New results on Gimli: full-permutation distinguishers and improved collisions

Antonio Flórez Gutiérrez, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, André Schrottenloher, Ferdinand Sibleyras

New results on Gimli: full-permutation distinguishers and improved collisions

Antonio Flórez Gutiérrez, Gaëtan Leurent, María Naya-Plasencia, Léo Perrin, André Schrottenloher, Ferdinand Sibleyras

Inria, France (firstname.lastname@inria.fr)

Abstract. Gimli is a family of cryptographic primitives (both a hash function and an AEAD scheme) that has been selected for the second round of the NIST competition for standardizing new lightweight designs. The candidate Gimli is based on the permutation Gimli, which was presented at CHES 2017. In this paper, we study the security of both the permutation and the constructions that are based on it. We exploit the slow diffusion in Gimli and its internal symmetries to build, for the first time, a distinguisher on the full permutation of complexity 2^{64}. We also provide a practical distinguisher on 23 out of the full 24 rounds of Gimli that has been implemented.
Next, we give (full state) collision and semi-free-start collision attacks on Gimli-Hash, reaching respectively up to 12 and 18 rounds. On the practical side, we compute a collision on 8-round Gimli-Hash. In the quantum setting, these attacks reach 2 more rounds. Finally, we perform the first study of linear trails in the permutation, and we propose differential-linear cryptanalysis that reach up to 17 rounds of Gimli.

Keywords: Gimli, symmetries, symmetric cryptanalysis, full-round distinguisher, collision attacks, linear approximations

1 Introduction

Gimli is a cryptographic permutation that was published at CHES 2017 [5]. It is also the core primitive of a submission to the NIST lightweight cryptography project [6] which is part of the 32 candidates that made it to the second round. It is intended to run well on a vast variety of platforms and contexts, from powerful processors supporting vector instructions to side-channel protected hardware.

A cryptographic permutation is a versatile primitive which is easily used to construct a hash function (as originally intended for this type of object [7]). It was later shown that they can also be used to build authenticated ciphers [10], pseudo-random number generators [9], etc. In all such structures, the security of the cryptographic function relies on the properties of the permutation. In

particular, it is assumed in the underlying security proofs that the permutation used behaves like a permutation picked uniformly at random—apart of course from the existence of a compact implementation, a property which should not be expected from a random object.

By definition, a cryptographic permutation does not have a key. Thus, we cannot define its security level using a game that relies on distinguishing a random permutation from a keyed instance with a random key. Still, since it should behave like a permutation picked uniformly at random, we can assess its security level by trying to identify properties that hold for the permutation studied but which should not be expected for one picked uniformly at random. In this context, cryptanalysts can re-use approaches originally intended for block cipher cryptanalysis (e.g. differential attacks [11]). In fact, given that no key material is involved, we can also borrow techniques from hash function cryptanalysis such as rebound attacks [24].

The aim is usually then to obtain inputs of the permutation satisfying a certain property using an algorithm which is more efficient than the generic one, i.e. the one that would work on a random permutation.

Our Contributions. In this paper, we complete the original security analysis of the designers of Gimli by targeting both the permutation on its own, and the NIST candidate Gimli-Hash. Our results on the permutation are summarized in Figure 1 (plain lines). In order to account for the different costs of the generic attacks, we divided the logarithm of the time complexity of our distinguishers by the logarithm of the time complexity of the corresponding generic distinguisher. In Figure 1, a distinguisher is valid if the ratio is under 1.0. Previous attacks from the literature are represented with dotted lines. The complexities of all our attacks (included those against the hash function) are given in Table 1, along with all the results from the literature we are aware of.

Our main result is a distinguisher of the full 24-round permutation with a cost of 2^{64}, while a similar generic distinguisher has a cost of 2^{96}. We also propose a distinguisher on 23 rounds that is practical, with a cost of 2^{32}, and has been successfully implemented. These distinguishers exploit internal symmetries that are encouraged by the round function. The 23-round distinguisher could be extended by 1 round for free if the rounds were shifted.[1]

Using similar guess-and-determine ideas, we increase to 12 the number of rounds susceptible to collision attacks on Gimli-Hash. A reduced-round version of this attack has been implemented. In the quantum setting, we obtain collisions up to 14 rounds. We also build semi-free start collisions, i.e. we show how to find one internal state value and two different messages (thus not affecting the capacity part) that provide a collision on the capacity after applying the permutation. This attack is more efficient than a generic one for 18 rounds classically, and up to 20 quantumly. As a side note, these results provide a new example where quantum attacks reach more rounds than classical ones, much like in [21].

In addition, we provide the first extensive study of the linear properties of the round function of Gimli, and use them to perform differential-linear distinguishers

[1] This behaviour appears because the linear layer of Gimli is round dependent.

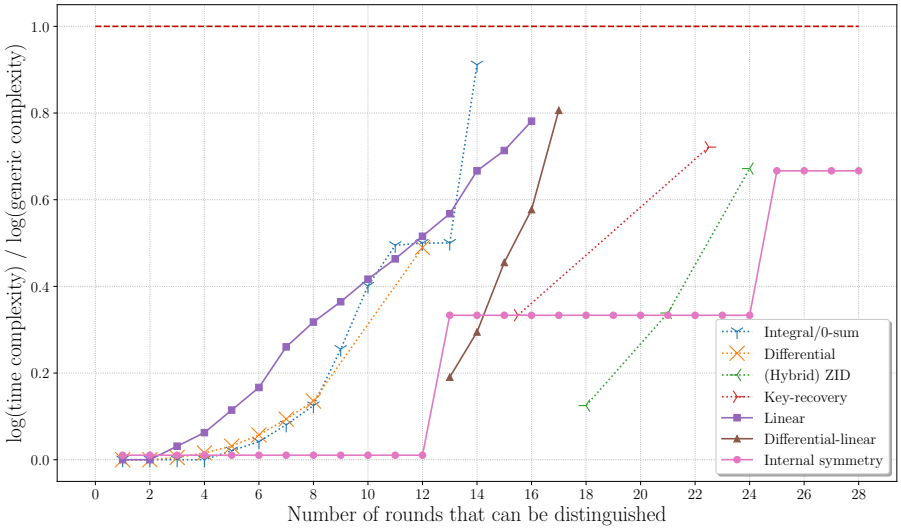

Fig. 1. Comparison of various cryptanalysis techniques. Note that we may consider "shifted" variants of Gimli that do not start at round 24. Dotted lines correspond to results from the literature.

up to 17 rounds. While this distinguisher is less efficient than the one based on internal symmetries, it is the most efficient statistical distinguisher in terms of rounds covered.

Our implementations (23-round distinguisher, reduced-round collision attack, search for linear trails) are available at this URL[2].

Organization of the paper. The organization of the paper is as follows. In Section 2 we provide the description of the Gimli permutation and primitive, as well as previous known results. Section 3 provides the new distinguishers exploiting the internal symmetries that allow to distinguish the full permutation, and to build practical distinguishers up to 23 rounds. Section 4 presents improved collision and semi-free start collision attacks, and Section 5 their quantum counterpart. Section 6 presents our new results regarding statistical distinguishers, with optimal linear trails and new differential-linear attacks. We conclude the paper in Section 7 with a summary, a discussion on the impact of our results and a proposal of tweak that would mitigate their reach.

2 Preliminaries

In this section we describe the Gimli permutation and we provide an overview of previous cryptanalysis results. The Gimli-Hash function is described directly in Section 4.

[2] https://project.inria.fr/quasymodo/files/2020/05/gimli_cryptanalysis_eprint.tar.gz

Table 1. (Quantum) results against algorithms of the Gimli family. Time is counted in evaluations of Gimli, and memory in 128-bit blocks. Attacks that were actually implemented are written in **bold**. ϵ is a term that we only estimated experimentally ($\epsilon \approx 10$, see Section 4). In rounds attacked, $r_1 \to r_2$ means rounds r_1 to r_2 included.

	Technique	Rounds	Time	Memory	Generic	Reference
Distinguishers on the permutation (real rounds: $24 \to 1$)	Key-recovery on Gimli-PRF	$25 \to 2.5$	138.5	128	192	[20]
		15.5	64	64	192	[20]
	Zero-sum	14	351	negl.	384	[14]
	ZID	18	2	negl.	4	[29]
	ZID	21	65	negl.	192	[29]
	ZID	24	129	negl.	192	[29]
	Linear	12	198	negl.	384	Sec. 6.1
	Linear	16	300	negl.	384	Sec. 6.1
	Differential-Linear	15	87.4	negl.	192	Sec. 6.2
	Differential-Linear	16	110.8	negl.	192	Sec. 6.2
	Differential-Linear	17	154.8	negl.	192	Sec. 6.2
	Symmetry	$23 \to 0$	**32**	**negl.**	**96**	Sec. 3
	Symmetry	$27 \to 0$	64	negl.	96	Sec. 3
Preimages on Gimli-Hash	Divide-and-conquer	2	42.4	32	128	[29]
		5	96	65.6	128	[29]
Preimages on Gimli-XOF-128		9	104	70	128	[29]
Collisions on Gimli-Hash	Divide-and-conquer	5	65	–	128	[27]
		3	**practical**	–	**128**	[27]
		6	64	64	128	[28]
	Symmetry	$21 \to 14$	**32** $+ \epsilon$	**negl.**	**128**	**Sec. 4**
	Symmetry	12	$96 + \epsilon$	negl.	128	Sec. 4
	Quantum	14	$64 + \epsilon$	negl.	85.3	Sec. 4
Semi-free start collisions on Gimli-Hash	Symmetry	8	64	negl.	128	[28]
	Symmetry	12	$32 + \epsilon$	negl.	128	Sec. 4
	Symmetry	16	$96 + \epsilon$	negl.	128	Sec. 4
	Symmetry	18	$96 + \epsilon$	64	128	Sec. 4
	Quantum	20	$64 + \epsilon$	64	85.3	Sec. 4

We adopt the following notations in this paper: $\ll, \gg, \lll, \ggg$ represent respectively shift left, shift right, *rotate left* and rotate right operations. x, y, z will denote elements of $\mathbb{F}_2^{32}$. SP is the 96-bit SP-Box. We denote x_i the $(i \mod 32)^{th}$ bit of x ($x_{33} = x_1$) with x_0 least significant (right-most). We denote the output of the SP box as $\mathrm{SP}(x, y, z) = (x', y', z')$ and $\mathrm{SP}^2(x, y, z) = (x'', y'', z'')$.

2.1 The Gimli Permutation

State Structure. We denote by S the 384-bit Gimli state, which is the concatenation of 4 *columns* of 96-bit, that we denote A, B, C, D, where A is column number

Algorithm 1 The full Gimli permutation.

Input: State $S = A, B, C, D$
Output: Gimli(S)

for $r = 24$ downto 1 inclusive **do**
  $A, B, C, D \leftarrow SP(A), SP(B), SP(C), SP(D)$ ▷ SP-Box layer
  if $r \mod 4 = 0$ **then**
    Swap A_x and B_x, swap C_x and D_x ▷ small swap
    $A_x \leftarrow A_x \oplus \mathsf{rc}_r$ ▷ Constant addition
  else if $r \mod 2 = 0$ **then**
    Swap A_x and C_x, swap B_x and D_x ▷ big swap
  end if
end for
Return S

0, and D is column number 3. Each column is cut into three 32-bit *words* x, y, z which are denoted *e.g.* A_x, A_y, A_z. Thus, the state is a $4 \times 3 \times 32$ parallelepiped. We will speak of the x *lane* to denote the sequence or concatenation of words A_x, B_x, C_x, D_x.

SP-Box. The only non-linear operation in Gimli is the SP-Box, which is applied columnwise. On input x, y, z, it updates the three words as follows:

1. Rotate x and y: $x \leftarrow x \lll 24, y \leftarrow y \lll 9$.
2. Perform the following non-linear operations in parallel (shifts are used rather than rotations):
   $x \leftarrow x \oplus (z \ll 1) \oplus ((y \wedge z) \ll 2)$,
   $y \leftarrow y \oplus x \oplus ((x \vee z) \ll 1)$,
   $z \leftarrow z \oplus y \oplus ((x \wedge y) \ll 3)$.
3. Swap x and z: $(x, z) \leftarrow (z, x)$.

