure [4]. Khazad uses an MDS diffusion layer which provides complete diffusion after one round (branch number is 9), while Anubis has a slower, Rijndael-like diffusion. In both cases linear transformations of the two ciphers are chosen to be involutions. The same is true for the S-boxes: Anubis and Khazad share the same 8x8 bit involutional S-box which is constructed from three layers of smaller 4x4 bit involutional S-boxes. Motivations behind such extensive use of involutional components is twofold: efficient implementation and equal security of encryption and decryption. The only difference between encryption and decryption of these ciphers is in the inverse key-schedule.

In this paper we study the structure of a generic involutional SPN on an example of Khazad. We show several interesting properties of Khazad and Anubis

* The work described in this paper has been supported in part by the Commission of the European Communities through the IST Programme under Contract IST-1999-12324 and by the Concerted Research Action (GOA) Mefisto.

which follow from their elegant involutional structure. The analysis that we provide holds for arbitrary S-boxes or affine mixing layers assuming only that these are involutions, and the S-boxes don't have fixed points (which is true both for Khazad and Anubis, since it was one of the design criteria). Whether these properties can be exploited in attacks on round-reduced or full Khazad is a matter of further research. Finally, we use involutional structure of Khazad in order to mount sliding-with-a-twist attack [3] on 5-rounds of this cipher, which works for $2^{64}$ out of $2^{128}$ keys. This attack might be of independent interest, since it may be applied to any cipher with the structure $P \circ F \circ Q$ , where $F$ is an arbitrary involution (for all the keys or for some of the keys), and $P, Q$ are arbitrary keyed bijections.

## 2 Previous Results for Khazad and Anubis

The best attack so far on Khazad is the very efficient square attack on 3-rounds ($2^9$ chosen plaintexts and $2^{16}$ S-box lookups). Extended by 64-bit subkey guessing one gets an attack on 4-round Khazad ($2^{80}$ S-box lookups). Gilbert-Minier's collision attack [5] which worked better than the square attack for Rijndael, will not work for Khazad since it will require full 64-bit block collisions which may happen only for equal inputs (in Rijndael one could provide partial 4-byte collisions, due to slower mixing). The designers claim that truncated differentials attacks can't be mounted for 4 or more rounds. No weak keys are known.

Many attacks that were devised against Rijndael can be applied to Anubis with similar results. The best attack is the Gilbert-Minier attack which breaks 7-round Anubis with $2^{32}$ chosen plaintexts and $2^{140}$ complexity of analysis. The square attack against 7-rounds has data complexity $2^{128}-2^{119}$ chosen plaintexts, and the analysis complexity $2^{120}$ steps. Anubis has more rounds than Rijndael, in order to increase its security margin.

## 3 Some Properties of Khazad

As mentioned before we will study the involutional properties on an example of Khazad, however we will not use any specific features of its S-boxes or mixing layers and thus all the following discussion will apply to a generic involutional SPN cipher (with involutional S-boxes without fixed points) and in particular will be applicable to the Anubis block cipher. We refer the reader to [1, 2] for detailed descriptions of Khazad and Anubis.

Throughout this paper we will use the following notation: S-box layers will be denoted by $S$, linear diffusion layers by $M$ (multiplication by an MDS matrix in Khazad) and key-addition by $k$ (sometimes with a subscript to enumerate different round subkeys). Since linear layer and key addition layer commute we can write:

$$M(x) \oplus k = M(x) \oplus M(k') = M(x \oplus k'),$$

where $k'$ is a version of the key $k$ transformed by the linear layer $M$. This gives us an equivalent representation of Khazad in which the unknown subkeys are grouped around the odd S-box layers and the $M$ operations are grouped around the even S-box layers.

Thus, for the 5-round Khazad, which initially can be presented like this (from the left to the right):

$$k_0 S M k_1 S M k_2 S M k_3 S M k_4 S k_5,$$

we will arrive at the following structure:

$$k_0 S k_1' [MSM] k_2 S k_3' [MSM] k_4 S k_5.$$

In this equivalent representation the odd round subkeys are the same as in the original scheme and the even-round subkeys must be transformed by the linear diffusion layer $k' = M^{-1}(k) = M(k)$ (recall that the $M$ is an involution). This new representation has several interesting properties.

**Definition 1** A **cycle type** of a permutation $A$ is a multiset containing the cycle lengths of the permutation together with their multiplicities – the number of times each cycle length occurs in the permutation. We will denote the cycle type of $A$ by $C(A)$.

