Probability 1 Iterated Differential in the SNEIK Permutation

Léo Perrin

Inria, France, leo.perrin@inria.fr

Abstract. SNEIK is a permutation at the core of a submission to the NIST lightweight cryptography project. In this note, we exhibit an iterated probability 1 differential in this permutation. However, it is still unclear if this differential can be used to construct attacks against the permutation in a mode, e.g., against the hash function SNEIKHA.
We also suggest a simple fix: adding a 32-bit rotation in one tap prevents this issue.

Keywords: SNEIK · NIST lightweight cryptography project · Differential Cryptanalysis · ARX · Permutation

1 The SNEIK Permutation

SNEIK [Saa19] is a submission to the NIST lightweight cryptography project. It relies on a 512-bit ARX-based permutation which is best described by the diagram in Figure 1.

It operates on an array s of sixteen 32-bit words indexed from 0 to 15. At time i, the word $s[i]$ is computed from the others words $s[j]$ using the following sequence of operations which we describe using C-style notations (a reduction modulo 16 of the indices is implicitely made):

```
t = s[i-1];
t ^= s[i-16] ^ d[i];
t = L1(t) ^ s[i-1];
t += s[i-14];
t = L2(t) ^ s[i-16];
s[i] = t;
```

where L1 and L2 are linear permutations and d$[i]$ is a round constant—there details do not matter for this observation.

2 Probability 1 Differential

Theorem 1. *Any n-bit string $x = (x_{n-1}, ..., x_0)$ is easily mapped to the integer $\mathsf{int}(x) = \sum_{j=0}^{n-1} x_i 2^i$. We let as usual $a \boxplus b$ denote the binary representation of $\mathsf{int}(a) + \mathsf{int}(b) \mod 2^n$ and $a \oplus b$ denote the bitwise exclusive or of a and b. Further, let δ_n be the n-bit string such that $\mathsf{int}(\delta_n) = 2^{n-1}$, i.e., $\delta_n = (1, 0, 0, ..., 0)$. Then the following equality holds with probability 1:*

$$(a \boxplus b) \oplus \big((a \oplus \delta_n) \boxplus (b \oplus \delta_n)\big) = 0 \ .$$

Proof. The bit of highest weight of $a \boxplus b$ is a linear function of a_{n-1} and b_{n-1} and is the only bit of $a \boxplus b$ depending on either of these variables. Hence, complementing both of these bits does not change the result of the modular addition. □

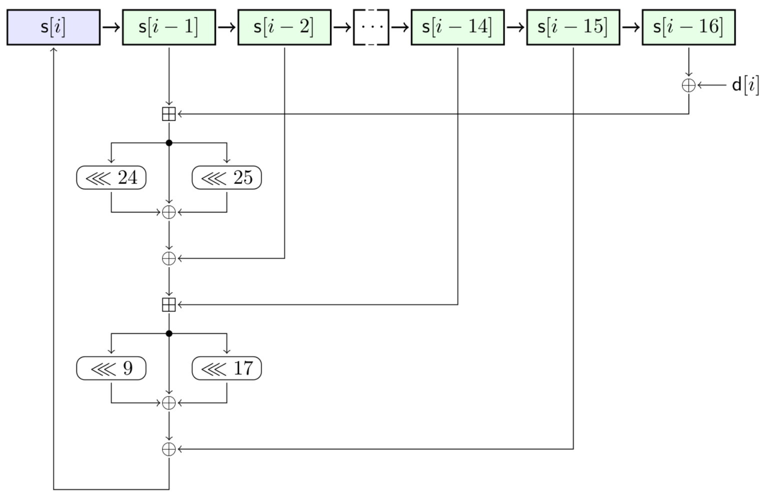

Figure 1: A diagram representing the SNEIK permutation (Figure 1 of [Saa19]).

Corollary 1. *let s and s' be two SNEIK states such that $s'[j] = s[j] \oplus \delta_n$ for all j. Then it holds that $t \oplus t' = \delta_n$, where t and t' are the words output by the state update function in s and s' respectively.*

In other words, if the difference on all words is equal to δ_n before *the state update, then it is equal to δ_n on all words* after *the state update.*

Proof. We simply need to show that the difference between t and t' is equal to δ_n when they are computed using the algorithm given above. Suppose that $s[j] \oplus s'[j] = \delta_n$ for all j. Then the difference in the input of `L1` is

$$\big(s[i] \boxplus (s[i-1] \oplus d[i])\big) \ \oplus \ \big((s[i] \oplus \delta_n) \boxplus (s[i-1] \oplus d[i] \oplus \delta_n)\big) \ ,$$

which means we can apply Theorem 1 and obtain that it is equal to 0. The difference is therefore equal to zero in the *output* of `L1`.

Let y be this output (which is thus the same for both s and s'). The difference in the input of `L2` is equal to

$$\big((s[i-2] \oplus y) \boxplus s[i-14]\big) \oplus \big((s[i-2] \oplus \delta_n \oplus y) \boxplus (s[i-14] \oplus \delta_n)\big) \ .$$

Again, we deduce from a direct application of Theorem 1 that this difference is equal to 0.

In the end, we therefore have that $t \oplus t'$ is equal to $s[i-15] \oplus s'[i-15]$ which, under our assumption, is equal to δ_n. Hence, we have that $t \oplus t' = \delta_n$. □

3 Conclusion

We verified this observation using the reference implementation. However, it is unclear at the moment if this observation can be turned into an attack against a cipher/hash function relying on this permutation.

Finally, we remark that this behaviour is simply avoided by adding a rotation in one of the taps, i.e., by replacing the operation $t \leftarrow t \oplus s[i-2]$ with $t \leftarrow t \oplus (s[i-2] \lll 1)$.

References

[Saa19] Markku Juhani O. Saarinen. SNEIKEN and SNEIKHA authenticated encryption and cryptographic hashing. Available online at `https://github.com/pqshield/sneik`., 2019.