

Distinguishers for the Squeezing of the Alternating Sponge and Duet

Gaëtan Leurent and César Mathéus

INRIA, Paris, France

Abstract

Recently, the Institute of Commercial Cryptography Standards (ICCS) launched the Next Generation Commercial Cryptographic Algorithm Program (NGCC)², which includes a standardization process for cryptographic hash functions. In this note, we analyze the mode of operation of Duet [SCL⁺26], one of the hash-function candidates.

Duet is a permutation-based hash function directly inspired by the Alternating Sponge (ASP) [LSY⁺26]. Both constructions use two permutations on $b = r + c$ bits, split into an outer part of r bits and an inner part of c bits. We first present a distinguisher against ASP requiring $\mathcal{O}(2^r)$ permutation queries, which contradicts the claimed indistinguishability bound in a non-trivial parameter regime. We then introduce a distinguisher and a state-recovery attack against Duet. Both attacks require only a constant number of h -bit output blocks, with $h \leq c$, and $\mathcal{O}(2^{r+c-h})$ permutation queries. These attacks do not affect the claimed collision or preimage resistance, but they distinguish Duet’s multi-block output from random, with consequences if Duet is used as a XOF or a KDF; state recovery also enables length extension. With the concrete parameters used in Duet, the distinguisher has a high complexity: 2^{449} for Duet-512 and 2^{897} for Duet-1024. Since the designers don’t give explicit claims for indistinguishability, it is unclear how this impacts the design.

Keywords: hash functions, alternating sponge, Duet, distinguishers

1 Introduction

Duet [SCL⁺26] is a hash-function candidate in the NGCC program. Its permutation-based construction is derived from the Alternating Sponge (ASP) [LSY⁺26], which comes with a beyond-birthday-bound indistinguishability claim. We identify structural weaknesses in the squeezing procedures of both constructions and obtain unexpectedly efficient distinguishers in some parameter regimes.

Our ASP distinguisher uses $\mathcal{O}(2^r)$ permutation queries and falls below the claimed indistinguishability threshold when $r < 2c/3$. For Duet, we give a general attack for output blocks of $h \leq c$ bits. It uses $\mathcal{O}(2^{r+c-h})$ permutation queries to find states consistent with the first three output blocks of a possibly unknown message. For Duet-512 and Duet-1024, the first two output blocks already yield a distinguisher from independent random outputs. Table 1 summarizes the original ASP claim and the costs of the attacks considered here.

Remark: After contacting the authors of ASP and Duet, we learned that Michele Montedoro had independently reported a similar $\mathcal{O}(2^r)$ distinguisher for ASP to them by email on August 24, 2026. We acknowledge this independent observation.

Emails: Gaetan.Leurent@inria.fr (Gaëtan Leurent), Cesar.Matheus@inria.fr (César Mathéus)

² <https://www.niccs.org.cn/en/>

Table 1: Original ASP indistinguishability claim and attacks considered in this paper. The rate is r , the capacity is c , and each Duet output block has $h \leq c$ bits. Constant factors are omitted from the attack costs.

Construction	Claim or attack	Query cost	Scope
ASP	Original claim	$2^{\min\{2c/3, (r+c)/2\}}$	Indistinguishability bound [LSY ⁺ 26].
ASP	Output-shift distinguisher	$\mathcal{O}(2^r)$	Below the claim when $r < 2c/3$.
Duet	Two-block distinguisher	$\mathcal{O}(2^{r+c-h})$	Requires two output blocks.
Duet	Three-block state search	$\mathcal{O}(2^{r+c-h})$	Requires three output blocks.

2 A Distinguisher for the Alternating Sponge

We present a distinguisher using $\mathcal{O}(2^r)$ permutation queries, $\mathcal{O}(2^r)$ elementary operations, and negligible memory, together with two construction queries. Under a suitable condition, absorbing one additional message block performs exactly the same permutation calls as the first squeezing round. The two resulting output streams therefore coincide after a shift of two blocks. This should not happen for an ideal hash function.

We first define the unpad function. Given a string P whose length is a multiple of r , and such that the last block is non-zero, $\text{unpad}(P)$ is the unique message M such that $\text{pad}(M) = P$.

The distinguisher proceeds as follows.

