Collision Attacks on Round-Reduced Gimli-Hash/Ascon-Xof/Ascon-Hash

Rui Zong[1], Xiaoyang Dong[1], and Xiaoyun Wang[1,2]

[1] Institute for Advanced Study, Tsinghua University, Beijing, China
[2] Shandong University, Jinan, China
{zongrui3}@163.com

Abstract. The NIST-approved lightweight cryptography competition is an ongoing project to look for some algorithms as lightweight cryptographic standards. Recently, NIST chooses 32 algorithms from the 57 submissions as Round 2 candidates.
Gimli and Ascon are both the Round 2 candidates. In this paper, we analyze the security of their hash mode against collision attacks. Concretely, we mount collision attacks on three hash functions: Gimli-Hash, Ascon-Xof and Ascon-Hash. These three hash functions are all based on sponge constructions.
We give two attack strategies for searching collisions in sponge-based hash functions. Following one strategy, we give two non-practical collision attacks: a 6-round collision attack on Gimli-Hash with time complexity 2^{113} and a 2-round collision attack on Ascon-Hash with time complexity 2^{125}. Following the other strategy, we give a practical attack on 2-round Ascon-Xof with a 64-bit output. The time complexity is 2^{15}.
We search for the differential characteristics using the MILP technique and the target differential algorithm.

Keywords: Collision Attack · Gimli-Hash · Ascon-Xof · Ascon-Hash · Attack Strategy · Sponge-based Hash Function.

1 Introduction

As early as in 2013, NIST started the lightweight cryptography project to evaluate [1] the performance of NIST-approved cryptographic algorithms on constrained devices, and in the meanwhile to study the requirements for a lightweight cryptographic algorithm standard. However, most of NIST-approved algorithms were designed for desktop/server environments and they are not acceptable when considering the requirements of the emerging areas, e.g., healthcare, sensor networks, the Internet of Things, cyber physical systems. In 2016, NIST provided an overview of the project and decided to seek for some new algorithms as lightweight cryptographic standards. The submission deadline is February 25, 2019. NIST received 57 submissions and 56 of them will be the first round candidates after the initial review. Recently the project proceeds into Round 2. NIST chooses 32 submissions as Round 2 candidates, including Gimli and Ascon.

Gimli [2] aims at applications that can achieve desirable performance in a broad range of environments, thus providing a satisfactory solution for communications between different environments. This submission includes a family Gimli-Cipher of authenticated ciphers and the recommended member Gimli-24-Cipher is with a 256-bit key, a 128-bit nonce and a 128-bit tag. In addition, it includes a family Gimli-Hash of hash functions. The recommended member is Gimli-24-Hash with a 256-bit output. All these families are built on top of a family of 384-bit permutations called Gimli. The recommended permutation is Gimli-24, i.e., the permutation consists of 24 round functions.

Ascon [7] aims at applications which should be easy to implement, even with added countermeasures against side-channel attacks. This submission also includes both authenticated encryption with associated data and hashing functionality. The suite consists of the authenticated ciphers Ascon-128 and Ascon-128a, these two ciphers have been selected as primary choice for lightweight authenticated encryption in the final portfolio of the CAESAR competition. For the hash mode, it provides three functions: Ascon-Hash with a 256-bit output, Ascon-Xof with an arbitrary length output and Ascon-80pq with increased resistance against quantum key-search. All schemes use the 320-bit permutation which has 12 round functions.

Our Contributions.

In this paper, we focus on the hash mode of these two proposals: Gimli-Hash, Ascon-Xof and Ascon-Hash. These three hash functions are all based on sponge constructions. We firstly describe two attack strategies for searching collisions of sponge-based hash functions. Following one of these two strategies, we give non-practical collision attacks on Gimli-Hash and Ascon-Hash. For Gimli-Hash, we can find a 6-round collision with time complexity 2^{113}. For Ascon-Hash, we can find a 2-round collision with time complexity 2^{125}. Following the other strategy, we give a practical attack on 2-round Ascon-Xof with a 64-bit output. The time complexity is 2^{15}. We search for the characteristics using the MILP technique and the target differential algorithm.

Table 1. Collision attack results in this paper

Hash Function	Round Number	Complexity
Gimli-Hash	6	2^{113}
Ascon-Xof	2	2^{15}(Practical)
Ascon-Hash	2	2^{125}

Related Results.

For Gimli, the authors [2] analyze the security of its reduced-round versions at two security requirements: 1) the number of rounds to achieve the avalanche effect for each bit, 2) the number of rounds that can reach a state full of 1 from a state that only has one bit is set. They states that 10 rounds are required for each bit to change half of the state and 8 rounds are required to fill up the state. They also give differential cryptanalysis results of the Gimli permutation. However, no specific results about the Gimli-Hash is given.

For Ascon, there have been numerous results. [8, 14, 6, 13, 10] and the submission document [7] give results about the security of the Ascon permutation.[12, 8, 11, 9] give analysis result of Ascon. [4] analyzes the security of Ascon-Xof against pre-image attack. [4] also gives a specific 4-round collision in Ascon-Xof. However, this is a semi-free-start collision with a chosen IV, without any complexity analysis. As far as we know, there have been no works about the security of Ascon-Hash and Ascon-Xof with the constant IVs.