For example a permutation $(1, 0, 3, 2, 5, 6, 7, 4)$ which can be written as a collection of three cycles $(1, 0), (3, 2)$ and $(5, 6, 7, 4)$ has the cycle type $(2, 2), (4, 1)$ (i.e. two cycles of length two, and one cycle of length 4). First of all, the $MSM$ structure is a fixed permutation that covers 1.5 rounds of a cipher and is known to the attacker. Moreover, since $M$ is an involution we have $MSM = MSM^{-1} = A$ which means that $A$ is a permutation isomorphic to $S$, and since $S$ in Khazad is an involution so is the permutation $A$, i.e. $(C(A) = C(MSM) = C(S))$[1]. In the following subsections we will study the properties of the $kSk$ and $MSM$ permutations.

### 3.1 Properties of the $k_1 S k_2$ Structure

In this subsection we study the function: $k_1 S k_2(x) = S(k_1 \oplus x) \oplus k_2$, for arbitrary 64-bit keys $k_1, k_2$ and a layer of eight S-boxes $S$.

**Theorem 1** The cycle structure of $k_1 S k_2$ depends only on $k_1 \oplus k_2$. Moreover each cycle length appears in the permutation $k_1 S k_2$ an even number of times.

**Proof**

Let us denote a permutation caused by XOR of $k_1 \oplus k_2$ as $\triangle_{k_1 k2}$. Then we can write: $k_1 S k_2(x) = S(k_1 \oplus x) \oplus k_2$ as

$$(S(k_1 \oplus x) \oplus k_1) \oplus (k_1 \oplus k_2) = \triangle_{k_1 k_2} \circ k_1 S k_1 = (S((k_1 \oplus k_2) \oplus k_2 \oplus x) \oplus k_2) = k_2 S k_2 \circ \triangle_{k_1 k_2}.$$

[1] The tweaked version of Khazad has a very structured S-box, thus $A$ is isomorphic to the $[QP]$ structure inside the S-box. This doesn't add information in terms of the cycle type of $A$ but may be helpful if we will find interesting properties of the isomorphism itself.

Both $k_1Sk_1$, and $k_2Sk_2$ are involutions. Since $k_1Sk_2$ is a product of two involutions without fixed points (in case $S$ has no fixed points and unless $k_1 = k_2$) it consists of cycles of the same length in even numbers (this fact was central to the cryptanalysis of the Enigma cipher [7]). □

The $k_1Sk_2$ layer is a parallel application of eight 8-bit permutations, each following the theorem above. Let us denote the number of cycles of the $i^{th}$ permutation by $2n_i$, then we can write the following corollary:

**Corollary 1** The permutation $k_1Sk_2$ consists of at least $2^8 \prod_i n_i$ cycles. The largest cycle size is smaller than $2^{56}$ elements. Cycle lengths are 128-smooth numbers.

**Proof**

Writing as an 8-component vector the initial point $x = (x_1, x_2, \dots, x_8)$, the cycle produced by iterations of $k_1Sk_2$ will have the length which is a LCM of lengths of individual components cycles starting at $x_i$. Since maximal cycle length for each component is 128 the corollary follows (the upper bound is reached if all cycle lengths are relatively prime)[2]. □

Moreover let us pick two points $x = (x_1, x_2, \dots, x_8)$ and $y = (y_1, y_2, \dots, y_8)$ at random. The GCD of the cycle sizes defined by the two initial points $x$ and $y$ has good chances to be quite large (which is unlikely for a random permutation). The probability that components $x_i$'s and $y_i$'s will belong to the same cycle or to the two different cycles of the same size is more than $\frac{2l_i}{256}$, where $2l_i$ is the size of the cycle starting at $x_i$. By collecting the cycle lengths and studying the factors we obtain initial information about the $\triangle_{k_1k_2}$, since each $\triangle_{k_1k_2}$ defines (but not uniquely) its cycle pattern.

### 3.2 Properties of the $[MSM]k_1Sk_2[MSM]$ Structure

Now let us look at the following piece of the Khazad's structure:

$$[MSM]k_1Sk_2[MSM] = Ak_1Sk_2A = B.$$

This piece covers 3.5 rounds of Khazad. Since $A$ is an involution $k_1Sk_2$ and $B$ are isomorphic permutations. Thus $B$ has all the nice cycle structure described in the Corollary 1, in the previous subsection.