- **Step 1: Generate and test states.** Generate $\mathcal{O}(2^r)$ distinct rooted states $S_i = (A_i, B_i, C_i)$ reached by absorbing short padded messages $\text{pad}(M_i)$. For each state, query

$$f_1((A_i \oplus B_i) \parallel C_i) = U_i \parallel D_i.$$

- **Step 2: Find an overlap.** Look for a state satisfying $B_i \neq 0^r$ and $U_i = \sigma(B_i)$. Define M'_i by

$$M'_i = \text{unpad}(\text{pad}(M_i) \parallel B_i).$$

Query the construction on M_i and M'_i and test the resulting output shift. In the real construction,

$$\mathcal{H}(M_i)[k+2] = \mathcal{H}(M'_i)[k] \quad \text{for every } k \geq 0,$$

where $\mathcal{H}(M)[k]$ denotes the k th r -bit output block, numbered from zero.

Correctness. After absorbing $\text{pad}(M_i)$, the first output block is $z_0 = A_i \oplus B_i$. The first squeezing call is

$$f_1((A_i \oplus B_i) \parallel C_i) = U_i \parallel D_i,$$

and the next output block is $z_1 = B_i \oplus U_i$. The second squeezing call is

$$f_2((B_i \oplus U_i) \parallel D_i) = V_i \parallel E_i,$$

which gives $z_2 = U_i \oplus V_i$. These operations are shown in the top part of Figure 1.

If we instead absorb the block B_i after reaching S_i , the first call is

$$f_1((A_i \oplus B_i) \parallel C_i) = U_i \parallel D_i,$$

followed by

$$f_2((B_i \oplus \sigma(B_i)) \parallel D_i) = V'_i \parallel E'_i.$$

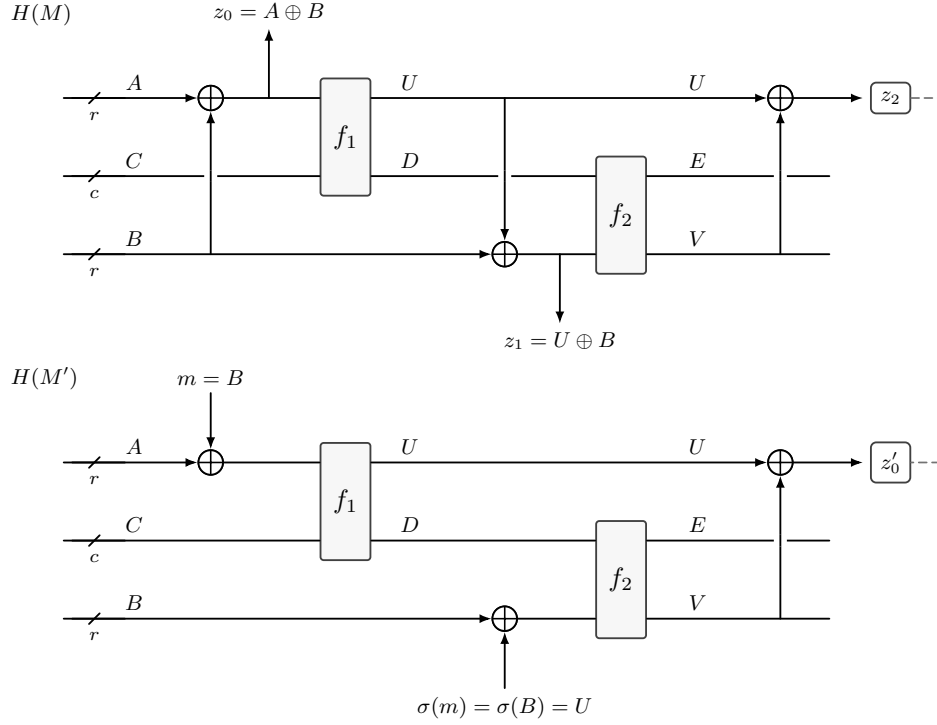


Figure 1: Absorption and squeezing use the same permutation calls when $\sigma(B) = U$. Here $\text{pad}(M') = \text{pad}(M) \parallel B$.