2 Preliminaries

In this section, we first introduce some related definitions and notations used in our work. Then we give the specification of three hash functions: Gimli-Hash, Ascon-Xof and Ascon-Hash, and some useful observations of them.

2.1 Notations

Table 2. Notations

p^i	the i-th inner primitive of a hash function
r	the length of the rate part
c	the length of the capacity part
S	the state of a hash function
S_r	the rate part of a sponge-based hash function state, r bits
S_c	the capacity part of a sponge-based hash function state, c bits
S^i	the state in the i-th primitive
S^i_j	the input state of the j-th round in the i-th primitive
S^i_{jr}	the rate part of S^i_j
S^i_{jc}	the capacity part of S^i_j
M	the message before padding
$\overline{M}$	the message after padding
M_i	one block of the padded message, $\|M_i\| = r$, $i \geq 1$

2.2 The Sponge-based Hash Function

Cryptographic sponge hash function is proposed in [3]. It takes a variable-length message as input and produces an infinite hash tag with a finite state S. The state S is composed of two parts: the rate part S_r of r bits and the capacity part S_c of c bits. To evaluate the sponge function, one proceeds in three phases with an inner primitive p: 1) Initialization: get the state value before proceeding the message blocks; 2) Absorbing: proceed each message block; 3) Squeezing: produce the hash tag. The mode of operation is illustrated in Figure 1 and specified in Algorithm 1.

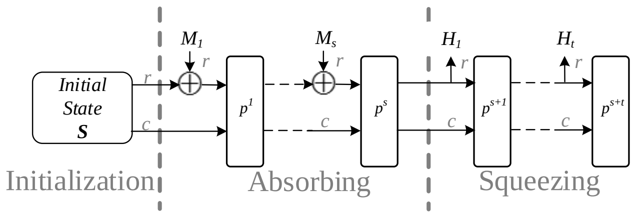

Fig. 1. The sponge-based hash function.

Algorithm 1 Hashing

```
Input: Message $M \in \{0,1\}^*$, output size $l$, inner primitive $p$
Output: hash tag $H \in \{0,1\}^l$
1: Initialization
       For Gimli-Hash,                  $S \leftarrow (0^r||0^c)$;
       For Ascon-Hash and Ascon-Xof,    $S \leftarrow p(IV||0^c)$;
2: Absorbing
       $\overline{M} \leftarrow Padding(M)$
       $M_1 \cdots M_s \leftarrow \overline{M}$
       for $i = 1, \cdots, s$ do:
           $S \leftarrow p((S_r \oplus M_i)||S_c)$
3: Squeezing
       for $i = 1, \cdots, t = \lceil l/r \rceil$ do:
           $H_i \leftarrow S_r$
           $S \leftarrow p(S)$
       return $\lfloor H_1 || \cdots || H_t \rfloor_l$
```

2.3 Gimli

Primitive. The primitive used in Gimli-Hash is called Gimli. It's a permutation that applies 24 rounds to a 384-bit state which can be represented as a 3×4 matrix of 32-bit words, i.e.,

$$state = \begin{bmatrix} s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \\ s_{1,0}, s_{1,1}, s_{1,2}, s_{1,3} \\ s_{2,0}, s_{2,1}, s_{2,2}, s_{2,3} \end{bmatrix}$$

Each round is composed of three operations:

- **Non-linear layer** f_S

  The non-linear layer is essentially four 96-bit SP-boxes applied to each column.The operation is as follows:

$$
\begin{aligned}
&x \leftarrow s_{0,j} <<< 24\\
&y \leftarrow s_{1,j} <<< 9\\
&z \leftarrow s_{2,j}\\
&s_{2,j} \leftarrow x \oplus (z \ll 1) \oplus ((y \wedge z) \ll 2)\\
&s_{1,j} \leftarrow y \oplus x \oplus ((x \vee z) \ll 1)\\
&s_{0,j} \leftarrow z \oplus y \oplus ((x \wedge y) \ll 3)
\end{aligned}
$$

– **Linear mixing layer f_L**
The linear layer consists of two operations, *Small-Swap* and *Big-Swap*. *Small-Swap* occurs every 4 rounds starting from the 1st round. *Big-Swap* occurs every 4 rounds starting from the 3rd round.

Small-Swap
$s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{0,1}, s_{0,0}, s_{0,3}, s_{0,2}, (r\ mod\ 4 = 0)$
Big-Swap
$s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{0,2}, s_{0,3}, s_{0,0}, s_{0,1}, (r\ mod\ 4 = 2)$

– **Constant addition f_C**
In every fourth round, we XOR the round constant $0x9e377900 \oplus r$ to the first state word $s_{0,0}$.