**Claim 1** For a randomly chosen pair of keys $k_1, k_2$ the structure $Ak_1Sk_2A$ has no fixed points[3] with probability larger than $1 - 2^{-8}$ . However if it has at least a single fixed point then at once it must have more than $2^8$ fixed points[4].

---

[2] If a point belongs to cycles of lengths $(l_1, \dots, l_8)$, then in a 'product' permutation it belongs to a cycle of length $lcm(l_1, \dots, l_8)$, and $\frac{\prod l_i}{lcm(l_1,\dots,l_8)}$ different cycles of the same length are generated.

[3] A random permutation has no fixed points with probability $\approx 1/e$.

[4] For a random permutation the average number of fixed points is $e^{-1} \sum_{i=1}^{N} \frac{1}{(i-1)!} \approx 1$.

**Proof**

Due to isomorphism of $Ak_1Sk_2A$ and $k_1Sk_2$ under an involution $A$ there is a 1-1 correspondence between the fixed points of the two functions. Similarly it is enough to study fixed points of the function $S(x) \oplus \triangle_k$, where $\triangle_k = k_1 \oplus k_2$. The question is how many solutions there are to the equation $S(x) \oplus x = \triangle_k$? Since $S$ is an involution, running through all possible $x$'s there are at most 128 different solutions for the difference of the keys (for the actual $S$-box there are 102 different solutions, some suggested by 6 different inputs). Thus if $\triangle_k$ is picked at random the chance to be in a "fixed-point" permitting combination is: $(102/256)^8 \approx 2^{-10.6}$. However if the key difference is in the proper combination each S-box has at least two inputs that will preserve the fixed point. Thus the total number of fixed points will be more than $2^8$. $\square$

### 3.3 Properties of the $kSk[MSM]kSk[MSM]kSk$ Structure

The structure $KH_5 = kSk[MSM]kSk[MSM]kSk = k_0Sk_1Ak_2Sk_3Ak_4Sk_5$ is actually a 5-round Khazad, so any non-randomness in it, will be a very interesting property.

For example if $k_1 = k_4$ (a 64-bit restriction on the key-schedule) and then writing $k_5 = k_0 \oplus (k_0 \oplus k_5)$ we obtain permutation isomorphic to just intermediate $k_2Sk_3$ further permuted by a XOR with $(k_0 \oplus k_5)$.

Another interesting property of 5-round Khazad is the following: suppose that the attacker guesses $\triangle_k = k_0 \oplus k_5$, and iterates the 5-round encryption function together with the key-difference XOR after each iteration: $(KH_5(x) \oplus \triangle_k)^n$. It is easy to see that due to cancellations that happen, this is almost equivalent to iterations of a 3-round Khazad. If we denote 3-round Khazad by $KH_3$ we may write

$$(KH_5(x) \oplus \triangle_k)^n = k_0SM(KH_3)^nMSk_0.$$

For example if one can detect any peculiarities in the cycle structure of 3-round Khazad in less than $2^{64}$ steps this property will provide a distinguishing attack on 5-round Khazad faster than the exhaustive key search.

## 4 Sliding Encryption Against Decryption for 5-round Khazad

In this section we will apply advanced sliding technique called slide-with-a-twist [3] to five round Khazad. The idea behind this attack method is to slide encryption against decryption in order to gain additional degree of self-similarity due to increased symmetry.

Consider 5-round Khazad written in equivalent representation and aligned as shown below:

$$P_1 = k_0Sk_1'[Ak_2Sk_3'A]k_4Sk_5 = C_1$$
$$C_2 = k_5Sk_4[Ak_3'Sk_2A]k_1'Sk_0 = P_2.$$

Here, as in the previous sections of this paper we have $k_1' = M(k_1)$, and $k_3' = M(k_3)$, where $k_1$, $k_3$ are the original subkeys of Khazad. The other subkeys $k_0, k_2, k_4, k_5$ are not changed.