When $\sigma(B_i) = U_i$, both permutation calls are identical to the squeezing calls above, resulting in $V'_i \parallel E'_i = V_i \parallel E_i$. Absorption therefore ends in (U_i, V_i, E_i) , and the first output block for M'_i is $U_i \oplus V_i = z_2$. Both computations then continue from the same squeezing state, so all subsequent output blocks agree. The bottom part of Figure 1 shows this additional absorption step.

The search tests an r -bit equality. For fresh permutation inputs in the regime of interest, each trial succeeds with probability about 2^{-r} , which gives constant success probability after $\mathcal{O}(2^r)$ trials. This cost is below the original claimed security threshold

$$2^{\min\{(r+c)/2, 2c/3\}}$$

whenever $r < 2c/3$ [LSY⁺26]. We remark that when $r \geq 2c/3$, the alternating sponge offers worse performances than the basic sponge³, so that the regime where $r < 2c/3$ should be the main regime for the ASP construction.

This distinguisher does not apply directly to Duet, which adds capacity feed-forward and extracts its output from the capacity. These changes lead to a different distinguisher, described next.

³For a fixed permutation width b , we compare the alternating sponge with capacity c and rate $r = b - c$, and the plain sponge with capacity $c' = 4c/3$ and rate $r' = b - c'$. Both constructions have a security bound of $2c/3 = c'/2$. The alternating sponge requires 2 permutation calls to absorb r bits of message, corresponding to $r/2 = b/2 - c/2$ bits per permutation call. On the other hand, the plain sponge absorbs $r' = b - 4c/3$ bits per permutation call. When $r \geq 2c/3$, we obtain $r/2 \geq r'$.

3 A State-recovery for Duet

Duet uses capacity feed-forward and outputs h bits of the capacity at each squeezing step, with $1 \leq h \leq c$ [SCL⁺26, Section 2.5]. The squeezing part of Duet is illustrated in Figure 2.

We describe two attacks that recovers (part of) the internal state from the output. In particular, we can distinguish valid outputs from impossible one, which breaks the PRF property if Duet is used with a keyed message.

3.1 Two-block distinguisher

We first give an attack using two blocks of output $z_0 \parallel z_1$, with $z_0, z_1 \in \{0, 1\}^h$. The first output block reveals h capacity bits. We guess the remaining $c - h$ capacity bits together with the first rate branch. This constitutes the full input to f_1 ; we can therefore compute the second output block to check this guess.

More precisely, for every $s \in \{0, 1\}^{c-h}$ and $X \in \{0, 1\}^r$, we define $C_0 := z_0 \parallel s$ and query

$$f_1(X \parallel C_0) =: U \parallel D.$$

Then, we compute $C_1 := D \oplus C_0$. If $\text{left}_h(C_1) = z_1$, then (X, C_0) is a valid candidate for the input to f_1 .

When $h = c$, the string s is empty and no capacity bits need to be guessed.

Distinguisher and Complexity. If $z_0 \parallel z_1$ is a valid output for Duet, then at least one guess (s, X) will be consistent with the output, and we expect 2^{r+c-2h} extra states to randomly match the output. On the other hand, if $z_0 \parallel z_1$ is a random value, we only expect 2^{r+c-2h} values to randomly match the output. When $h > (r + c)/2$, this gives a distinguisher because random values are unlikely to have a matching state.

The distinguisher requires 2^{r+c-h} calls to the permutation.

3.2 State recovery

We extend the distinguisher into a full state recovery, recovering the state (C_0, X, Y) at the beginning of the squeezing phase from the first blocks of output stream (assuming a long output). If Duet is used as a stream cipher (with a secret input), this means we can predict the keystream from the first few blocks.

We first recover all states consistent with the first three output blocks; we expect $1 + 2^{2r+c-3h}$ compatible states. When $h < (2r + c)/3$, we filter those states using $t = \lceil (c + 2r)/h \rceil$ blocks of output to obtain a single state. We follow the notations of Figure 2.

- **Step 1: Recover candidates for the first f_1 input.** Using the distinguisher above, we recover on average $1 + 2^{r+c-2h}$ solutions (C_0, X) using 2^{r+c-h} calls to the permutation.