We denote the state within the i-th round function as follows:

$$S_i \xrightarrow{f_S} S_i' \xrightarrow{f_L} S_i'' \xrightarrow{f_C} S_{i+1}$$

Gimli-Hash. The Gimli-Hash has a 384-bit state with $r = 128$ and $c = 256$. Its hash output is 256 bits.

The Gimli-Hash firstly initializes the 384-bit to all-zero. It then reads sequentially through a variable-length input as a series of 16-byte input blocks after padding.

The input ends with one final empty or partial block, having b bytes, $0 \leq b \leq 15$. The padding rule is as follows:

– XOR the block into the first b bytes of the state.
– XOR 1 into the next byte of the state, position b.
– XOR 1 into the last byte of the state, position 47.

The complete description of Gimli-Hash is given in Algorithm 2. And the Gimli-Hash's security claim against all attacks is 2^{128}.

Algorithm 2 Gimli-Hash

```
Input: $M \in \{0,1\}^*$
Output: Gimli-Hash($\overline{M}$) $= tag \in \{0,1\}^{256}$
  s←0
  $m_1, \cdots m_s \leftarrow = \overline{M} =$ pad($M$)
  for $i$ from 1 to s, do:
      if $i = s$,then:
          $s_{2,3} \leftarrow s_{2,3} \oplus 0x01000000$
      end if
      s←abosrb(s,$m_i$)
  end for
  $h$ ←squeeze(s)
  s = Gimli(s)
  $h \leftarrow h||$squeeze(s)
  return $h$
```

2.4 Ascon

Primitive. For Ascon-Xof and Ascon-Hash, the primitive applies 12 round functions to a 320-bit state. The state is split into five 64-bit words, i.e., $S = x_0||x_1||x_2||x_3||x_4$. The round function employs an SPN-based construction that consists of three operations $p_L \circ p_S \circ p_C$.

- **Addition of Constants** f_C
  This step adds a round constant c_r to word x_2, i.e., $x_2 \leftarrow x_2 \oplus c_r$. The round constants c_r for Ascon-Xof and Ascon-Hash is shown in Table 2.4.
- **Substitution Layer** f_S
  This step updates the state using 64 parallel applications of the 5-bit S-box defined in Table 2.4.
- **Linear Diffusion Layer** f_L
  This step provides diffusion within each 64-bit word x_i.

$$x_0 \leftarrow \sum_0(x_0) = x_0 \oplus (x_0 >>> 19) \oplus (x_0 >>> 28)$$

$$x_1 \leftarrow \sum_1(x_1) = x_1 \oplus (x_1 >>> 61) \oplus (x_1 >>> 39)$$

$$x_2 \leftarrow \sum_2(x_2) = x_2 \oplus (x_2 >>> 1) \oplus (x_2 >>> 6)$$

$$x_3 \leftarrow \sum_3(x_3) = x_3 \oplus (x_3 >>> 10) \oplus (x_3 >>> 17)$$

$$x_4 \leftarrow \sum_4(x_4) = x_4 \oplus (x_4 >>> 7) \oplus (x_4 >>> 41)$$

Table 3. Constants c_r used in the permutation of Ascon-Xof and Ascon-Hash

Round	Constant c_r	Round	Constant c_r
0	000000000000000000f0	6	00000000000000000096
1	000000000000000000e1	7	00000000000000000087
2	000000000000000000d2	8	00000000000000000078
3	000000000000000000c3	9	00000000000000000069
4	000000000000000000b4	10	0000000000000000005a
5	000000000000000000a5	11	0000000000000000004b

Table 4. The 5-bit S-box

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

We denote the state within the i-th round function as follows:

$$S_i \xrightarrow{f_C} S_i' \xrightarrow{f_S} S_i'' \xrightarrow{f_L} S_{i+1}$$

Ascon-Hash and Ascon-Xof. The state of the inner primitive has 320 bits with $r = 64$ and $c = 256$. For Ascon-Hash, the output size is 256 bits; for Ascon-Xof, the output can have arbitrary length.

The hash function initializes the 320-bit state using a constant IV. For Ascon-Xof, $IV = 00400c0000000000$; for Ascon-Hash, $IV = 00400c0000000100$. Then, the permutation is applied to initialize the state S, i.e., $S = f^{12}(IV||2^{256})$.

We regard S as the initial state of the absorbing phase, and its value for Ascon-Hash (left) and Ascon-Xof (right) is:

$$\begin{array}{rl} & ee9398aadb67f03d \\ & 8bb21831c60f1002 \\ S \leftarrow & b48a92db98d5da62 \\ & 43189921b8f8e3e8 \\ & 348fa5c9d525e140 \end{array} \qquad \begin{array}{rl} & b57e273b814cd416 \\ & 2b51042562ae2420 \\ S \leftarrow & 66a3a7768ddf2218 \\ & 5aad0a7a8153650c \\ & 4f3e0e32539493b6 \end{array}$$

The padding process for Ascon-Xof and Ascon-Hash is simple and the same: it appends a single 1 and the smallest number of 0s to M such that the length of the padded message is a multiple of r bits.

The complete description of the hashing process is given in Algorithm 3.