Rounds. Gimli applies a sequence of 24 rounds numbered from 24 downto 1 inclusively. Each round applies an SP-Box layer, then performs a *swap* (every two rounds, either a "big swap" or a small "small swap" as shown in Algorithm 1) and a constant addition (every four rounds). The constant at round i, if there is one, will be denoted rc_i in what follows. In Gimli we have: $\mathsf{rc}_i = \mathtt{0x9e377900} \oplus i$. Note that all the attacks studied in this paper are independent of the choice of round constants.

An algorithmic depiction of full Gimli is given in Algorithm 1and it is depicted in Figure 6, where each wire represents a word.

Boolean Description of the SP-Box Now we give a full description of the SP box using Boolean functions:

– for x':

$$\begin{cases} x'_0 = y_{23} + z_0 \\ x'_1 = y_{24} + z_1 \\ x'_2 = y_{25} + z_2 \\ x'_i = y_{i-9} + z_i + x_{i+5} y_{i-12}, \quad 3 \le i \le 32 \ , \end{cases} \tag{1}$$

– for y':

$$\begin{cases} y'_0 = x_8 + y_{23} \\ y'_i = x_{i+8} + y_{i-9} + x_{i+7} + z_{i-1} + x_{i+7}z_{i-1}, \quad 1 \leq i \leq 32 \ , \end{cases} \tag{2}$$

– and for z':

$$\begin{cases} z'_0 = x_8 \\ z'_1 = x_9 + z_0 \\ z'_i = x_{i+8} + z_{i-1} + y_{i-11}z_{i-2}, \quad 2 \leq i \leq 32 \ . \end{cases} \tag{3}$$

Description of the SP^2 Box. If $x'_0 = y_{23} + z_0$ as in Equation (1) then it naturally holds that $x''_0 = y'_{23} + z'_0$ and thus we can use Equations (2) and (3) to get the full formula. Here we write some of them:

$$x'' \begin{cases} x''_0 = & x_8 + x_{30} + x_{31} + y_{14} + z_{22} + x_{30}z_{22} \\ x''_1 = & x_9 + x_{31} + x_0 + y_{15} + z_0 + z_{23} + x_{31}z_{23} \\ x''_2 = & x_{10} + x_0 + x_1 + y_{16} + z_1 + z_{24} + y_{23}z_0 + x_0z_{24} \\ x''_i = & x_{i-2} + x_{i-1} + x_{i+8} + y_{i-18} + z_{i-10} + z_{i-1} + x_{i-2}z_{i-10} + y_{i-11}z_{i-2} \\ & + x_{i-4}y_{i-4} + x_{i-4}z_{i+5} + y_{i-4}y_{i+11} + y_{i+11}z_{i+5} + x_{i-5}z_{i+5} + x_{i-5}y_{i-4} \\ & + y_{i-4}z_{i-13} + z_{i-13}z_{i+5} + x_{i-4}x_{i+10}y_{i-7} + x_{i+10}y_{i-7}y_{i+11} \\ & + x_{i-5}y_{i-4}z_{i-13} + x_{i-5}z_{i-13}z_{i+5} + x_{i-5}x_{i+10}y_{i-7} + x_{i+10}y_{i-7}z_{i-13} \\ & + x_{i-5}x_{i+10}y_{i-7}z_{i-13}, \ i \neq 0, 1, 2, 9, 12, 27, 28, 29 \mod 32 \end{cases} \tag{4}$$

$$y'' \begin{cases} y''_0 = x_{30} + x_{31} + y_{14} + y_{31} + z_8 + z_{22} + x_{13}y_{28} + x_{30}z_{22} \end{cases} \tag{5}$$

$$z'' \begin{cases} z''_0 = y_{31} + z_8 + x_{13}y_{28} \\ z''_1 = x_8 + y_0 + z_9 + x_{14}y_{29} \end{cases} \tag{6}$$

The 2-round probability 1 linear relation $x''_0 + y''_0 + z''_0 = x_8$ follows.

2.2 Previous work

We provide here a brief overview of the main previous third-party results of cryptanalysis against either the permutation or the NIST candidate Gimli. Notice that all the cryptanalysis previously considered were classical attacks, while in this paper, we will also give quantum attacks on reduced-round Gimli-Hash. Let us point out that no search of linear trails was done prior to our work.

Zero-sum permutation distinguishers on 14 rounds. In [14], Cai, Wei, Zhang, Sun and Hu present a zero-sum distinguisher on 14 rounds of Gimli. This distinguisher uses the inside-out technique and improves by one round the integral distinguishers given by the designers.

Structural permutation distinguisher on 22.5 rounds. In [20], Hamburg proposed the first third-party cryptanalysis of the Gimli permutation, providing distinguishers on reduced-round versions of the permutation. This analysis does not depend on the details of the SP-Box, and is based only on the slow diffusion of Gimli. Thus, it follows a similar path as the distinguishers of Section 3. In his work, Hamburg defines a PRF with 192-bit input x and 192-bit key k that computes $F(k, x) = \text{trunc}_{192}(\text{Gimli}(k\|x))$. He gives a distinguishing attack in time 2^{64} for 15.5 rounds (omitting the final swap), and a key-recovery attack on F when using 22.5 rounds of Gimli, precisely rounds 25 to 2.5 (omitting again the final swap). This attack runs in time $2^{138.5}$ with a memory requirement of 2^{129}, which is faster than the expected 2^{192}, and thus shows that 22.5-round Gimli behaves differently than what could be expected from a random permutation.

Hamburg's attacks are based on a meet-in-the-middle approach, exploiting the slow diffusion by tabulating some of the values that are passed from an SP-Box to another. The 15.5-round distinguisher relies on a table of size 2^{64}, and the 22.5-round attack on a table of size 2^{128}. None of these attacks are practical.

ZID Permutation Distinguishers. In an independent and simultaneous work posted very recently on ePrint [29], Liu, Isobe, and Meier present a "hybrid zero-internal differential" (ZID) distinguisher on full Gimli, which extends a ZID distinguisher of previous unpublished work. The basic ZID distinguisher happens to be what we call an *internal symmetry distinguisher*, where states with symmetries are produced in the input and in the output of a reduced-round variant of Gimli. A "hybrid" one adds a limited birthday-like property (which is absent from our distinguishers). The steps that they take are however different from ours, as this distinguisher only spans 14 rounds. Compared with our analysis in Section 3, they will actually start from a much more constrained middle state, which limits the number of rounds by which one can extend the distinguisher afterwards (or significantly increases the complexity). In contrast, we complete the middle state in multiple successive steps, each step ensuring that more rounds will be later covered.

Collisions and Preimages on Gimli-Hash. In [32], Zong, Dong and Wang study Gimli among other candidates of the competition. They present a 6-round collision attack on Gimli-Hash of complexity 2^{113}, using a 6-round differential characteristic where the input and output differences are active only in the rate. This differential characteristic was invalidated in [28].

In [27], [29] and [28] Liu, Isobe and Meier give collision and preimage attacks on reduced-round Gimli-Hash. Their attacks rely on divide-and-conquer methods, exploiting the lack of diffusion between the columns, as did Hamburg, but they also rely on SP-Box equations in order to attack the hash function itself. These equations are different from those that we will solve in Section 4, and they mostly relate the input and outputs of a single SP-Box, whereas we study directly two SP-Boxes. Their analysis is also much more precise, since they prove running times of solving these equations.

After giving a meet-in-the-middle generic preimage attack of time and memory complexity 2^{128}, which sets a bound against the sponge construction used

in Gimli-Hash, they give practical preimage attacks on 2-round Gimli-Hash and practical collision attacks on 3-round Gimli-Hash. They give a collision attack on 5-round Gimli-Hash with a time complexity 2^{65} and a second preimage attack with time complexity 2^{96}. They give in [29] a preimage attack on 5-round Gimli-Hash. In [28], they give a semi-free start collision attack on 8 rounds and a state-recovery attack on the AE scheme for 9 rounds.

3 Internal Symmetry Distinguishers against Gimli

In this section we present new distinguishers on the Gimli permutation. Our distinguishers improve upon the best previously known ones, reaching the full 24-round permutation. They are practical on 23 rounds and have been implemented. The results presented in this section do not exploit the specifics of the SP-Box: they would work equally well if the SP-Box was replaced with a permutation picked uniformly at random. Like all the other analyses presented in this paper, they do not depend on the values of the round constants.

Our distinguishers rely on internal symmetries. The general idea consists in identifying a specific form of symmetry (formally, a vector space) that is preserved by the round function under some circumstances, and then trying to craft an input for the permutation such that this symmetry traverses all the rounds so that the output has the same type of property.

In our case, we formalize the symmetry using the notion of *2-identical* states.

Definition 1 (2-identical states). *A state S is* 2-identical *if $B = D$, if $A = C$, or if one of these properties holds up to a swap and a constant addition.*

Our *internal symmetries distinguisher* aims at finding a 2-identical input that is mapped to a 2-identical output. Since there are 96 bits of constraint, a generic algorithm returning such an input should run in time 2^{96} by evaluating the permutation on a set of inputs satisfying the property until the output matches it by chance. Our aim is to find more efficient algorithms in the case of Gimli.

This definition is similar to the one used in [15]. In fact, an internal symmetry distinguisher can be seen as a stronger variant of a *limited birthday distinguisher* of the type used in [15]. Indeed, we can build a limited birthday pair using our distinguisher: by producing a pair of inputs S, S' satisfying the internal symmetry property, we obtain $S \oplus S' \in V_{in}$ and $\Pi(S) \oplus \Pi(S') \in V_{out}$. Further, since the converse is not true, an internal symmetry distinguisher is *strictly* stronger.

From now on, S^i denotes the Gimli state before round i.

3.1 23-round Practical Distinguisher

We design an internal symmetry distinguisher on 23 rounds of Gimli, that is represented in Figure 2, running in time equivalent to 2^{32} evaluations of Gimli on average. Algorithm 2 starts from a symmetric state in the middle and completes

Algorithm 2 23-round internal symmetry distinguisher.

Output: a 2-identical state S such that $\mathsf{Gimli}(23,1)(S)$ is 2-identical
We start from the middle. We will be interested in the state S^{11}.

1. Select $A_x^{15}, A_y^{15}, A_z^{15}$ and $C_x^{15} = A_x^{15} \oplus \mathsf{rc}_{16}$, $C_y^{15} = A_y^{15}$, $C_z^{15} = A_z^{15}$ such that $B_x^{11} = D_x^{11}$.
   Notice that due to the small swap operation, the values B_x^{11} and D_x^{11} actually come from A and C and depend only on A^{15} and C^{15}. At this point, we have ensured that for any values of $B^{15} = D^{15}$:
   - S^{23} is 2-identical: indeed, A and C will remain identical from rounds 16 to 19 backwards. Then, the small swap backwards injects the same value in A and C since B and D are also identical. Thus, $A^{23} = C^{23}$.
   - S^7 is 2-identical: indeed, since $B_x^{11} = D_x^{11}$, B and D remain equal until the SP-Box layer of round 8, and the 2-identical property remains after the small swap of round 8.

   Once good values have been found, we can compute part of the state S^{11}: $A_{y,z}^{11}$, $C_{y,z}^{11}$, and $B_x^{11} = D_x^{11}$ are fixed. The rest remains free.
2. Select $A_x^{11} = C_x^{11} \oplus \mathsf{rc}_{12}$ such that $B_x^7 = C_x^7$. At this point, the two-identicality of the output state is preserved through 4 more rounds (until round 4 included): S^3 is 2-identical.
   In the state S^{11}, $B_{y,z}^{11} = D_{y,z}^{11}$ remain free.
3. Select $B_{y,z}^{11} = D_{y,z}^{11}$ such that $B_x^3 = C_x^3$. Thus, the output S^0 is 2-identical.

the state S^{11} in three steps. Each step assigns a value to more words of the state, and ensures that the 2-identical symmetry property traverses more rounds.