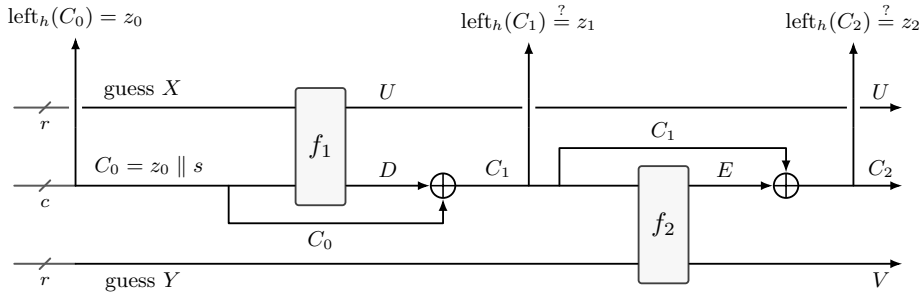
We store the corresponding input-output pairs for f_1 $((C_0, X), (D, U))$.

- **Step 2: Recover candidates for the first f_2 input.** We repeat the same approach to recover candidates for (C_1, Y) . We recover on average $1 + 2^{r+c-2h}$ solutions using 2^{r+c-h} calls to the permutation.

We store the corresponding input-output pairs for f_2 $((C_1, Y), (E, V))$.

- **Step 3: Merge the lists to recover full state candidates.** We merge the two lists, and filter candidates such that $C_1 = C_0 \oplus D$. Since h bits of this equation are already satisfied, we expect $(1 + 2^{r+c-2h})^2 2^{h-c} \approx 1 + 2^{2r+c-3h}$ solutions.

Duet squeezing

**Figure 2:** Duet distinguisher and state recovery.

- **Step 4: Filter candidates using more output blocks.** When $h < (2r + c)/3$, we have many remaining candidates for the initial state (C_0, X, Y) . We enumerate them and check the full t output blocks to obtain a single solution.

Correctness and cost. The correct initial state satisfies all the filtering steps in the state recovery, therefore it will be among the candidates.

The state recovery requires $t = \max(3, \lceil (c + 2r)/h \rceil)$ blocks of output to recover a unique internal state. It uses at most $(t - 1)2^{r+c-h}$ distinct permutation queries (because each function only has $r + c - h$ unknown input bits), and $2^{r+c-h+1} + (t - 2)2^{2r+c-3h}$ operations. With high probability, a single state matches the t blocks of output.

In particular, we obtain a state recovery with complexity $\mathcal{O}(2^{r+c-h})$ when $h \geq r/2$, extending the range of parameters of the distinguisher above without increasing the complexity.

Relation with sponge squeezing. The phenomenon exploited by this distinguisher is not specific to Duet. It is related to the amount of state that remains hidden during squeezing. For the standard sponge, the absorbing and squeezing rates are both r , so squeezing leaves $c = b - r$ hidden bits. A term with security scale 2^c is therefore dominated by the usual, tight $2^{c/2}$ indistinguishability bound of the sponge [BDPA08]. When the squeezing rate is increased, however, fewer than c state bits remain hidden, and this effect can become security-relevant. This phenomenon already appears in the analysis of PHOTON: Naito and Ohta show that increasing the squeezing rate may reduce the indistinguishability security, and that the usual $2^{c/2}$ level is retained when the capacity left hidden during squeezing is at least roughly $c/2 + \log_2 c$ [NO14].

A particularly close comparison is Sponge-F, which also allows squeezing from the capacity. Its indistinguishability bound contains a term corresponding to the security scale $2^{b-r'}$, where r' is the squeezing rate [GHJ⁺26]. Setting $r' = h$ gives the same exponent as our Duet distinguisher, namely $2^{b-h} = 2^{r+c-h}$. This is especially interesting because Duet maintains a larger state of $2r + c$ bits, so recovering all the hidden state at once would seemingly require guessing $2r + c - h$ bits. However, the alternating structure inherited from ASP allows the two r -bit rate branches to be recovered successively rather than simultaneously. By exploiting consecutive output blocks, our attack first recovers one rate branch and then uses this information to recover the other. As a result, the recovery cost is only 2^{r+c-h} , rather than 2^{2r+c-h} . Despite their different state sizes, Sponge-F and Duet exhibit the same security limitation during squeezing: the relevant exponent is determined by the number $b - h$ of bits hidden within a single permutation state, rather than by the total number of bits in the construction's state.