The security claim of Ascon-Hash is 2^{128}. The security claim of Ascon-Xof is $min(2^{128}, 2^{l/2})$, l is the output size.

Algorithm 3 Ascon-Hash and Ascon-Xof

```
Input: $M \in \{0,1\}^*$, output bitsize $l$, $l = 256$ for Ascon-Hash, $l$ is arbitrary for Ascon-Xof.
Output: hash $H \in \{0,1\}^l$
  Initialization
      $S \leftarrow f^{12}(IV||0^c)$
  Absorbing
      $M_1, \cdots, M_s \leftarrow M||1||0^*$
      for $i = 1, \cdots, s$, do:
          $S \leftarrow f^{12}(S_r \oplus M_i)||S_c$
  Squeezing
      for $i = 1, \cdots, t = \lceil l/r \rceil$, do:
          $H_i \leftarrow S_r$
          $S \leftarrow f^{12}(S)$
      return $\lfloor H_1|| \cdots H_t \rfloor_l$
```

2.5 Target Difference Algorithm

The target difference algorithm (TDA) was proposed in [5]. It is a technique that can leverage a $(k-1)$-round differential characteristic that leads to a collision at the output, to a k-round collision of a hash function.

More precisely, the initial state difference of the $(k-1)$-round difference characteristic is called the target difference; the algorithm can output many message pairs that satisfy the target difference after one permutation round. Hence, the algorithm can be used to link the initial state of the internal permutation to the target difference, using one extra permutation round.

2.6 Observations

This section describes the observations used when using the target differential algorithm to derive the k-round differential characteristic from the original $(k-1)$-round one.

Observation 1 *We regard the operations in Equation (1) as the non-linear layer of a Gimli round function.*

$$\begin{cases} xout \leftarrow xin \oplus (zin << 1) \oplus ((yin \wedge zin) << 2) \\ yout \leftarrow yin \oplus xin \oplus ((xin \vee zin) << 1) \\ zout \leftarrow zin \oplus yin \oplus ((xin \wedge yin) << 3) \end{cases} \tag{1}$$

Then, when the input difference of the non-linear layer is active only in the X plane, i.e., $\Delta y_{in} = \Delta z_{in} = 0$, the input value and the output difference satisfy the following constraints:

– *For the output difference:*

$$\begin{cases} \Delta xout = \Delta xin \\ (\Delta yout \oplus \Delta xout)[i] \leq (\Delta xout \ll 1)[i], \\ \Delta zout[i] = (\Delta xin \ll 3)[i], \qquad 0 \leq i \leq 31 \end{cases} \tag{2}$$

– *For the input value:*

$$\begin{cases} if(\Delta xin \lll 1)[i] = 1, (zin \lll 1)[i] \oplus 1 = (\Delta yout \oplus \Delta xout)[i] \\ if(\Delta xin \lll 3)[i] = 1, (yin \lll 3)[i] = \Delta zout[i], 0 \leq i \leq 31 \end{cases} \tag{3}$$

Observation 2 *The ANF of Ascon's 5-bit S-box with input $x_0x_1x_2x_3x_4$ and output $y_0y_1y_2y_3y_4$ is shown in Equation 4.*

$$\begin{cases} y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0, \\ y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0, \\ y_2 = x_4x_3 + x_4 + x_2 + x_1 + 1, \\ y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0, \\ y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1. \end{cases} \tag{4}$$

When the last four input bits are all inactive, i.e., $\Delta x_1 = \Delta x_2 = \Delta x_3 = \Delta x_4 = 0$, the following constraints holds:

– *For the output difference:*

$$\begin{cases} \Delta y_0 \oplus \Delta y_4 = 1, \\ \Delta y_1 = \Delta x_0, \\ \Delta y_2 = 0, \end{cases} \tag{5}$$

– *For the input value:*

$$\begin{cases} x_1 = \Delta y_0 \oplus 1, \\ x_3 \oplus x_4 = \Delta y_3 \oplus 1. \end{cases} \tag{6}$$

3 The Attack Strategy

As described above, Gimli-Hash, Ascon-Hash and Ascon-Xof are all sponge-based hash functions. In this section, we describe two different collision attack strategies aiming at sponge-based hash functions. Assume that the inner primitive is composed of k round functions.

The first strategy utilizes k-round differential characteristics with the input and output difference are both nonzero only in the rate part. We choose the next block pair with difference value equal to the characteristic output difference, then the difference of subsequent states will all be zero. And we can get collisions.

The second strategy is appropriate for hash functions with output size less than or equal to the rate part. It utilizes k-round differential characteristics with the input difference is only active in the rate part and the output difference is inactive in the rate part. The characteristic is applied in the last primitive of the absorbing phase. Thus, the hash output difference is zero, and we can get collisions.

Next we give detailed introduction of these two strategies.

