Forgery Attack on SNEIKEN

Mustafa Khairallah

School of Physical and Mathematical Sciences
Nanyang Technological University
mustafam001@e.ntu.edu.sg

Abstract. This document includes a collision/forgery attack against SNEIKEN, where every message with more than 128 bytes of associated data can be converted into another message with different associated data and the same ciphertext/tag. The attack is a direct application of the probability 1 differential of the SNEIK permutation found by Léo Perrin in [Per19]. We verify the attack using the reference implementation of SNEIKEN128 provided by the designers, providing an example of such collisions.

Keywords: AEAD · forgery · SNEIK · collision

1 Introduction

SNEIKEN [Saa19] is a family of Authenticated Encryption with Associated Data (AEAD) algorithms. It is a part of the round 1 cadidate of the NIST Lightweight Cryptography Standardization process, SNEIK. The modes in this family are permutation based, using the SNEIK permutation. The permutation operates iteratively on an array of 16 32-bit words. In [Per19], Léo Perrin found a probability 1 iterative differential property for the SNEIK permutation, where if we have a pair of inputs such that each of the input 16 words have a difference 0x80000000, each of the output words will have the same difference. However, it was unclear whether such a weakness directly leads to attacks on the modes that use the SNEIK permutation.

2 Attack on the SNEIK AEAD modes

For our attack we rely on the associated data (AD) absorption part of the algorithm. SNEIKEN128/192/256 use a full state absorption approach for AD. Hence, for every 64 bytes of AD, the full state is updated. The attack works as follows:

1. The attacker choose any message with more than 128 bytes of AD, and random M and N. He then sends an encryption query (AD,M,N), receiving (C,T).
2. The attacker changes every 4th byte of the first 128 bytes of AD by adding a difference 0x80, the is called AD$'$.
3. The attacker sends a decryption query (AD$'$,C,T,N). The decryption will succeed with probability 1, outputting M.

In order to see how the attack works, we trace the difference up to the absorption of the second input AD block. The first AD block, A_1 is XORed with a random value R_1, which the output of the initialization phase. The SNEIK permutation operates on $R_1 \oplus A_1$, generating $P(R_1 \oplus A_1)$, which is finally XORed with A_2. Now, we change A_1 to

$A_1 \oplus \Delta$ and A_2 to $A_2 \oplus \Delta$, where Δ is the differential found in [Per19]. We note that the trace will be changed to $R_1 \oplus A_1 \oplus \Delta$, $P(R_1 \oplus A_1 \oplus \Delta)$ and $P(R_1 \oplus A_1 \oplus \Delta) \oplus A_2 \oplus \Delta$. However, we know from [Per19] that $P(R_1 \oplus A_1 \oplus \Delta) = P(R_1 \oplus A_1) \oplus \Delta$. Hence,

$$P(R_1 \oplus A_1 \oplus \Delta) \oplus A_2 \oplus \Delta = P(R_1 \oplus A_1) \oplus \Delta \oplus A_2 \oplus \Delta = P(R_1 \oplus A_1) \oplus A_2 \quad (1)$$

Which is a collision on the intermediate value of the construction, and since no more differences are introduced after this point, then the ciphertext and tag are identical.

3 Example

We have verified our attack using the reference implementation of SKEINEN128 [Saa19]. We generated the example collision shown below:

```
Key = 000102030405060708090A0B0C0D0E0F
Nonce = 000102030405060708090A0B0C0D0E0F
PT = empty-string
AD =
000102030405060708090A0B0C0D0E0F
101112131415161718191A1B1C1D1E1F
202122232425262728292A2B2C2D2E2F
303132333435363738393A3B3C3D3E3F
404142434445464748494A4B4C4D4E4F
505152535455565758595A5B5C5D5E5F
606162636465666768696A6B6C6D6E6F
707172737475767778797A7B7C7D7E7F
808182838485868788898A8B8C8D8E8F
909192939495969798999A9B9C9D9E9F
A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF
CT = 517ACD4E2A0919A30FBEB1BC3436598C

Nonce = 000102030405060708090A0B0C0D0E0F
PT = empty-string
AD =
000102830405068708090A8B0C0D0E8F
101129314151697181 91A9B1C1D1E9F
202122A3242526A728292AAB2C2D2EAF
303132B3343536B738393ABB3C3D3EBF
404142C3444546C748494ACB4C4D4ECF
505152D3545556D758595ADB5C5D5EDF
606162E3646566E768696AEB6C6D6EEF
707172F3747576F778797AFB7C7D7EFF
808182838485868788898A8B8C8D8E8F
909192939495969798999A9B9C9D9E9F
A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF
CT = 517ACD4E2A0919A30FBEB1BC3436598C
```

4 Conclusion

The attack we show in this document disproves the security claims of SNEIKEN128, SNEIKEN192 and SNEIKEN256 against forgery attacks. However, it is not related to SNEIKHA and it is is still not clear if the differential found in [Per19] can be used to find collision attacks against SNEIKHA.

References

[Per19] Léo Perrin. Probability 1 Iterated Differential in the SNEIK Permutation. Cryptology ePrint Archive, Report 2019/374, 2019. https://eprint.iacr.org/2019/374.

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