3.3 Application to Duet's parameters.

The Duet proposal gives the following parameter sets [SCL⁺26, Table 5]:

Instance	r	c	h	$c - h$
Duet-512	384	576	512	64
Duet-768	1088	832	768	64
Duet-1024	768	1152	1024	128

In particular, none of the instances reveals the full capacity in a single output block.

Duet-512 and Duet-1024. These two instances use the same ratio

$$r = \frac{3h}{4}, \quad c = \frac{9h}{8}, \quad c - h = \frac{h}{8}.$$

Since $h > (r + c)/2$ the distinguisher of Section 3.1 returns a single candidate with high probability, and the full state recovery has complexity $2^{r+c-h+1} = 2^{7h/8+1}$ using $t = 3$ blocks of output:

Instance	Complexity
Duet-512	2^{449}
Duet-1024	2^{897}

Duet-768. In Duet-768, r is larger than the security goal, therefore our attacks are not meaningful.

Scope of the attacks. We obtain distinguishers and state recovery for Duet-512 and Duet-1024, using two output blocks for distinguishing and three for state recovery. This does not apply to the standard fixed-length digest with a single output block, but the Duet specification also specifies a XOF with several blocks of output. The attacks do not target the standard security notions of collision resistance or preimage resistance, but they break the KDF security notion if Duet is used for key derivation as suggested in the specification. The state-recovery attack also enables a length-extension attack.

Our attacks have a complexity $2^{7h/8}$, above the security claim for collision resistance ($2^{h/2}$) and below the claim for preimage resistance (2^h). The Duet specification does not give a concrete indistinguishability claim, nor a security claim for distinguishing attacks or for KDF security, so it is unclear whether this violates the security goals, but it seems that this type of analysis was not anticipated by the authors.

In particular, the authors refer to the analysis of the Alternating Sponge [SCL⁺26, Section 4.2.2] with security threshold $2^{\min\{2c/3, (r+c)/2\}}$ to indicate that the alternating schedule provides a strong generic-security foundation. As shown in Section 2, the Alternating Sponge does not actually reach this security level, and this section shows that Duet does not reach it either in general: when $h = c$ and $r \ll c$, the security of Duet is at most 2^r , which is significantly below the ASP claimed bound.

We note that the attacks use only forward queries to f_1 and f_2 , so that they apply both to random permutations and to random functions.

References

- [BDPA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indistinguishability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer*

- Science*, pages 181–197. Springer, 2008. doi:[10.1007/978-3-540-78967-3_11](https://doi.org/10.1007/978-3-540-78967-3_11).
- [GHJ⁺26] Chun Guo, Kai Hu, Shuntian Jiang, Yanhong Fan, Yong Fu, Bart Preneel, and Meiqin Wang. Permutation-based hash from non-idealized assumptions: Adding feed-forward to sponge. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part VI*, volume 16805 of *Lecture Notes in Computer Science*, pages 65–97. Springer, 2026. doi:[10.1007/978-3-032-35412-9_3](https://doi.org/10.1007/978-3-032-35412-9_3).
- [LSY⁺26] Ziyang Luo, Yaobin Shen, Hailun Yan, Lei Wang, and Dawu Gu. Alternating sponge: A low-memory hash function with beyond-birthday-bound security. Cryptology ePrint Archive, Paper 2026/707, 2026. URL: <https://eprint.iacr.org/2026/707>.
- [NO14] Yusuke Naito and Kazuo Ohta. Improved indifferentiable security analysis of PHOTON. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2014. doi:[10.1007/978-3-319-10879-7_20](https://doi.org/10.1007/978-3-319-10879-7_20).
- [SCL⁺26] Yaobin Shen, Tingting Cui, Ziyang Luo, Yan He, and Feng Li. Duet: A dual-branch alternating sponge hash function. Algorithm specification, Next Generation Commercial Cryptographic Algorithm Program, June 2026. Proposal dated June 30, 2026. URL: <https://www.niccs.org.cn/niccs/Proposal/Cryptographic%20Hash%20Algorithms/Round%201%20candidates/Duet.zip>.