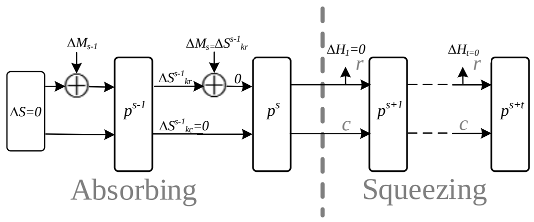

Fig. 2. The 1st attack strategy.

3.1 The First Attack Strategy

The starting point of the first strategy is illustrated in Figure 2. During the absorbing phase, if the output difference ΔS^{s-1}_{kr} of the penultimate primitive p^{s-1} is active only in the rate part, we choose the last message block pair with the same difference, i.e., $\Delta M_s = \Delta S^{s-1}_{kr}$. The input difference ΔS^s_0 of the last primitive p^s will be zero, and all subsequent state difference will also be zero. Thus, we can get collisions.

Concretely, the attack strategy is composed of three steps.

- Find a k-round differential characteristic.
  We add all constraints imported due to Observation 1 or Observation 2 for the input difference and search for a $(k-1)$-round differential characteristic with the output difference is active only in the rate part. We assume its probability is p.
  After that, we can derive a k-round characteristic by linking this characteristic to the initial state of the primitive using the target differential algorithm. Its input and output difference are both active only in the rate part.
- Construct message pairs.
  We choose message pairs that differs with each other only in the last two blocks, i.e., $\Delta M_i = 0$, $1 \leq i \leq s-2$. Each message is composed of s blocks. As shown in Observation 1 and Observation 2, when given the input difference of the origin $(k-1)$-round characteristic, there will be corresponding constraints on the value of S^{s-1}_{0c} and ΔS^{s-1}_{0r}.
  For the value of S^{s-1}_{0c}, we assume that there are X constraints to be satisfied. Then we need at least $\lceil \frac{X}{r} \rceil$ blocks to get a qualified S^{s-1}_{0c}.
  For ΔS^{s-1}_{0r}, we can just choose message pairs that satisfy ΔM_{s-1} is equal to the input difference of the k-round characteristic, as $\Delta S^{s-1}_{0r} = \Delta M_{s-1}$. We also need to set the last message block difference ΔM_s is equal to the output difference of the characteristic.
  Thus, each message has at least $\lceil \frac{X}{r} \rceil + 2$ blocks.
- Search for the collisions.

Firstly we try to find some qualified S_{0c}^{s-1} by randomly choosing the first $\lceil \frac{X}{r} \rceil$ blocks. The filter probability is $\frac{1}{2^X}$.

For each qualified S_{0c}^{s-1}, as the probability of the characteristic is p, on average, we can find a pair that conforms to the difference model of the characteristic using $\frac{1}{p}$ pairs of the $(s-1)$-th block. The block M_{s-1} can provide r-bit degree freedom.

1) If $2^r \geq \frac{1}{p}$, we need compute X values of the first $\lceil \frac{X}{r} \rceil$ blocks and $\frac{1}{p}$ pairs of the $(s-1)$-th block;
2) If $2^r \leq \frac{1}{p}$, we need $\frac{1}{p \cdot 2^r}$ qualified values of S_{c0} in p^{s-1}. Thus we need compute $\frac{2^X}{p \cdot 2^r}$ values of the first $\lceil \frac{X}{r} \rceil$ blocks and 2^r pairs of the $(s-1)$-th block.

After that, we need one last message block pair of which the difference is equal to the output difference of the characteristic. The input difference of the last primitive p^s in the absorbing phase will be zero, and so will be all subsequent state differences. And we get a collision.

3.2 The Second Attack Strategy

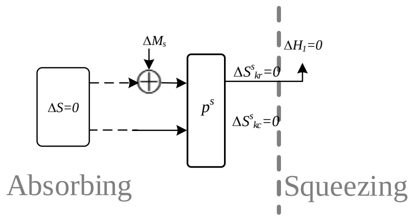

Fig. 3. The 2nd attack strategy.

As illustrated in Figure 3, the second attack strategy is appropriate for sponge-based hash functions if the hash tag can be get by one primitive, i.e., there is no primitives in the squeezing phase. In this case, the hash tag is part of the rate part of the output S_k^s of the last primitive p^s in the absorbing phase. Thus, if we can find some characteristic that has zero difference in ΔS_{kr}^s, we will get a collision.

This strategy is also composed of three steps. As it is very similar with the first strategy in Section 3.1, we only introduce the different points.

- Find a k-round differential characteristic.
  In this step, the output difference of the origin $(k-1)$-round differential characteristic should be inactive in the rate part.
  The input difference of the k-round characteristic we derive from the $(k-1)$-round one is active only in the rate part, and the output difference is inactive in the rate part.
  The k-round differential characteristic is in the last primitive p^s of the absorbing phase.
- Construct message pairs.
  Firstly, the message pairs differs with each other only in the last block. The difference value of the last block is equal to the input difference of the k-round differential characteristic.
  Secondly, because there is no need to use one extra block to eliminate the output difference of the characteristic, in this strategy, each message is composed of at least $\lceil \frac{X}{r} \rceil + 1$ blocks.
- Search for the collisions.
  In this step, the only different point is that each message is composed of at least $\lceil \frac{X}{r} \rceil + 1$ blocks.