If $k_2 = k_3'$, then a piece in the brackets $[Ak_2Sk_3'A]$ is an involution and sliding with a twist applies. We thus can write two very simple equations:

$$S(P_1 \oplus k_0) \oplus k_1' = S(C_2 \oplus k_5) \oplus k_4,$$

$$S(P_2 \oplus k_0) \oplus k_1' = S(C_1 \oplus k_5) \oplus k_4.$$

Although there are many unknowns in these equations: $(k_0, k_1', k_4, k_5)$, there is no diffusion. If we encrypt a pool of texts $P_i, i = 0, \ldots, 2^{32}-1$ with fixed 32-bits, by setting inputs of four out of eight S-boxes to arbitrary constant, a proper slid pair will have 32-bits in common between the plaintexts and as a consequence of the slid-equations it will have 32-bits in the same positions in common between the ciphertexts. Thus instead of checking all the $2^{63}$ possible pairs we can check only $2^{31}$ pairs which satisfy the 32-bit filtration condition. These pairs can be easily identified in a pool of ciphertexts sorted by the positions corresponding to fixed input S-boxes. We will call the four S-boxes with a fixed input – "non-active" S-boxes and the four S-boxes with varying input – "active" S-boxes.

There are several possible ways to exploit the set of remaining non-filtered pairs which contains at least one slid-pair with non-negligible probability. One method would be to look at possibilities for the $4 \cdot 8 \cdot 3 = 96$ key bits of the keys $k_0$, $k_1' \oplus k_4, k_5$ suggested by the slid pairs. Notice that using the slid pair we can't split the keys $k_1'$ and $k_4$. Since each equation provides us with 8-bit test condition per S-box, total filtration power of two equations with four active S-boxes is $8 \cdot 2 \cdot 4 = 64$. That means that each pair will suggest about $2^{32}$ possible variants for the 96 key bits. For each analyzed pair all these key variants can be written in a compact cross-product form and stored using $4 \cdot 2^8 = 2^{10}$ bytes of memory. Using lookup tables it is also possible to perform this key enumeration efficiently in about $2^{10}$ steps. Thus each pool suggests $2^{63}$ choices for 96 key bits and these choices can be stored compactly in $2^{41}$ bytes of memory. The idea would be to request several pools in order to be sure that the correct key is listed several times and then try to find the frequently suggested correct variant of the key among the masses of incorrect guesses. While this approach might be marginally faster than exhaustive search we would prefer to find a better technique.

### 4.1 Generating More Slid Pairs

Another method is based on an observation that given a slid pair, it is relatively easy to produce many more slid pairs due to absence of diffusion in the slid equations. Indeed if one makes a small change to one of the S-boxes in plaintext $P_1$ and another change in the corresponding S-box at the ciphertext $C_2$, with a chance higher than $2^{-8}$ the pair will remain a slid pair (this idea is due to Gustaf Dellkrantz).

Moreover, we apply the "variation" to one of the four S-boxes that were active in the pool-generation step. Since plaintexts already run through all possible values for these S-boxes we only have to apply a single change to the ciphertext side. We then decrypt a new ciphertext into a new plaintexts. If the original pair was a properly slid pair we are guaranteed, that a new slid pair will be formed by the new query and one of the pairs from the old pool. The attack will proceed as follows (suppose we vary S-box $i$):

1. For each probable slid pair, find about $2^8$ choices for the 24-bits of the three key bytes $k_0$, $k_1' \oplus k_4, k_5$ at $i$-th S-box location. This step can be done in about $2^8$ lookups given a 32-bit lookup table and byte-values of plaintext and ciphertext from the $i$-th byte location of the analyzed pair.
2. Change $i$-th byte of the ciphertext $C_2$ to an arbitrary value, to get $C_2'$. Decrypt $C_2'$ to get the new plaintext $P_2'$.
3. For each 24-bit key guess from the 1st step, pick corresponding $P_1', C_1'$ pair from the pool. This is done given the first slid equation (and can be precomputed and stored in a table for each key-candidate, if the variation value is fixed).
4. Check that the pair is a properly slid pair, i.e. that the second equation holds as well. This is an 8-bit filtering condition for the wrong key guesses.
5. After steps 1-4 on the average one candidate for the 24-bits of $k_0$, $k_1' \oplus k_4, k_5$ at $i$-th location survives. We can thus apply second variation at the same $i$-th location, to check this single key candidate. The change in the ciphertext byte and the candidate key will define on the average a single value of the plaintext byte via the first slid equation. This plaintext byte will define which plaintext-ciphertext pair we need to pick from the pool. The second slid equation will thus provide us with an 8-bit filtration for the wrong pairs. After this step only about $2^{31}/2^8 = 2^{23}$ pairs will remain. For these we can repeat step 5 again, till only the proper slid pairs remains.
6. Given that only few wrong pairs (if any) may survive steps 1-5, we apply variation to other S-box locations and recover 96-bits of the keys $k_0$, $k_1' \oplus k_4, k_5$ corresponding to four active S-boxes.