Each step of Algorithm 2 requires to evaluate a few SP-Boxes 2^{32} times (we do not even need to evaluate the inverse SP-Box). The total amount of computations is smaller than 2^{32} evaluations of 23-round Gimli. Notice also that the algorithm uses only a small amount of memory. Our implementation of Algorithm 2 ran in less than one hour on a regular laptop.

The time complexity of the algorithm can be computed as follows: 8×2^{32} SP-Box evaluations for the first step, 8×2^{32} for the second and 16×2^{32} for the third, meaning a total of $8 \times 2^{32} + 8 \times 2^{32} + 16 \times 2^{32} = 40 \times 2^{32}$ which is less than 2^{32} evaluations of 23-round Gimli (each of them consisting essentially of 92 SP-Box evaluations). This complexity is to be compared to that of the generic algorithm for obtaining our internal symmetry property, which costs 2^{96}.

Below, we provide an example of input-output pair that we obtained, with a 2-identical input S that remains 2-identical after $\mathsf{Gimli}(23,1)$:

```
        7f9fcf70 6aedf7e6 7f9fcf70 cb2f0e6a
Input:  0ba2f1f9 f339b619 0ba2f1f9 f70cf15c
        b2ee8259 df0b4801 b2ee8259 3856106d
        a8ef848d 8c17b743 9615b3bc 8c17b743
Output: 541122c5 30530879 8d9d5d30 30530879
        74b6dbe6 18885a6e 744b55c1 18885a6e
```

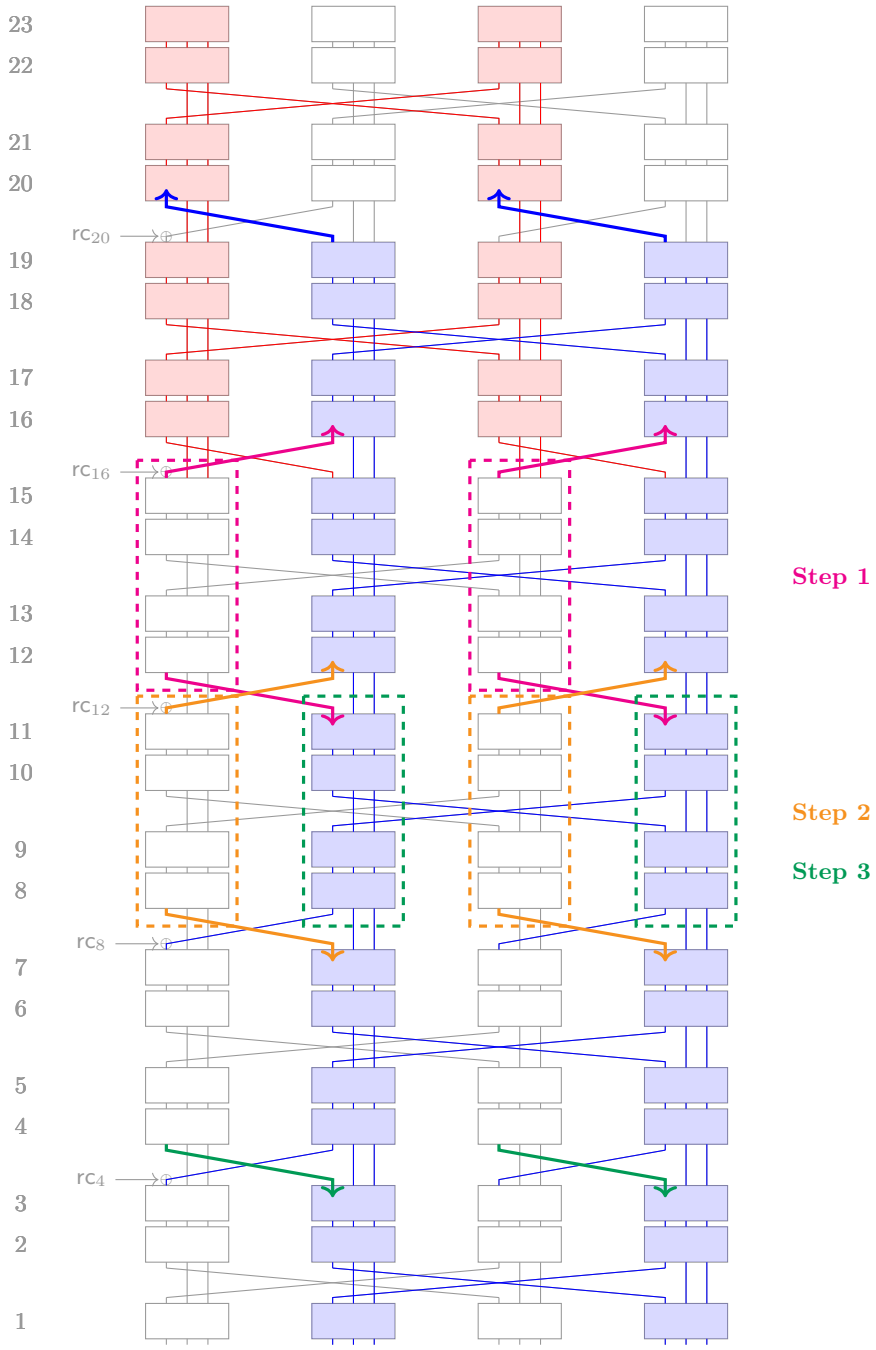

Fig. 2. Distinguisher on 23 rounds. The same color for symmetric branches or columns at a given round means that they are equal.

3.2 Distinguisher on full Gimli and Extensions

Here we will describe how to extend the 23-round distinguisher to the full Gimli permutation, and even to more rounds. All these results are summarized in Figure 1 from Section 1. An extension of our distinguisher to the full Gimli is a trivial matter. Indeed, after running Algorithm 2, we obtain a 2-identical input state $S^{23} = A^{23}, B^{23}, C^{23}, D^{23}$ with $A^{23} = C^{23}$. Then, if $B_x^{23} = D_x^{23}$, which is a 32-bit condition, the state remains 2-identical after the inverse round 24. By repeating the previous procedure 2^{32} times, we should find an input value that verifies the output property. The generic complexity of finding a 2-identical input that generates a 2-identical output is still 2^{96}. Thus, full Gimli can be distinguished in time less than $2^{32+32} = 2^{64}$ full Gimli evaluations, and constant memory.

An interesting question is: how many rounds of a Gimli-like permutation can we target? The distinguisher works mainly because the diffusion in Gimli is somewhat slow. Thus, a possible fix would be to increase the number of swaps, for example by having one in each round instead of every two rounds. An attack exploiting this behaviour that worked previously for r rounds would now a priori work for $r/2$ rounds only. Of course, the details of the SP-box could allow further improvement of these results given that a single iteration would now separate the swaps rather than a double.

Extending to 28 Rounds. It is trivial to adapt this distinguisher to an extended version of Gimli with more rounds. The 2-identicality of S^0 is preserved after one round since the next round would apply only an SP-Box layer and a small swap. Similarly, the 2-identicality of S^{24} is preserved after 3 more inverse rounds since the next swap operation is a big swap which exchanges data between A and C only. Thus, our practical distinguisher works against Gimli$(23, 0)$ (a 24-round version of Gimli shifted by one round), and our extended distinguisher works against Gimli$(27, 0)$ (a 28-round version of Gimli).

4 Classical Collisions on Reduced-Round Gimli-Hash

In this section, we describe collision attacks on Gimli-Hash when it is instantiated with a round-reduced variant of Gimli. Table 2 summarizes our results.

4.1 The Gimli-Hash Function

This function is built using the Gimli permutation in a sponge construction [8], represented in Figure 3.

Gimli-Hash (Algorithm 5) initializes the Gimli state to the all-zero value. The message is padded and separated into blocks of size $r = 128$, which corresponds to the rate r, introducing message blocks of 128 bits between two permutation applications by XORing them to the first 128 bits of the state. Once all the padded message blocks are processed, a 32-byte hash is generated by outputting

Table 2. Collision attacks on round-reduced Gimli

Type	Nbr of rounds	Time complexity	Memory complexity
Standard	8	$8 \times 2^{32} \times t_e$ (practical)	negl.
Standard	12	$8 \times 2^{96} \times t_e$	negl.
Quantum	14	$\simeq 8 \times 2^{64} \times t_e$	negl.
Semi-free start	12	$10 \times 2^{32} \times t_e$	negl.
Semi-free start	16	$10 \times 2^{96} \times t_e$	negl.
Semi-free start	18	$7 \times 2^{96} \times t_e$	2^{64}
Semi-free start	18	2^{96}	2^{96}
Semi-free start, quantum	20	$\simeq 2^{64} \times 10 \times t_e$	2^{64}

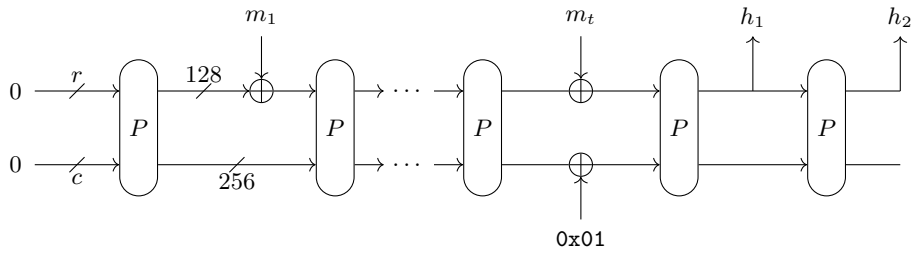

Fig. 3. Gimli-Hash (P stands for the Gimli permutation). The rate is $r = A_x, B_x, C_x, D_x$. The capacity is $c = A_{y,z}, B_{y,z}, C_{y,z}, D_{y,z}$.

16 bytes of the internal state, applying once more the permutation, and outputting 16 additional ones. In Gimli-Hash, the rate is $r = A_x, B_x, C_x, D_x$ and the capacity is $c = A_{y,z}, B_{y,z}, C_{y,z}, D_{y,z}$.

We will consider two kinds of collision attacks:

- Full-state collision attacks: we will build pairs of two-block messages M_0, M_1 and M_0, M_1' such that the state after absorbing these pairs becomes again equal. Thus, one can append any sequence of message blocks after this and obtain the same hash.
- Semi-freestart collision attacks: we will build pairs of (384-bit) states S, S' such that S differs from S' only in a single x, and after r rounds of Gimli, $\pi(S)$ and $\pi(S')$ differ only in a single x as well. This does not yield a collision on the hash function as we would need to choose the value of the same initial state; however, it represents a vulnerability that may be used in the context of the Gimli modes of operation. For example, in Gimli-cipher, the initial state contains a key of 256 bits and a nonce of 128 bits which is put in the x values. Then each block of plaintext is handled in the same way as Gimli-hash. Thus, by XORing the right values before and after π, one can create a key, a nonce and a pair of messages which yield the same tags.

4.2 SP-Box Equations and How to Solve Them

All collision attacks in this section exploit the slow diffusion of Gimli and the simplicity of the SP-Box (contrary to the distinguishers on the permutation, which worked regardless of the SP-Box used). In this section, we describe a series of "double SP-Box equations"; solving them will be the main building block of our attacks. We define the following equations.

$$\text{Given } y, z, \text{ find } x \neq x' \text{ such that } SP^2(x, y, z)_x = SP^2(x', y, z)_x \quad . \tag{7}$$

$$\text{Given } y, z, y', z', \text{ find } x \text{ such that } SP^2(x, y, z)_x = SP^2(x, y', z')_x \quad . \tag{8}$$

$$\text{Given } y, z, y', z', \text{ find } x \text{ such that } SP^2(x, y, z)_z = SP^2(x, y', z')_z \quad . \tag{9}$$

$$\text{Given } y, z, x', \text{ find } x \text{ such that } SP^2(x, y, z)_x = x' \quad . \tag{10}$$

Number of Solutions. Except Equation (7), all these equations have *on average*, when the inputs are drawn uniformly at random, a single solution. However, the variance on the number of solutions depends on the equation considered. For example, only approx. 6.2% of inputs to Equation (8) have a solution, and they have on average 82.4 solutions each. Equation (10) gives a little more than 1.5 solutions. This variance is not a problem for us, as long as we can produce efficiently all solutions of the equations, which remains the case. In order to simplify our presentation, we will do as if equations (8), (9) and (10) always gave exactly a single solution for each input.