4 Collision Attacks on Round-Reduced Gimli-Hash

In this section, following the first strategy in Section 3.1, we derive a 6-round differential characteristic from a 5-round one, and use it mounting a 6-round collision attack on Gimli-Hash.

4.1 The Differential Characteristic

The origin 5-round differential characteristic is shown in Table 5. It spans from the state $S_0^{'}$ after the non-linear layer of the first round to the state S_6 after the 5-th round with probability 2^{-112}.

We first show that the input difference conforms to the constraints in Observation 1. $\Delta S_0^{'}$ is the output difference of the non-linear layer of the first round. The second and fourth column in $S_0^{'}$ have nonzero difference values. Notice that the difference value of these two active columns are equivalent. We only need to check one of them.

The checking process.

$$\begin{aligned}
\Delta xout &= 81ff8980,\\
\Delta yout &= 80618880,\\
\Delta zout &= 0,\\
\Delta yout \oplus \Delta xout &= 019e0100,\\
\Delta xout \lll 1 &= 03ff1300,\\
\Delta xout \lll 3 &= 0ffc4c00.
\end{aligned}$$

Table 5. The differential characteristic of 5-round Gimli.

State	Difference Value				Probability($\log_2$)
	0	81ff8980	0	81ff8980	
$S_0^{'}$	0	80618880	0	80618880	0
	0	0	0	0	
	0	0	0	0	
S_1	0	80618880	0	80618880	-56
	0	81ff8980	0	81ff8980	
	0	42668080	0	42668080	
S_2	0	c0400000	0	c0400000	-40
	0	00011100	0	00011100	
	0	80010080	0	80010080	
S_3	0	00402000	0	00402000	-16
	0	80400080	0	80400080	
	0	00000080	0	00000080	
S_4	0	00400000	0	00400000	0
	0	80000000	0	80000000	
	0	0	0	0	
S_5	0	0	0	0	0
	0	80000000	0	80000000	
	0	80000000	0	80000000	
S_6	0	0	0	0	
	0	0	0	0	

For each i, $0 \leq i \leq 31$, $(\Delta yout \oplus \Delta xout)[i] \leq (\Delta xout \lll 1)[i]$, $\Delta zout[i] \leq (\Delta xout \lll 3)[i]$. Thus, the input difference conforms to the constraints derived from Observation 1.

Derive the 6-round differential characteristic. As we already check that the input difference conforms to the difference constraints in Observation 1, we can use the target differential algorithm deriving a 6-round characteristic. We show the input difference of the 6-round characteristic in Table 6.

Table 6. Input difference of the 6-round characteristic.

State	Difference Value			
	0	ff898081	0	ff898081
S_0	0	0	0	0
	0	0	0	0

4.2 The Attack Process

The bit constraints. In the meanwhile, we can also get the bit constraints on S_{c0} in p^{s-1}.

Concretely, for each active bit $(\Delta xin \lll 1)[i]$, the corresponding bit $(z \lll 1)[i]$ is equal to $(\Delta yout \oplus \Delta xin)[i] \oplus 1$. For each active bit $(\Delta xin \lll 3)[i]$, the corresponding bit $(y \lll 3)[i]$ is equal to $\Delta zout[i]$. We list the constraints of one column in Table 7.

Table 7. Bit constraints on S_{c0} in p^{s-1}.

State	Difference Value	Constraint Number
$\Delta xout << 1$	00000011111111110001001100000000	
$\Delta yout \oplus \Delta xout$	00000001100111100000000100000000	13
$(z \oplus ffffffff) \lll 1$	******0110011110***0**01********	
$\Delta xin << 3$	00011111111110001001100000000000	
$\Delta zout$	00000000000000000000000000000000	13
$y \lll 3$	***0000000000***0**00***********	

* means there is no constrains for the bit value,
0 or 1 means the bit value should be equal to 0 or 1 respectively.

In total, there are $(13 + 13) \times 2 = 52$ constraints for our characteristic. For Gimli-Hash, each block provides 128-bit degree freedom. We only need one message block to get qualified input state values of the capacity part of p^{s-1}.Thus we can find collisions following the strategy in Section 3.2 with messages composed of $(\lceil \frac{52}{128} \rceil + 2) = 3$ blocks.

Search for collisions. The attack process is as follows:

1. Each message is composed of three blocks. The message pair should satisfy three constraints: $\Delta M_1 = (0, 0, 0, 0)$, $\Delta M_2 = (0, ff898081, 0, ff898081)$, $\Delta M_3 = (0, 80000000, 0, 80000000)$.
2. The filter probability of finding a qualified S_{c0} in p^2 is $\frac{1}{2^{52}}$. For Gimli-Hash, each block provides $r = 128$ bits degree freedom. And the probability of our 6-round characteristic p is 2^{-112}. As $2^{128} \geq 2^{112}$, thus, to find a collision, we need 2^{52} the first blocks, 2^{112} the second block pairs and one the third block pair.