To summarize, we will need two pools of size $2^{32}$ texts each (in order to increase the probability to find a slid pair), for each pool we request additional $2 \cdot 2^{31}$ adaptive chosen ciphertext queries. We could then repeat this attack with another set of pools, now fixing another 32-bits of the plaintexts to constant to completely recover the subkeys $k_0$, $k_1' \oplus k_4, k_5$. However this approach would double amount of data for the attack. Instead we will use already obtained slid pairs, and apply "variations" to the S-box locations that were non-active during the first phase of the attack. This way we completely recover the keys in just a few more adaptive chosen ciphertext/chosen plaintext queries. Using the two outer subkeys $k_0$ and $k_5$ we partially decrypt one round at the top and one round at the bottom. We are left with 3-round Khazad which can now be attacked using auxiliary techniques, for example Square attack, which has complexity of $2^9$ additional chosen plaintexts and $2^{16}$ S-box lookups (we may also use the assumption that $k_2 = k_3'$ which is the condition of the weak key class). The total data complexity of this attack is $2^{34}$ blocks and the analysis complexity is $2^{40}$ table lookups and $O(2^{32})$ memory. The attack works for one in $2^{64}$ keys.

### 4.2 A Generalization for an Arbitrary Involutional Cipher

This attack works whenever a cipher can be written as: $E_k = P \circ F \circ Q$ , where $F$ is an involution at least for some of the keys, and $P, Q$ are arbitrary keyed bijections. Then one can slide with a twist:

$$E_k(x_1) = Q \circ F \circ P(x_1) = y_1,$$
$$D_k(y_2) = P \circ F \circ Q(y_2) = x_2.$$

This provides two slid equations:

$$P(x_1) = Q(y_2),$$
$$P(x_2) = Q(y_1).$$

If these equations are easy to solve the attack will work.

## 5 Properties of the KeySchedule

Khazad uses iterations of a 128-bit Feistel scheme with the internal $F$-function being a round of Khazad with the constants (taken from the S-box) used as the 64-bit key. The user specified 128-bit key is used as a plaintext and the intermediate 64-bit values after each round become the actual subkeys of Khazad. Such keyschedule would possess nice 'sliding' or symmetry properties, but the randomized constants spoil them. Still the keyschedule is a Feistel block cipher with a bijective $F$-function. Thus given any pair of consequent subkeys $k_i, k_{i+1}$ it is possible to reconstruct all the subkeys backwards or forward (including the master key). Moreover it can be done for $k_i$, $k_{i+2}$ and even for $k_i$, $k_{i+3}$ in some cases.

## 6 Conclusions

In this paper we have shown structural properties of SPN ciphers built with involutional components. Using these we have shown interesting features of recently designed ciphers Khazad and Anubis, which where submitted to the NESSIE European pre-standardization project. The main observation is that there might be a distinguisher for 5-round Khazad if 3-round Khazad has any peculiarities in its cycle structure. Using the equivalent representation of Khazad we show a class of $2^{64}$ weak keys out of total $2^{128}$ keys. For a weak key, 5-round Khazad can be broken using $2^{34}$ chosen plain/ciphertext queries and $2^{40}$ table lookups for the analysis. Full round Khazad is not threatened by this result.

**Acknowledgment.** The author would like to thank Christophe De Cannière, Gustaf Dellkrantz and the members of the NESSIE project team for fruitful discussions. We would also like to thank the anonymous referees, whose comments helped to improve this paper.

## References

1. P. Barreto, V. Rijmen, *The Khazad Legacy-Level Block Cipher*, Submission to the NESSIE Project.
2. P. Barreto, V. Rijmen, *The Anubis Block Cipher*, Submission to the NESSIE Project.
3. A. Biryukov, D. Wagner, *Advanced Slide Attacks*, Proceedings of Eurocrypt'2000, LNCS 1807, pp.589–606, Springer-Verlag, 2000.
4. J. Daemen, V. Rijmen, *The Design of Rijndael*, Springer-Verlag, 2001.
5. H. Gilbert, M. Minier, *A collision attack on seven rounds of Rijndael*, In Proceedings of the third AES Conference, pp.230-241, NIST, 2000.
6. NESSIE, New European Schemes for Signatures, Integrity, and Encryption, IST-1999-12324, `http://www.cryptonessie.org`
7. M. Rejewski, *Mathematical Solution of the Enigma Cipher*, Cryptologia, Vol. 6, No. 1, pp. 1–18, 1982.