Solving the Equations. We use an off-the-shelf SAT solver [31]. In some cases, more time seems spent building the SAT instance rather than solving it, and we believe that our current implementation is highly unoptimized.

The solver allows us to retrieve all solutions of a given equation (we treat Equation (7) differently because it has on average 2^{32} of them). Let us consider the average time to produce a solution when random inputs are given. On a standard laptop, this time varies between approximately 0.1 milliseconds (Equation (8)) and 1 millisecond (Equation (10)). This difference mainly stems from the fact that Equation (8) often has no solutions, and that the solver quickly finds a counterexample, while Equation (10) practically always has solutions that must be found.

On the same computer, an evaluation of the full Gimli permutation (not reduced-round) takes about 1 microsecond, so there is approximately a factor 1000 between computing Gimli and solving a double SP-Box equation.

We consider that all equations have approximately the same complexity and introduce a factor t_e that expresses the time taken to solve them in number of evaluations of Gimli or a reduced-round version (depending on the studied case).

4.3 Practical 8-round Collision Attack

We consider 8 rounds of Gimli, *e.g.* rounds 21 to 14 included, and name Gimli(21, 14) this reduced-round permutation. We omit the last swap, because it has no incidence (it only swaps x values). The situation is represented on Figure 4. As before, we name S^i the partial state immediately before round i.

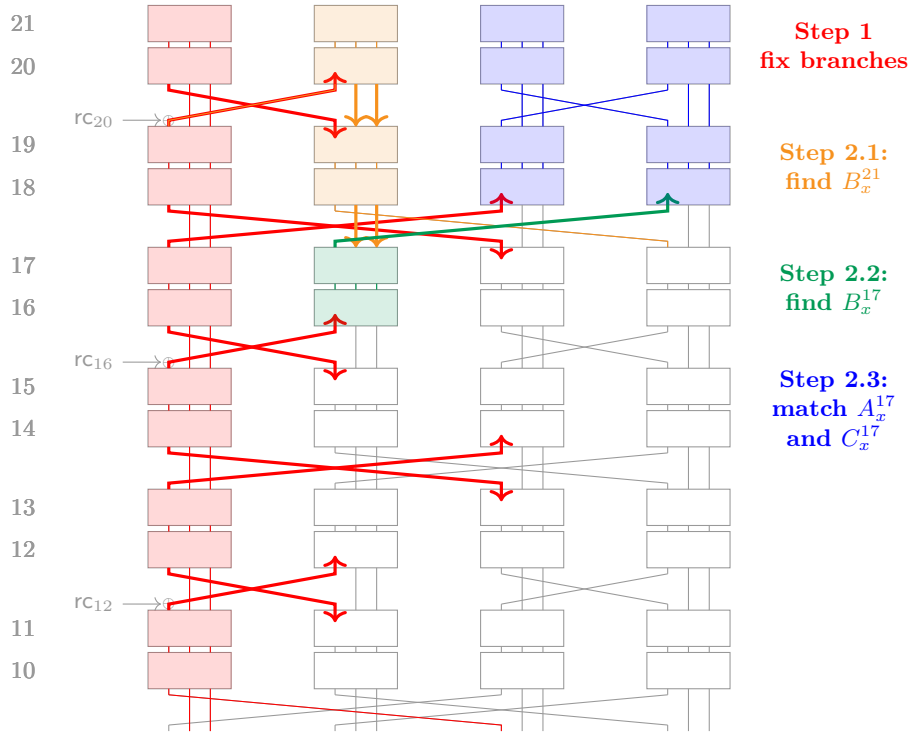

Fig. 4. Collision attack on 8 rounds of Gimli, extended to 12 rounds. The first step fixes the branches in red, which have equal values for the two inputs $A_x^{21}, A_x'^{21}$. Then we find values of $B_x^{21}, C_x^{21}, D_x^{21}$ that will conform to these branches. Then, the whole states are deduced. The branches A_x^{13} and A_x^{11} remain to match.

Algorithm 3 finds on average a single solution, with any input state. There is some variance on the number of solutions, that is induced by the SP-Box equations, but it is small in practice. Furthermore, we can eliminate the memory requirement by solving Equation (7) for many input random states. Starting from a given state, it suffices to apply one more Gimli permutation with a random message block, in order to re-randomize the input.

Remark that if we omit the second step then we already have a semi-free-start collision attack, because we can reconstruct the inputs C^{21} and D^{21} immediately from the middle.

Practical Application: first step. In our practical computations, we considered rounds 21 to 14 included. We solved step 1, starting from $0, 0, 0, 0$ and using a random message $m_1, 0, 0, 0$ to randomize the first block. We also solved at the same time the two Equations (10) that enabled us to go back to A_x^{17}, B_x^{17}.

We had to produce $15582838652 \simeq 2^{33.86}$ solutions for Equation (7) until we found a solution for Step 1 *and* for both equations. We verified experimentally that each solution for Equation (7) yielded on average a solution for the final

Algorithm 3 8-round collision attack.

Input: an input state $A^{21}, B^{21}, C^{21}, D^{21}$.
Output: values $A_x^{21}, A_x'^{21}, B_x^{21}, C_x^{21}, D_x^{21}$ such that by putting $A_x^{21}, B_x^{21}, C_x^{21}, D_x^{21}$ and $A_x'^{21}, B_x^{21}, C_x^{21}, D_x^{21}$ respectively in the rate, after $\mathsf{Gimli}(21, 14)$ (without the last swap), the state differs only on A_x.
Complexity: $7 \times 2^{32} \times t_e$ time and 2^{32} memory *or* $8 \times 2^{32} \times t_e$ and negligible memory.

The attack runs in two main steps, both of which must solve 2^{32} times a sequence of SP-Box equations.

Step 1: find good $A_x^{21}, A_x'^{21}$.

1. Find all pairs $A_x^{21}, A_x'^{21}$ such that the branch B_x^{19} collides (there are 2^{32} such pairs, that can be found in time 2^{32}).
2. For each pair, compute $A_y^{19}, A_z^{19}, A_y'^{19}, A_z'^{19}$ and solve the SP-Box equation (8): find A_x^{19} such that the branch C_x^{17} collides (there is on average one solution)
3. Given this value, compute $A_y^{17}, A_z^{17}, A_y'^{17}, A_z'^{17}$ and solve the SP-Box equation (8) again: find A_x^{17} such that the branch B_x^{15} collides (there is on average one solution)
4. Given these values, compute $A_y^{15}, A_z^{15}, A_y'^{15}, A_z'^{15}$ and solve Equation (9): find A_x^{15} such that A_z^{13} and $A_z'^{13}$ collide.

Since we do that 2^{32} times, we expect on average a single solution such that A_y^{13} and $A_y'^{13}$ also collide.
Now that we have found $A_x^{21}, A_x'^{21}$, it remains to find $B_x^{21}, C_x^{21}, D_x^{21}$ that give the wanted $A_x^{19}, A_x^{17}, A_x^{15}$ (in red on Figure 4). We expect on average a single solution, and little variation on the number of solutions, as only Equation (10) is involved.

Step 2: find $B_x^{21}, C_x^{21}, D_x^{21}$.

1. Find B_x^{21} by solving Equation (10), given the input y and z, and the output x wanted. Deduce the values of B_y^{17}, B_z^{17}
2. Given B_y^{17}, B_z^{17}, and A_x^{15}, solve Equation (10) again to get B_x^{17}.
3. Now find C_x^{21}, D_x^{21} that lead to the wanted A_x^{17}, B_x^{17}. First guess the value of C_x^{21}, deduce C_y^{19}, C_z^{19} and with $C_y^{19}, C_z^{19}, A_x^{17}$, solve Equation (10) to obtain C_x^{19}. Next, given D_y^{21}, D_z^{21} and C_x^{19}, solve Equation (10) to obtain D_x^{21}. Deduce a value for B_x^{17} and check if it matches what we want; we expect to find a match after trying all 2^{32} guesses for C_x^{21}.

equation. We obtained in total 5 solutions (Table 3). There are two different solutions for $A_x^{15} \oplus \mathsf{rc}_{16}$, which yield two and three solutions respectively for B_x^{17}. The total computation ran in less than 5000 core-hours. It was easy to run on many concurrent processes as this algorithm is trivial to parallelize.

Practical Application: second step. We solved step 2, that is, looking for C_x^{21}, D_x^{21} that lead to one of the pairs A_x^{17}, B_x^{17}. This step was much faster than the previous one, although it ought to have the same complexity: this is because we paid in step 1 the probability to find a solution (twice) in Equation (10), while in step 2 we benefited from having 5 different possible solutions. We found two solutions: $C_x^{21}, D_x^{21} =$ `819b1392, 9f4d3233` and $C_x^{21}, D_x^{21} =$ `aa9f6f2d, 3a6e613a`.

Table 3. Results of the first step

m_1	A_x^{21}	$A_x'^{21}$	$A_x^{19} \oplus \mathsf{rc}_{20}$	A_x^{17}	B_x^{21}
dc84bf38	bbdb41f3	1b1da6e4	07f25303	f793fb5f	aae48b72

$A_x^{15} \oplus \mathsf{rc}_{16}$	B_x^{17}	$A_x^{15} \oplus \mathsf{rc}_{16}$	B_x^{17}
ddfbc88b	92f536b6	ddfbc803	f72044db
ddfbc88b	0d9605fe	ddfbc803	b1c91a60
		ddfbc803	55d2252a

Putting both Steps Together. With these solutions, we built two collisions on 8-round Gimli$(21, 14)$. We start from $m_1, 0, 0, 0$, then after one round, we inject the values $A_x^{21}, B_x^{21}, C_x^{21}, D_x^{21}$ and $A_x'^{21}, B_x^{21}, C_x^{21}, D_x^{21}$ respectively in the rate; then we obtain two states that differ only on the x-coordinate of the third column (not the first, due to a big swap), and we inject two different blocks to cancel out this difference, obtaining the same state. The full state then collides, and we can append any message block that we want. The two collisions are given in Table 4.

Table 4. Two 8-round collisions on Gimli-Hash

Starting state (first message block)
dc84bf38 00000000 00000000 00000000||dc84bf38 00000000 00000000 00000000
Second message block
bbdb41f3 4333192c bc17e444 8a9d06c7||1b1da6e4 4333192c bc17e444 8a9d06c7
Third message block
00000000 00000000 00000000 00000000||00000000 00000000 afad801e 00000000
Starting state (first message block)
dc84bf38 00000000 00000000 00000000||dc84bf38 00000000 00000000 00000000
Second message block
bbdb41f3 4333192c 971398fb 2fbe55ce||1b1da6e4 4333192c 971398fb 2fbe55ce
Third message block
00000000 00000000 00000000 00000000||00000000 00000000 afad801e 00000000

Extending the Attack. Remark that the first step can be extended to span any number of SP^2-boxes. However, each time we add two more rounds, there is one more branch coming from the B, C, D states which has to match an expected value, so we add a factor 2^{32} in complexity. Since $t_e \ll 2^{32}$, we can do that twice before meeting the bound 2^{128}. Thus, a collision on 12-round Gimli-Hash can be built in time $2^{96} \times 4 \times t_e$.