The time complexity for finding qualified M_1 is 2^{52}, the time complexity for finding qualified M_2 pair is $2^{112} \times 2 = 2^{113}$, the time complexity for finding qualified M_3 pair is 2.

5 Collision Attacks on Round-Reduced Ascon-Xof and Ascon-Hash

In this section, we present collision attacks on 2-round Ascon-Xof with a 64-bit output and 2-round Ascon-Hash. The attack on 2-round Ascon-Xof follows the second strategy in Section 3.2, it is a practical attack; the attack on 2-round Ascon-Hash follows the first strategy in Section 3.1, it is a practical attack.

5.1 Practical Collisions in 2-round Ascon-Xof

In this section, we present our practical 2-round collision attack on Ascon-Xof with a 64-bit output. Due to the birthday bound, the attack complexity needs to be less than 2^{32}.

We use the difference characteristic in Table 8. It spans the linear layer of the first round and the whole second round function with probability 2^{-14}. The input difference conforms to all constraints imported from Observation 2. After that, we derive a 2-round differential characteristic using the target differential algorithm. Its input difference is shown in Table 9. In this characteristic, $\Delta S_{00}[34]$ [3] is the only active bit in ΔS_{00}. According to Observation 2, the value of $(S_{00}[34], S_{30}[34], S_{40}[34])$ needs to satisfy two constraints. The input and output difference of the corresponding S-box is $(10000, 11000)$, Table 10 shows the corresponding two constraints.

Table 8. The characteristic of the 2-round Ascon-Xof

State	Difference Value	Probability($\log_2$)
S^s_{00}	0000000020000000	
S^s_{10}	0000000020000000	
S^s_{20}	0000000000000000	
S^s_{30}	0000000000000000	
S^s_{40}	0000000000000000	
S_{01}	0000000020000402	
S_{11}	0040000120000000	
S_{21}	0000000000000000	-14
S_{31}	0000000000000000	
S_{41}	0000000000000000	
S_{02}	0000000000000000	
S_{12}	0040000924002412	
S_{22}	00610001b4800000	
S_{32}	0040102100408000	
S_{42}	040000020100040a	

Table 9. The input difference of the 2-round characteristic of Ascon-Xof

State	Difference Value	Active Bit Number
S_{00}	0000000020000000	1
S_{10}	0000000000000000	
S_{20}	0000000000000000	
S_{30}	0000000000000000	
S_{40}	0000000000000000	

[3] We use S_{ij} denoting the i-th word in the j-th round.

Table 10. Constraints for the characteristic in Table 8.

Active Bits	$\varDelta S_{00}[34] = 1$
Constraints	$S_{10}[34] = 0$
	$S_{30}[34] \oplus S_{40}[34] = 1$

The padded message pair need to satisfy $\varDelta M_1 = 0$, $\varDelta M_2 = 0000000020000000$. For Ascon-Xof, each block is composed of 64 bits, i.e., $r = 64$. The number X of constraints is 2. Thus, we can use $\lceil \frac{2}{64} \rceil = 1$ block to get qualified capacity part value in p^1. The probability of the characteristic is 2^{-14}. We can get one block pair that conforms to the 2-round differential characteristic by trying 2^{14} pairs.

On average, we need compute 2^2 first blocks to get a qualified capacity part of the input value, and 2^{14} the second block pairs to get a collision. The time complexity is 2^{15}.

One concrete example of 2-block collision in 2-round Ascon-Xof is given in Table 11.

Table 11. A collision in 2-round Ascon-Xof with a 64-bit output

Message 1	05b93c0000000000fd000000fb
Message 2	05b93c0000000000fd000000db
Output	387fc9cc6fc5e428

5.2 Collision Attacks on 2-round Ascon-Hash

In this section, we use the difference characteristic in Table 12 to mount a non-practical 4-block collision attack on 2-round Ascon-Hash. The attack follows the strategy in Section 3.1.

As shown in Table 12, the input difference conforms to the constraints imported due to Observation 2. Using the target differential algorithm, we derive a 2-round characteristic and its input difference is shown in Table 13.

There are 43 active bits in $\varDelta S_{00}$. According to Observation 2, for all indexes i that $\varDelta S_{00}[i] = 1$, there are two constraints that the value of $S_{10}[i], S_{30}[i], S_{40}[i]$ need to satisfy. In total, there are $43 \times 2 = 86$ constraints. For Ascon-Hash, each block provides 64-bit freedom. We need at least $\lceil \frac{86}{64} \rceil = 2$ blocks to get one qualified S_0 in p^3.