4.4 Semi-free Start Collisions on Reduced-round Gimli

We will now design semi-free start collision attacks based on the same principle. This time, our goal is to obtain two input states S, S' that differ only in the rate (in practice, only in A_x) and such that after applying a reduced-round Gimli, the output states differ only in the rate (the x values). They can also be seen

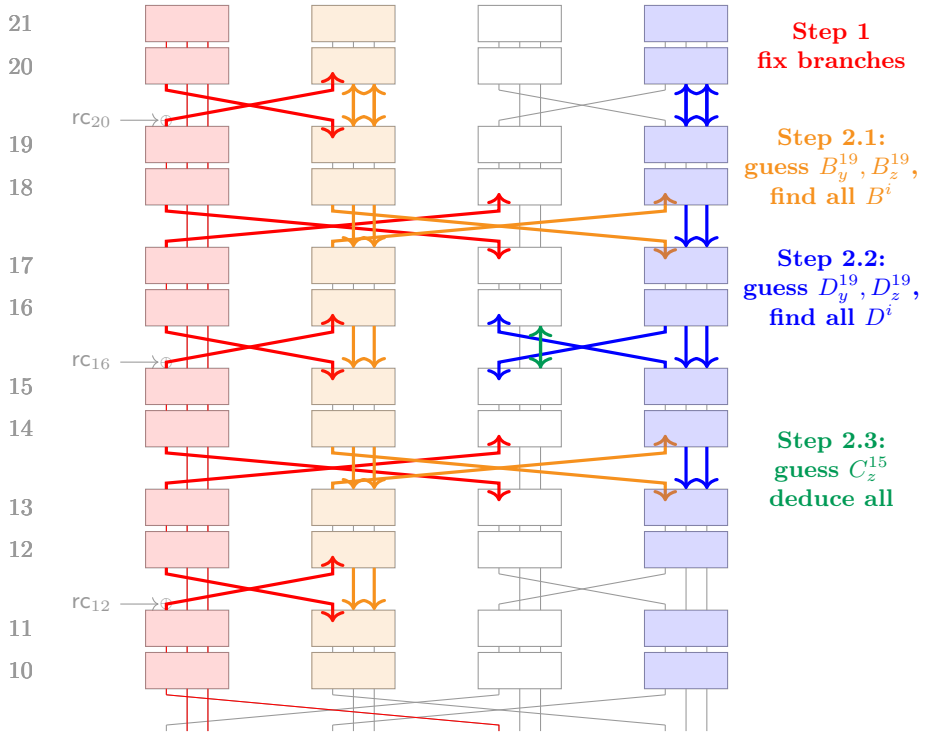

Fig. 5. Semi-free start collision attack on 12 rounds of Gimli (see Algorithm 4).

as finding one state and two pairs of 2-block messages such that after inserting both messages we obtain a collision. The previous "first step" remains the same, with an extension to whichever number of rounds we are targeting. The "second step" is changed, because we can now choose completely the columns B, C, D, *e.g.* by starting from the middle instead of having to choose only the input rate.

Doing this allows us to reach 4 rounds more for the same cost as before, as outlined on Figure 5 and Algorithm 4. We can then append new rounds as before, reaching 16 rounds classically in time $2^{96} \times 10 \times t_e$.

Another Improvement using Precomputations. We are going to win a factor 2^{32} using $2^{64} \times t_e$ precomputations and a table of size 2^{64}. This way, we can attack two more rounds. Indeed, once we have computed the first step, the two branches C_x^{17} and A_x^{13} contain arbitrary fixed values. Then, when we try to find the right C, we could have a table that for all C_y^{15}, C_z^{15}, gives all input-output values for C^{17} and C^{14}, and we could directly use this table to match the values C_x^{15} and D_x^{15} that come from D (instead of having to make a guess of C_z^{15}.

Let us fix $C_x^{17} = A_x^{13} = 0$. Thus, we repeat step 1 in Algorithm 4 a total of 2^{64} times in order to have $C_x^{17} = A_x^{13} = 0$. Step 1 now costs $2^{96} \times t_e$.

The table that we precompute shall contain: for each x', x'', all values (on average 1) of y', z' such that $SP^2(0, *, *) = x', y', z'$ and $SP^2(x'', y', z') = 0, *, *$.

Algorithm 4 12-round semi-free start collision attack (see Figure 5).

Input: an initial A (can be given)
Output A_x, A'_x, B, C, D such that after $\mathsf{Gimli}(21, 10)$, only the rate differs.

As before, we don't write the last swapping step.

Step 1: Same step as in Algorithm 3, extended to 12 rounds. It gives a total of 10 32-bit branches (input values) that are required, that are represented in red on Figure 5.

Step 2: we will start from the middle.

1. We take an arbitrary value for $B^{19}_{y,z}$. This guess enables to deduce all values of the column B, from B^{21} to B^{10}, either by simply computing the SP-Box, or by solving Equation (10) (given two input branches y, z, given the output x, deduce the input x). From this, we deduce the value in all branches that go from B to D on the figure, hence 4 branches. They are represented in orange on Figure 5.
2. We take an arbitrary value for $D^{19}_{y,z}$. Again, this enables to deduce the whole sequence of states from D^{20} to D^{10}, either by computing the SP-Box when possible, or by finding the input x value corresponding to a given output. We also obtain the values of branches that are transmitted from D to C.
3. We now guess C^{15}_z. Given this, and C^{15}_x, and the output A^{13}_x that must be met, we obtain the whole state by solving another simple SP-Box equation (which is not Equation (10), but has a similar form).
4. Having deduced C^{15}, we have only 2^{-32} chances of obtaining the right C^{17}_x, so we have to repeat all of this 2^{32} times.

In total, we have to solve 5 SP-Box equations, 2^{32} times, in both steps, so the time complexity is $2^{32} \times 10 \times t_e$.

Now, in Algorithm 4, for each guess of $B^{19}_{y,z}$, and for each guess of $D^{19}_{y,z}$, we can find the value of C that matches all the fixed branches in time 1, using this table. Thus, we can repeat this 2^{96} times, extending the attack by 6 rounds.

- Step 1 costs $2 \times 2^{96} \times t_e$ (we solve only 2 equations most of the time, before aborting if the wanted "0" do not appear).
- The table costs $2^{64} \times t_e$, which is negligible
- Step 2 costs $2^{96} \times 5 \times t_e$, since it is the same as before, and we only need forwards computation of SP-Boxes to check if the full path is correct.

Note that we can get rid of the term t_e if we use a memory of size 2^{96} to store the solutions of the SP-Box equations. In that case, the overall time complexity is slightly below 2^{96} evaluations of Gimli, since fewer SP-Boxes are evaluated in each step than in the full primitive.

5 Better Quantum Collision Attacks

In this section, we explain how our attacks can be extended in the quantum setting, where even more rounds can be broken. We want to emphasize that, as

our goal is simply to determine a security margin, we will not go into the details of the implementation of these attacks as quantum algorithms. We will only show how to use well-known building blocks of quantum computing in order to build these new attacks, and show why they perform better than the corresponding generic quantum attacks. At this point, we assume that the reader is familiar with the basics of quantum computing that are covered in textbooks such as [30]. We define quantum algorithms in the *quantum circuit model*. The circuit starts with a set of qubits (elementary quantum systems) initialized to a basis state and applies quantum operations. The state of the system lies in a Hilbert space of dimension 2^n if there are n qubits. Quantum operations are linear operators of this space, and a quantum circuit is built from such elementary operators coined *quantum gates*. The result of a quantum computation is accessed through *measurement* of the qubits, which destroys their state.

The cryptanalytic algorithms that we consider in this section do not require any form of query to a black-box, since we want only to build a collision on the hash function. Thus, they do not require any more specific model (*e.g.* the Q2 model used in some works in quantum cryptanalysis).

5.1 Tools, Model and Complexity Estimates

Most of the collision attacks presented in this section rely on an exhaustive search. For example, consider the 8-round attack of Algorithm 3. Both steps are exhaustive searches in spaces of size 2^{32} that contain on average a single solution:

- In the first step, we find A_x^{21} such that, after solving a sequence of SP-Box equations, a 32-bit condition is met: the first equation finds $A_x'^{21}$ such that there is a collision in x after two SP-Boxes, the second equation finds A_x^{19} such that there is a collision in x after two SP-Boxes, *etc.*, and the final 32-bit condition is that $A_z'^{13}$ and A_z^{13} must collide.
- In the second step, we find the good C_x^{21} by guessing it and trying to match with a 32-bit condition.

Quantumly, Grover's algorithm [19] speeds up exhaustive search quadratically. Amplitude Amplification [12] is a powerful generalization which applies to any pair $\mathcal{A}, \chi$ such that:

- $\mathcal{A}$ is a quantum algorithm without measurements (a unitary and reversible operation), that takes no input and produces an output $x \in X$.
- $\chi : X \to \{0,1\}$ is a function that decides whether $x \in X$ is a "good" output of $\mathcal{A}$ ($\chi(x) = 1$) or a "failure" of $\mathcal{A}$, such that χ can also be implemented as a quantum algorithm.

Theorem 1 (Amplitude Amplification [12], informal). *Let $\mathcal{A}$ be a quantum algorithm without measurements that succeeds with probability p and O_χ be a quantum algorithm that tests whether an output of $\mathcal{A}$ is a failure or not. Then there exists a quantum algorithm that finds a good output of $\mathcal{A}$ using $O(\sqrt{1/p})$ calls to $\mathcal{A}$ and O_χ.*

Quantum Embeddings. Any classical algorithm admits a *quantum embedding*, that is, a quantum algorithm that returns the same results. Note that this is not a trivial fact, because a quantum algorithm without measurement is reversible.

Definition 2. *Let $\mathcal{A}$ be a randomized algorithm with no input. A* quantum embedding *for $\mathcal{A}$ is a quantum algorithm $\mathcal{A}'$ that has no input, and the distribution over the possible outcomes of $\mathcal{A}'$ (after measurement) is the same as the distribution over possible outcomes of $\mathcal{A}$.*

This quantum embedding admits similar time and space complexities, where classical elementary operations (logic gates) are replaced by quantum gates and classical bits by qubits. Generic time-space trade-offs have been studied in [4,26,23], but precise optimizations are required in practice, where the bulk of the work comes from making the computation reversible. As we just want to compare costs with quantum generic attacks, the following fact will be useful.

Remark 1. The ratio in time complexities is approximately preserved when embedding classical algorithms into quantum algorithms.

For example, if a classical algorithm has a time complexity equivalent to 1000 evaluations of Gimli, we can consider that the corresponding quantum embedding has a time complexity equivalent to 1000 quantum evaluations of Gimli. In all quantum attacks, we will give quantum time complexities relatively to quantumly evaluating Gimli. In order to use Amplitude Amplification (Theorem 1 above), we simply need to define classical randomized algorithms for $\mathcal{A}$ and O_χ.

5.2 Example

We take the example of the classical 8-round collision attack. Both steps run in classical time $2^{32}\times 4\times t_e$ by running 2^{32} iterates of a randomized algorithm of time complexity $4 \times t_e$. Using Amplitude Amplification, we obtain a corresponding quantum algorithm with time complexity *approximately* $2^{16} \times 4 \times t_{qe}$, where t_{qe} is the time to solve quantumly an SP-Box equation, relative to the cost of a quantum implementation of Gimli. As we remarked above, we can approximate $t_{qe} \simeq t_e$.

This approximation comes from different factors:

- a small constant factor $\frac{\pi}{2}$ which is inherent to quantum search.
- the trade-offs between time and space in the detailed implementations of the primitive and its components. Let us simply notice that Gimli, compared to other primitives that have been studied in this setting, *e.g.* AES [22], seems fairly easy to implement using basic quantum computing operations. In the example of AES, the most costly component is the S-Box [22], and Gimli does not have such.

We are mainly interested in the security margin, and these approximations will be sufficient for us to determine whether a given algorithm runs faster or slower than the corresponding quantum generic attack. Thus, we will write that the quantum 8-round attack on Gimli-Hash runs in time $\simeq 2^{16} \times 4 \times t_e$.