We construct 4-block message pairs that satisfy $\varDelta M_1 = \varDelta M_2 = 0$, $\varDelta M_3 = e6765f2bfb737f78$, $\varDelta M_4 = d255739452530b86$. We need compute 2^{86} values of (M_1, M_2) to get a qualified value S_0 in p^3. The probability of the differential characteristic is 2^{-103}. Thus, on average, we can get one pair that conforms to the differential of the characteristic by computing 2^{103} pairs. For Ascon-Hash, each block is 64 bit and $64 < 103$. We compute $2^{86+103-64} = 2^{125}$ the first two

Table 12. The differential characteristic of one round Ascon-Hash

State	Difference Value	Probability ($\log_2$)
S^s_{00}	00144000c0404000	
S^s_{10}	e6765f2bfb737f78	
S^s_{20}	0000000000000000	0
S^s_{30}	0400000008101000	
S^s_{40}	e6621f2b3b333f78	
S_{01}	0c10400249045804	
S_{11}	8232408ad1246801	
S_{21}	0000000000000000	-103
S_{31}	0c0102000812100c	
S_{41}	8233428ad1366809	
S_{02}	d255739452530b86	
S_{12}	0000000000000000	
S_{22}	0000000000000000	
S_{32}	0000000000000000	
S_{42}	0000000000000000	

Table 13. The input difference of the 2-round differential characteristic

State	Difference Value	Active Bit Number
S_{00}	e6765f2bfb737f78	43
S_{10}	0000000000000000	
S_{20}	0000000000000000	
S_{30}	0000000000000000	
S_{40}	0000000000000000	

block values and get 2^{39} qualified values of S_0 in p^3. Then, for each S_0, we try 2^{64} pairs of the third block. After that, we use one pair of the fourth block and get one collision.

We compute 2^{125} values of the first two blocks, 2^{103} pairs of the third block and one pair of the fourth block. In total, the time complexity is 2^{125}.

6 Conclusion

In this paper, we mount collision attacks on three sponge-based hash functions: Gimli-Hash, Ascon-Xof and Ascon-Hash. Our attacks follow two interesting strategies, one utilizes differential characteristics with the input and output difference is active only in the rate part and the other strategy utilizes characteristics with the input difference is inactive in the capacity part and the output difference is inactive only in the rate part. We search for the differential characteristics using the MILP technique and the target differential algorithm. Finally, we give a practical attack on 2-round Ascon-Xof and non-practical attacks on 6-round Gimli-Hash, 2-round Ascon-Hash.

References

1. The nist lightweight cryptography project, https://csrc.nist.gov/Projects/lightweight-cryptography
2. Bernstein, D.J., Kolbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli 20190329. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/gimli-spec.pdf
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: E-CRYPT hash workshop. vol. 2007. Citeseer (2007)
4. Christoph Dobraunig, Maria Eichlseder, F.M., Schláffer, M.: Preliminary analysis of ascon-xof and ascon-hash. Technique Report (2019)
5. Dinur, I., Dunkelman, O., Shamir, A.: New attacks on keccak-224 and keccak-256. In: International Workshop on Fast Software Encryption. pp. 442–461. Springer (2012)
6. Dobraunig, C., Eichlseder, M., Mendel, F.: Heuristic tool for linear cryptanalysis with applications to caesar candidates. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 490–509. Springer (2015)
7. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR competition: http://ascon. iaik. tugraz. at (2014)
8. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of ascon. In: Cryptographers Track at the RSA Conference. pp. 371–387. Springer (2015)
9. Dwivedi, A.D., Kloucek, M., Morawiecki, P., Nikolic, I., Pieprzyk, J., Wójtowicz, S.: Sat-based cryptanalysis of authenticated ciphers from the caesar competition. IACR Cryptology ePrint Archive **2016**, 1053 (2016)
10. Leander, G., Tezcan, C., Wiemer, F.: Searching for subspace trails and truncated differentials. IACR Transactions on Symmetric Cryptology pp. 74–100 (2018)
11. Li, Y., Zhang, G., Wang, W., Wang, M.: Cryptanalysis of round-reduced ascon. Science China Information Sciences **60**(3), 038102 (2017)
12. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ascon. IACR Transactions on Symmetric Cryptology pp. 175–202 (2017)
13. Tezcan, C.: Truncated, impossible, and improbable differential analysis of ascon. IACR Cryptology ePrint Archive **2016**, 490 (2016)
14. Yosuke, T.: Structural evaluation by generalized integral property. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 287–314. Springer (2015)

Appendix A

Proof of Observation 1

When the input difference is only active in the x plane, the output difference is as follows:

$$\begin{cases} \Delta xout = \Delta xin \\ \Delta yout = \Delta xin \oplus (\Delta xin \cap (z \oplus ffffffff) << 1 \\ \Delta zout = (\Delta xin \cap y) << 3 \end{cases}$$

The constraints in Equation (2) and Equation (3) is are easy to get.

Proof of Observation 2

The relation of the input and the output difference is shown as follows:

$$\begin{cases} \Delta y_0 = \Delta x_0 \cdot (x_1 + 1) \\ \Delta y_1 = \Delta x_0 \\ \Delta y_2 = 0 \\ \Delta y_3 = \Delta x_0 \cdot (x_4 + x_3 + 1) \\ \Delta y_4 = \Delta x_0 \cdot x_1 \end{cases}$$

The constraints in Equation (5) and Equation (6) are easy to get.