5.3 Quantum Collision Bounds and Quantum Attacks

The best quantum generic attack for finding collisions depends on the computational model, more precisely, on the cost assigned to quantum-accessible memory. Different choices are possible, which are detailed *e.g.* in [21]. In short, the overall cost of quantum collision search depends on the cost that is assigned to quantum hardware.

In this paper, we will simply consider the most conservative setting, where quantum memory is free. Note that this actually makes our attacks overall *less* efficient, since the generic algorithm is the most efficient possible (and they'll also work in the other settings). In this situation, the best collision search algorithm is by Brassard, Høyer and Tapp [13]. It will find a collision on Gimli-Hash in approximately $2^{256/3} \simeq 2^{85.3}$ quantum evaluations of Gimli, using a quantum-accessible memory of size $2^{85.3}$.

Quantum collision attacks reaching more rounds than classical ones. In [21], Hosoyamada and Sasaki initiated the study of *dedicated quantum* attacks on hash functions. They remarked that quantum collision search does not benefit from a square-root speedup (it goes from roughly $2^{n/2}$ to $2^{n/3}$ with the BHT algorithm, and the gain is even smaller in more constrained models of quantum hardware), while some collision-finding procedures may have a better speedup, say, quadratic. Thus:

- there may exist quantum collision attacks such that the corresponding classical algorithm is *not* an attack (it gets worse than the generic bound);
- the *quantum* security margin of hash functions for collision attacks is likely to be smaller than the *classical* one.

Hosoyamada and Sasaki studied differential trails in the hash functions AES-MMO and Whirlpool. Although our attacks are based on a different framework, we show that similar findings apply for Gimli.

5.4 Quantum Collision Attacks on Gimli

We assume that $t_e < 2^{20}$, hence solving an equation costs less than evaluating reduced-round Gimli 2^{20} times, which is suggested by our computations, and should hold in the quantum setting as well.

Full-state collisions. By adding another 32-bit condition in the classical 12-round collision attack, we obtain a procedure which runs classically in time $4 \times 2^{128} \times t_e$, which is too high. However, using Amplitude Amplification, we obtain a procedure that runs in quantum time $\simeq 4 \times 2^{64} \times t_e$ and reaches 14 rounds, with less complexity than the quantum collision bound.

Semi-free start collisions. We can extend the 18-round semi-free start collision attack in the same way. Building the table will still cost a time 2^{64}. This table must be stored in a classical memory with quantum random access. The first step goes from $2 \times 2^{96} \times t_e$ classically to approximately $2 \times 2^{48} \times t_e$ quantumly. The second step does as well. Thus, adding a 32-bit condition enables us to attack 20 rounds in quantum time $2^{64} \times 4 \times t_e$.

6 Statistical Analyses of Gimli

6.1 Linear cryptanalysis

This section aims to provide the first analysis of the linear properties of the Gimli permutation and its components. We use a Mixed Integer Linear Programming (MILP) modelization of the operations constructed according to [1], and then solve it with the SCIP software [17,18] to search for linear trails with optimal correlation.

Linear trails of the (double) SP-box We begin by studying the linear trails of the SP-Box. Since the Gimli permutation mainly uses the composition of the SP-Box with itself, we focus on the "double" SP-Box SP^2.

Let us consider that we apply the double SP-box to $A = (x, y, z)$ to obtain $A'' = (x'', y'', z'') = SP^2(x, y, z)$. We are interested in correlated linear approximations, that is, masks $\alpha = (\alpha_x, \alpha_y, \alpha_z)$ and $\beta = (\beta_x, \beta_y, \beta_z)$ for which

$$c(SP^2, \alpha, \beta) = 2^{-96}\Big(\big|\{A : \alpha \cdot A \oplus \beta \cdot A'' = 0\}\big| - \big|\{A : \alpha \cdot A \oplus \beta \cdot A'' = 1\}\big|\Big)$$

is as large (in absolute value) as possible. From Section 2.1 we already know that the relationship $x_8 + x''_0 + y''_0 + z''_0 = 0$ always holds. This is a linear trail of the double SP-box with correlation 1, and it is unique.

An automated MILP-based search for linear trails of correlation 2^{-1} and 2^{-2} shows that there exist at least 41 trails of the former kind and 572 of the latter, but this is not an exhaustive count. Although these approximations probably only account for a very small fraction of the possible ones, a more thorough study of the distribution of the different correlation values among all the trails would be of interest.

We have found no signs of significant linear-hull effects within the double SP-box, although since we have not considered every linear trail, they might still exist.

Some linear trails of round-reduced Gimli. In order to provide some linear trails for reduced-round Gimli, we first focus on trails with only one active SP-Box in each round, or more specifically, with masks which only cover one column in each round. They do not provide an upper bound on the correlation of more general trails, but we still think they could be of interest, and this restriction greatly limits the search space.

More specifically, we consider linear trails on powers of the SP-box such that the mask for the x word is zero every two rounds. This means that the mask is unaffected by the big and small swaps, and these trails easily translate into trails for the reduced-round Gimli construction with the same correlation.

We first look at iterative linear trails for the double SP-box so that both the input and output masks have the x word set to zero. We find that the optimal correlation is 2^{-26}, and this is the (maybe not unique) associated trail:

$$\Gamma_1 : \begin{matrix}\texttt{00000000}\\ \texttt{0a064e03}\\ \texttt{0c08e406}\end{matrix} \xrightarrow[2^{-14}]{} \begin{matrix}\texttt{0c8b0507}\\ \texttt{01040322}\\ \texttt{00054302}\end{matrix} \xrightarrow[2^{-12}]{} \begin{matrix}\texttt{00000000}\\ \texttt{0a064e03}\\ \texttt{0c08e406}\end{matrix}.$$

Since this trail is iterative, we can construct $2l$-round trails with correlation 2^{-26l}. Next, we provide a similar iterative trail for four rounds with correlation 2^{-47}, though other trails with larger correlation might exist with the same restrictions:

$$\Gamma_2: \begin{matrix} \texttt{00000000} \\ \texttt{02060000} \\ \texttt{00020541} \end{matrix} \xrightarrow[2^{-19}]{} \begin{matrix} \texttt{06422511} \\ \texttt{088a8131} \\ \texttt{08828111} \end{matrix} \xrightarrow[2^{-11}]{} \begin{matrix} \texttt{00000000} \\ \texttt{15024215} \\ \texttt{0405003a} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{04054102} \\ \texttt{00010280} \\ \texttt{000182a0} \end{matrix} \xrightarrow[2^{-7}]{} \begin{matrix} \texttt{00000000} \\ \texttt{02060000} \\ \texttt{00020541} \end{matrix}.$$

With this, we can construct trails of $4l$ rounds with correlation 2^{-47l}. At this point the search for iterative trails becomes computationally expensive so we search for non-iterative trails. We find an optimal four-round trail with correlation 2^{-16}:

$$\Gamma_3: \begin{matrix} \texttt{00000000} \\ \texttt{90002000} \\ \texttt{00400110} \end{matrix} \xrightarrow[2^{-3}]{} \begin{matrix} \texttt{00400100} \\ \texttt{00000020} \\ \texttt{00000000} \end{matrix} \xrightarrow[2^{-1}]{} \begin{matrix} \texttt{00000000} \\ \texttt{00004000} \\ \texttt{00000001} \end{matrix} \xrightarrow[2^{-2}]{} \begin{matrix} \texttt{00000001} \\ \texttt{00800001} \\ \texttt{00800001} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{00000000} \\ \texttt{000002aa} \\ \texttt{010002aa} \end{matrix}.$$

Next, we attempt to extend this trail at the end. We find the following four-round trail with correlation 2^{-48} which has the output mask of the previous one as its input mask:

$$\Gamma_4: \begin{matrix} \texttt{00000000} \\ \texttt{000002aa} \\ \texttt{010002aa} \end{matrix} \xrightarrow[2^{-18}]{} \begin{matrix} \texttt{01448312} \\ \texttt{01094200} \\ \texttt{0101f260} \end{matrix} \xrightarrow[2^{-11}]{} \begin{matrix} \texttt{00000000} \\ \texttt{18040003} \\ \texttt{0a054480} \end{matrix} \xrightarrow[2^{-12}]{} \begin{matrix} \texttt{0a040580} \\ \texttt{02450200} \\ \texttt{02050200} \end{matrix} \xrightarrow[2^{-7}]{} \begin{matrix} \texttt{00000000} \\ \texttt{88040004} \\ \texttt{080c0401} \end{matrix}.$$

Combining both trails, we obtain an eight-round trail of correlation 2^{-64}. There are no approximations for the double SP-box for which the output mask is the input mask of Γ_3 and so that the input mask has the x word set to zero. However, by removing the last condition we can add two rounds with a 2^{-16} correlation:

$$\Gamma_5: \begin{matrix} \texttt{68009800} \\ \texttt{40202088} \\ \texttt{403510d4} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{40211090} \\ \texttt{00480010} \\ \texttt{00200088} \end{matrix} \xrightarrow[2^{-6}]{} \begin{matrix} \texttt{00000000} \\ \texttt{90002000} \\ \texttt{00400110} \end{matrix}.$$

In the same way, we can add two additional rounds at the end of Γ_4 with correlation 2^{-19}:

$$\Gamma_6: \begin{matrix} \texttt{88040004} \\ \texttt{40202088} \\ \texttt{080c0401} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{080a0281} \\ \texttt{000c0901} \\ \texttt{000c0901} \end{matrix} \xrightarrow[2^{-9}]{} \begin{matrix} \texttt{48000800} \\ \texttt{70100a00} \\ \texttt{e0180002} \end{matrix}.$$

By combining these four trails, we obtain a twelve-round linear trail for Gimli with correlation 2^{-99}. Then, by combining several trails in a similar manner we obtain the following 14-round trail with correlation 2^{-128}:

$$\begin{aligned} \Gamma_7: & \begin{matrix} \texttt{00408000} \\ \texttt{20e04060} \\ \texttt{e0c1c000} \end{matrix} \xrightarrow[2^{-19}]{} \begin{matrix} \texttt{e0e9e078} \\ \texttt{206c202e} \\ \texttt{206060a0} \end{matrix} \xrightarrow[2^{-12}]{} \begin{matrix} \texttt{00000000} \\ \texttt{f8606840} \\ \texttt{80808180} \end{matrix} \xrightarrow[2^{-7}]{} \begin{matrix} \texttt{80808180} \\ \texttt{40400060} \\ \texttt{40400040} \end{matrix} \xrightarrow[2^{-4}]{} \begin{matrix} \texttt{00000000} \\ \texttt{80008080} \\ \texttt{00800001} \end{matrix} \xrightarrow[2^{-6}]{} \\ & \begin{matrix} \texttt{00000001} \\ \texttt{01010101} \\ \texttt{01010101} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{00000000} \\ \texttt{02020202} \\ \texttt{03020202} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{00000000} \\ \texttt{04040404} \\ \texttt{04040404} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{00000000} \\ \texttt{08080808} \\ \texttt{08080808} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{00000000} \\ \texttt{10101010} \\ \texttt{10101010} \end{matrix} \xrightarrow[2^{-8}]{} \\ & \begin{matrix} \texttt{00000000} \\ \texttt{20202020} \\ \texttt{20202020} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{00000000} \\ \texttt{40404040} \\ \texttt{40404040} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{00000000} \\ \texttt{80c08080} \\ \texttt{80e08080} \end{matrix} \xrightarrow[2^{-10}]{} \begin{matrix} \texttt{80110000} \\ \texttt{01800101} \\ \texttt{01c20101} \end{matrix} \xrightarrow[2^{-8}]{} \begin{matrix} \texttt{01020800} \\ \texttt{01000802} \\ \texttt{01801c02} \end{matrix} \end{aligned}.$$

Table 5. Linear trails for reduced-round Gimli. Some of them apply to shifted versions of the algorithm starting with two consective SP-box substitutions instead of one.

# Rounds	Correlation	Construction	Shift
1	1	Probability 1 trail from 2.1	No
2	1	Probability 1 trail from 2.1	Yes
3	2^{-6}	First three rounds of Γ_3	Yes
4	2^{-12}	Last round of Γ_5, first three rounds of Γ_3	No
5	2^{-22}	Last round of Γ_5, Γ_3	No
6	2^{-32}	Γ_5, Γ_3	Yes
7	2^{-50}	Γ_5, Γ_3, first round of Γ_4	Yes
8	2^{-61}	Γ_5, Γ_3, first two rounds of Γ_4	Yes
9	2^{-70}	Last round of Γ_5, Γ_3, Γ_4	No
10	2^{-80}	Last round of Γ_5, Γ_3, Γ_4, first round of Γ_6	No
11	2^{-89}	Last round of Γ_5, Γ_3, Γ_4, Γ_6	No
12	2^{-99}	Γ_5, Γ_3, Γ_4, Γ_6	Yes
13	2^{-109}	Last thirteen rounds of Γ_7	No
14	2^{-128}	Γ_7	Yes
15	2^{-137}	Last rounds of Γ_8 and Γ_9,Γ_7	No
16	2^{-150}	Γ_8,Γ_9,Γ_7	Yes

Finally, this trail can be extended at the top by adding the following two-round trails (they now have two active SP-Boxes in each round because of a swap):

$$\Gamma_8: \begin{matrix} \texttt{48f00060} \\ \texttt{6818cc18} \\ \texttt{21a404c8} \end{matrix} \xrightarrow[2^{-9}]{} \begin{matrix} \texttt{018060c0} \\ \texttt{20085810} \\ \texttt{40408000} \end{matrix} \xrightarrow[2^{-11}]{} \begin{matrix} \texttt{00000000} \\ \texttt{20e04060} \\ \texttt{e0c1c000} \end{matrix}$$

$$\Gamma_9: \begin{matrix} \texttt{40a04000} \\ \texttt{20000008} \\ \texttt{00204000} \end{matrix} \xrightarrow[2^{-4}]{} \begin{matrix} \texttt{00000000} \\ \texttt{00002040} \\ \texttt{00408000} \end{matrix} \xrightarrow[2^{-2}]{} \begin{matrix} \texttt{00408000} \\ \texttt{00000000} \\ \texttt{00000000} \end{matrix}.$$

Using these, we obtain a 16-round trail with correlation 2^{-150}. In general, by combining these trails in different ways, we provide the linear trails for up to 16 round of Gimli shown in Table 5.

These are just some linear trails of Gimli which belong to a very specific subfamily, and for more than four rounds we have not proven optimality even within that family, so it is quite possible that better linear trails exist. We have also searched for any significant linear trails which share the same input and output masks to see if there is a noticeable linear hull effect for these approximations, but we have found no additional trails of large correlation.

All these trails can be used to mount distinguishing attacks on the Gimli permutation with a data complexity proportional to the inverse of the square of the correlation, which also works for the block cipher built with the Even-Mansour construction from the Gimli permutation. It is possible to reduce the complexity slightly by using multiple linear cryptanalysis. By considering the same trail but in the four columns we can increase the capacity by a factor of

four. By shifting the iterative trail by two rounds we can obtain an additional factor two in the 16-round attack.

6.2 Differential-Linear Cryptanalysis

We now consider differential-linear cryptanalysis, a technique that combines a differential trail and a linear trail built independently.

We use the approach of Leurent [25] where we actually split the cipher in three parts $E = E_\perp \circ E_\mathtt{I} \circ E_\top$, with a differential trail in $E_\top$, a linear trail in $E_\perp$, and an experimental evaluation of the bias in $E_\mathtt{I}$. This gives a more accurate evaluation of the complexity. More precisely, we consider

- a differential trail $\delta_{\text{in}} \to \delta_{\text{out}}$ for $E_\top$ with probability $p = \Pr_X\left(E_\top(X) \oplus E_\top(X \oplus \delta_{\text{in}}) = \delta_{\text{out}}\right)$.
- an experimental bias b from δ_{out} to β for $E_\mathtt{I}$:

$$\begin{aligned} b &= c(\alpha \cdot E_\mathtt{I}(W), \alpha \cdot E_\mathtt{I}(W \oplus \delta_{\text{out}})) \\ &= 2 \Pr_W(\alpha \cdot E_\mathtt{I}(W) = \alpha \cdot E_\mathtt{I}(W \oplus \delta_{\text{out}})) - 1 \end{aligned}$$

- a linear trail $\alpha \to \beta$ for $E_\perp$ with correlation $c = 2\Pr_Y(\alpha \cdot Y = \beta \cdot E_\perp(Y)) - 1$.

If the three parts are independent then we can estimate the bias of the differential-linear distinguisher as:

$$c(\beta \cdot E(X), \beta \cdot E(X \oplus \delta_{\text{in}})) = 2 \Pr_X(\beta \cdot E(X) = \beta \cdot E(X \oplus \delta_{\text{in}})) - 1 \approx pbc^2$$

Therefore, the complexity of the distinguisher is about $2/p^2b^2c^4$.

In Gimli, there are no keys, so the assumption of independence does not hold, but experiments show that the computed bias is close to the reality. In practice, the best results are obtained when δ_{out} and α have a low hamming weight [25].

Differential Trail. We start by picking a trail that mainly follows the one given by the designers [5] with slight changes to optimize it for for our number of rounds. We chose a trail with a difference pattern δ_{out} with two active bits. A differential trail over 5 rounds with probability $p = 2^{-28}$ is given in Table 6. We considered trade-offs between the different phases, and it never seems to be worth it to propagate the trail any further.

Experimental Bias. Starting from the target difference pattern δ_{out} at round 19, we experimentally evaluate the bias after a few rounds with all possible masks α with a single active bit. Concretely, we choose the state at random, build the second state by adding δ_{out} and observe the bias a few rounds later.

The most useful results are on the least significant bit z_0 of the last word, where the probability of having a difference is smaller than $1/2$. After computing 8 round, the probability of having an active difference on this bit in round 12 is $\frac{1}{2} - 2^{-6.2}$, a correlation of $b = -2^{-5.2}$. After 9 rounds, at the end of round 11, there is a correlation of $b = -2^{-16.9}$. These probabilities are large enough to be experimentally significant after the 2^{40} trials we have made.

Table 6. A 5-round differential trail.

δ_{in}	40418080	02010000	00000000	00000000
	40400010	00000000	00000000	00000000
	80002080	80010080	00000000	00000000
Round 24	80010080	00000000	00000000	00000000
	00402000	00000000	00000000	00000000
$p = 2^{-18}$	80400080	00000000	00000000	00000000
Round 23	00000080	00000000	00000000	00000000
	00400000	00000000	00000000	00000000
$p = 2^{-8}$	80000000	00000000	00000000	00000000
Round 22	00000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000000
$p = 1$	80000000	00000000	00000000	00000000
Round 21	80000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000000
$p = 1$	00000000	00000000	00000000	00000000
δ_{out}	00000000	00000000	00000000	00000000
Round 20	00800000	00000000	00000000	00000000
$p = 2^{-2}$	00800000	00000000	00000000	00000000

Linear Trail. We use assisted tools to find good linear trails, starting from the mask corresponding to z_0. The diffusion is not the same depending whether we start after round 12 or 11 so we show the best 3 rounds linear approximation for both case. We find a correlation c of 2^{-17} and 2^{-16} respectively, see Table 7.

Table 7. Diffusion of z_0 starting at the end of round 12.

Round 12	00000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000001
Round 11	00000000	00000000	00000000	00000001
	00000000	00000000	00000000	00000001
corr $= 2^{-0}$	00000000	00000000	00000000	00000001
Round 10	00000000	00800001	00000000	00000000
	00000000	00000000	00000000	00800201
corr $= 2^{-5}$	00000000	00000000	00000000	01c00201
Round 9	00000000	00880000	00000000	01000201
	00000000	00f10000	00000000	01040000
corr $= 2^{-12}$	00000000	01e00000	00000000	01840000

Complexity of the distinguishers. We can combine the trails in different way to obtain distinguishers on 15, 16 or 17 rounds (starting from round 24).

15 rounds We use 5 rounds for $E_{\top}$, 8 rounds for $E_{\mathbb{T}}$, 2 rounds for $E_{\perp}$. The corresponding complexity is $2/pbc^2 = 2 \times 2^{2\times 28} \times 2^{2\times 5.2} \times 2^{4\times 5} = 2^{87.4}$.

Table 8. Diffusion of z_0 starting at the end of round 11.

Round 11	00000000	00000000	00000000	00000000
	00000000	00000000	00000000	00000000
	00000001	00000000	00000000	00000000
Round 10	00000000	00000000	00000001	00000000
corr $= 2^{-0}$	00000001	00000000	00000000	00000000
	00000001	00000000	00000000	00000000
Round 9	00000001	00000000	00800000	00000000
corr $= 2^{-5}$	00000201	00000000	00800000	00000000
	00000201	00000000	01c00000	00000000
Round 8	00000000	00200201	00000000	01004000
corr $= 2^{-11}$	00000001	00000000	01004001	00000000
	01000001	00000000	0180e001	00000000

16 rounds We use 5 rounds for $E_\top$, 9 rounds for $E_\pitchfork$, 2 rounds for $E_\perp$. The corresponding complexity is $2/pbc^2 = 2 \times 2^{2\times 28} \times 2^{2\times 16.9} \times 2^{4\times 5} = 2^{110.8}$.

17 rounds We use 5 rounds for $E_\top$, 9 rounds for $E_\pitchfork$, 3 rounds for $E_\perp$. The corresponding complexity is $2/pbc^2 = 2 \times 2^{2\times 28} \times 2^{2\times 16.9} \times 2^{4\times 16} = 2^{154.8}$.

Those distinguishers can be used when the Gimli permutation is used to build a block cipher with the Even-Mansour construction. Such a cipher should ensure a birthday bound security of up to 2^{192} query, which is less efficient than our differential-linear distinguisher if the number of rounds Gimli is reduced to 17 (or fewer). Further improvement should be possible with the partitioning technique of [25], but we leave this to future work.

7 Conclusion

A common point of the results presented in this paper is that they exploit the relatively slow diffusion between the columns of the Gimli state. This issue has trivial causes: swaps are effectively the identity for 256 out of the 384 bits of the internal state, and occur only every second round. Thus, the Gimli SP-Box is always applied twice, except at the first and last rounds. This means that the permutation can be viewed as an SPN with only 12 rounds, and with very simple linear layers. Meanwhile, the double SP-Box is a rather simple function, and some of our attacks rely crucially on solving efficiently equations that relate its inputs and outputs.

Though our results do not pose a direct threat to the Gimli NIST candidate, low-complexity full-round distinguishers on the permutation or reduced-round attacks for a high proportion of the rounds (specially when not predicted by the designers) have been considered in some cases as an issue worth countering by proposing a tweak, as can be seen for instance in the modification [3] recently proposed by the Spook team [2] to protect against the cryptanalysis results from [15].

In addition, Gimli designers studied other linear layers instead of the swaps, like using an MDS or the linear transformation from SPARX [16], and they found some advantages in proving security against various types of attacks. On the other hand, they also found it unclear whether these advantages would outweight the costs. We believe our results show some light in this direction: the other variants that were considered seem a priori to be stronger regarding our analysis, though an extensive study should be performed.

We believe the distinguishers might still be improved by exploiting the properties of the SP-Box, which we have not done yet.

In order to mitigate the attacks based on internal symmetries and guess-and-determine methods (including our distinguishers on the permutation) a simple fix would be to perform a swap at each round instead of every second round. This would however imply a renewed cryptanalysis effort.

Acknowledgments. The authors would like to thank all the members of the *cryptanalysis party* meetings, for many useful comments and discussions, in particular many thanks to Anne Canteaut, Virginie Lallemand and Thomas Fuhr for many interesting discussions over previous versions of this work. Thanks to Donghoon Chang for finding some mistakes and inaccuracies, including an error in a 32-round version of our distinguisher. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 714294 - acronym QUASYModo).

References

1. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Trans. Symm. Cryptol. 2017(4), 99–129 (2017)
2. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.X., Wiemer, F.: Spook: Sponge-based leakage-resilient authenticated encryption with a masked tweakable block cipher. Submission to the NIST Lightweight Cryptography project. Available online https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Spook-spec-round2.pdf. (2019)
3. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.X., Udvarhelyi, B., Wiemer, F.: Spook: Sponge-based leakage-resilient authenticated encryption with a masked tweakable block cipher. IACR Trans. Symm. Cryptol. 2020(Special Issue on NIST Lightweight) (2020), to appear.
4. Bennett, C.H.: Time/space trade-offs for reversible computation. SIAM J. Comput. 18(4), 766–776 (1989)
5. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli : A cross-platform permutation. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 299–320. Springer, Heidelberg (Sep 2017)

6. Bernstein, D.J., Kölbl, S., Lucks, S., Massolino, P.M.C., Mendel, F., Nawaz, K., Schneider, T., Schwabe, P., Standaert, F.X., Todo, Y., Viguier, B.: Gimli. Submission to the NIST Lightweight Cryptography project. Available online https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/gimli-spec.pdf. (2019)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge functions. In: ECRYPT hash workshop (2007)
8. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (Apr 2008)
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge-based pseudo-random number generators. In: Mangard, S., Standaert, F.X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 33–47. Springer, Heidelberg (Aug 2010)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer, Heidelberg (Aug 2012)
11. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (Jan 1991)
12. Brassard, G., Hoyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. Contemporary Mathematics 305, 53–74 (2002)
13. Brassard, G., Høyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 163–169. Springer, Heidelberg (Apr 1998)
14. Cai, J., Wei, Z., Zhang, Y., Sun, S., Hu, L.: Zero-sum distinguishers for round-reduced GIMLI permutation. In: Mori, P., Furnell, S., Camp, O. (eds.) Proceedings of the 5th International Conference on Information Systems Security and Privacy, ICISSP 2019, Prague, Czech Republic, February 23-25, 2019. pp. 38–43. SciTePress (2019)
15. Derbez, P., Huynh, P., Lallemand, V., Naya-Plasencia, M., Perrin, L., Schrottenloher, A.: Cryptanalysis results on spook. Cryptology ePrint Archive, Report 2020/309 (2020), https://eprint.iacr.org/2020/309
16. Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: Sparx and LAX. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 484–513. Springer, Heidelberg (Dec 2016)
17. Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. Technical report, Optimization Online (July 2018), http://www.optimization-online.org/DB_HTML/2018/07/6692.html
18. Gleixner, A., Bastubbe, M., Eifler, L., Gally, T., Gamrath, G., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Lübbecke, M.E., Maher, S.J., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Schubert, C., Serrano, F., Shinano, Y., Viernickel, J.M., Walter, M., Wegscheider, F., Witt, J.T., Witzig, J.: The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin (July 2018), http://nbn-resolving.de/urn:nbn:de:0297-zib-69361
19. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC. pp. 212–219. ACM Press (May 1996)

20. Hamburg, M.: Cryptanalysis of 22 1/2 rounds of gimli. Cryptology ePrint Archive, Report 2017/743 (2017), http://eprint.iacr.org/2017/743
21. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 249–279. Springer, Heidelberg (May 2020)
22. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and LowMC. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 280–310. Springer, Heidelberg (May 2020)
23. Knill, E.: An analysis of bennett's pebble game. CoRR abs/math/9508218 (1995)
24. Lamberger, M., Mendel, F., Schläffer, M., Rechberger, C., Rijmen, V.: The rebound attack and subspace distinguishers: Application to Whirlpool. Journal of Cryptology 28(2), 257–296 (Apr 2015)
25. Leurent, G.: Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 344–371. Springer, Heidelberg (May 2016)
26. Levin, R.Y., Sherman, A.T.: A note on bennett's time-space tradeoff for reversible computation. SIAM J. Comput. 19(4), 673–677 (1990)
27. Liu, F., Isobe, T., Meier, W.: Preimages and collisions for up to 5-round gimli-hash using divide-and-conquer methods. Cryptology ePrint Archive, Report 2019/1080 (2019), https://eprint.iacr.org/2019/1080
28. Liu, F., Isobe, T., Meier, W.: Automatic verification of differential characteristics: Application to reduced gimli. CRYPTO 2020, to appear (2020), https://eprint.iacr.org/2020/591
29. Liu, F., Isobe, T., Meier, W.: Exploiting weak diffusion of gimli: A full-round distinguisher and reduced-round preimage attacks. Cryptology ePrint Archive, Report 2020/561 (2020), https://eprint.iacr.org/2020/561
30. Nielsen, M.A., Chuang, I.L.: Quantum information and quantum computation. Cambridge: Cambridge University Press 2(8), 23 (2000)
31. Soos, M., Nohl, K., Castelluccia, C.: Extending SAT solvers to cryptographic problems. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5584, pp. 244–257. Springer (2009)
32. Zong, R., Dong, X., Wang, X.: Collision attacks on round-reduced Gimli-Hash/Ascon-Xof/Ascon-Hash. Cryptology ePrint Archive, Report 2019/1115 (2019), https://eprint.iacr.org/2019/1115

A Appendix

A.1 SP-Box Inverse

The SP-Box is a bijective operation, but its inverse is difficult to write (and it is never used).

1. Swap x and z
2. Perform:[3]

$$
\begin{aligned}
x_0 &\leftarrow x'_0 \\
y_0 &\leftarrow y'_0 + x'_0 \\
z_0 &\leftarrow z'_0 + x'_0 + y'_0 \\
x_1 &\leftarrow x'_1 + z_0 \\
y_1 &\leftarrow y'_1 + x'_1 + z_0 + (x_0 \vee z_0) \\
z_1 &\leftarrow z'_1 + y'_1 + x'_1 + z_0 + (x_0 \vee z_0) \\
x_2 &\leftarrow x'_2 + z_1 + (y_0 \wedge z_0) \\
y_2 &\leftarrow y'_2 + x'_2 + z_1 + (y_0 \wedge z_0) + (x_1 \vee z_1) \\
z_2 &\leftarrow z'_2 + y'_2 + x'_2 + z_1 + (y_0 \wedge z_0) + (x_1 \vee z_1) \\
\forall 3 \leq i \leq 32, x_i &\leftarrow x'_i + z_{i-1} + (y_{i-2} \wedge z_{i-2}) \\
y_i &\leftarrow y'_i + x_i + (x_{i-1} \vee z_{i-1}) \\
z_i &\leftarrow z'_i + y_i + (x_{i-3} \wedge y_{i-3})
\end{aligned}
$$

3. Rotate x and y: $x_i = x_{i+24 \mod 32}$ and $y_i = y_{i+9 \mod 32}$

A.2 Gimli-Hash

Algorithm 5 Gimli-Hash (from [6])

Input: $M \in \{0,1\}^*$
Output: $h \in \{0,1\}^{256}$

$S \leftarrow 0$ ▷ Initialize state to 0
$m_1, \ldots, m_t \leftarrow \text{pad}(M)$
for i from 1 to t **do**
  if $i = t$ **then**
    $D_z \leftarrow D_z \oplus$ `0x01000000`
  end if
  $S \leftarrow \text{absorb}(S, m_i)$ ▷ XOR m_i in the rate: A_x, B_x, C_x, D_x, apply Gimli
end for
$h \leftarrow A_x \| B_x \| C_x \| D_x$
$S \leftarrow \text{Gimli}(S)$
$h \leftarrow h \| A_x \| B_x \| C_x \| D_x$
Return h

[3] Note that the formulas given page 15 of the specification of Gimli are erroneous. In the line $z'_n \leftarrow z_n + y_n + (x_{n-3} \wedge z_{n-3})$, z_{n-3} should be replaced by y_{n-3} and $x_j \wedge z_j$ must be replaced by $x_j \wedge y_j$ in the subsequent formulas.

A.3 Representation of Full Gimli

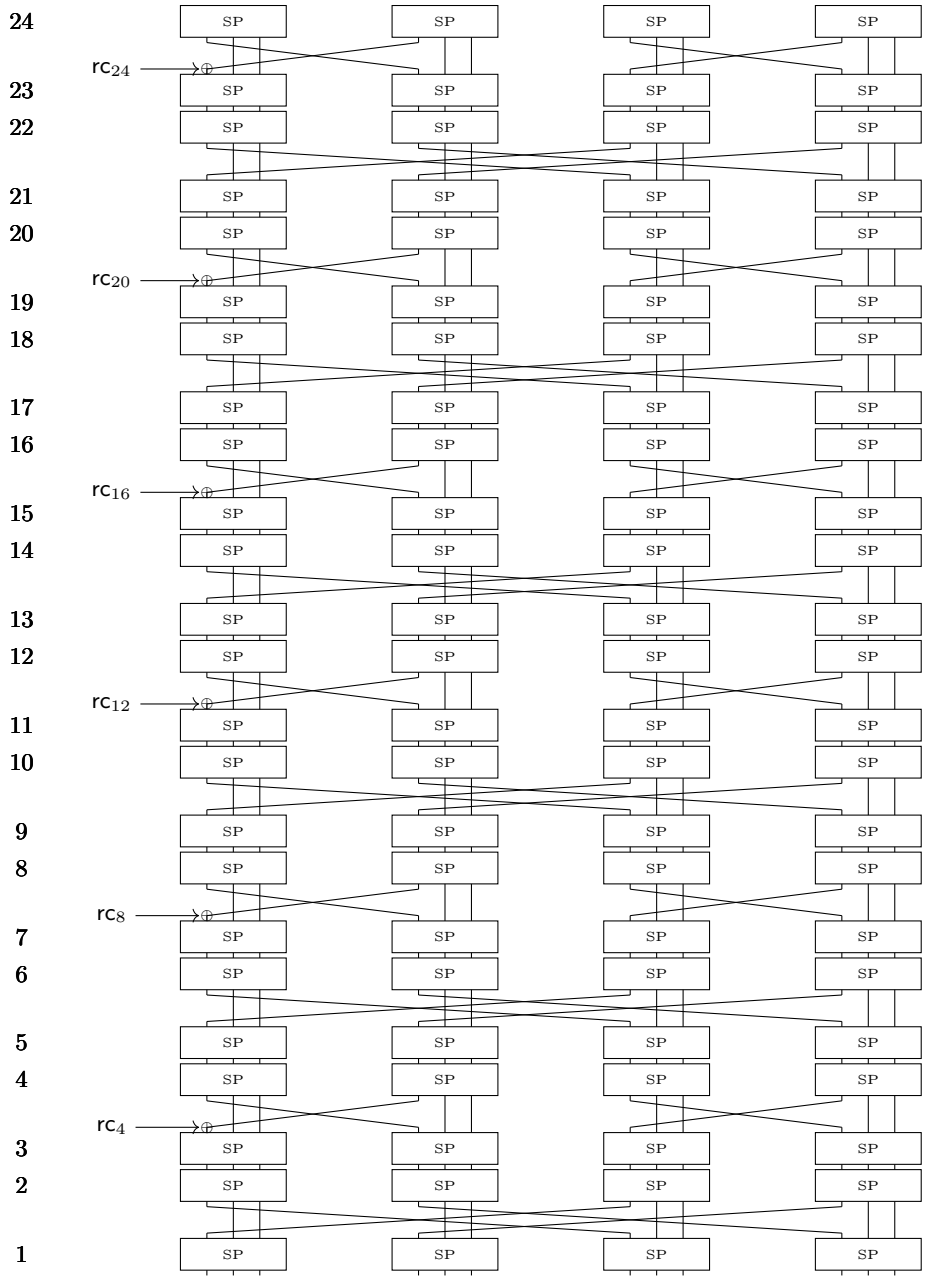

Fig. 6. A representation of full Gimli.