AlgSAT — a SAT Method for Search and Verification of Differential Characteristics from Algebraic Perspective

Huina Li[1,2], Haochen Zhang[1], Guozhen Liu[2], Kai Hu[2], Jian Guo[2] and Weidong Qiu[1]

[1] School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[2] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore

Abstract. A good differential is a start for a successful differential attack. However, a differential might be invalid, *i.e.*, there is no right pair following the differential, due to some contradictions in the conditions imposed by the differential. This paper presents a novel and handy method for searching and verifying differential trails from an algebraic perspective. From this algebraic perspective, exact Boolean expressions of differentials over a cryptographic primitive can be conveniently established, which allows for the convenient verification of a given differential trail. This verification process can be naturally formulated as a Boolean satisfiability problem (SAT).
To demonstrate the power of our new tool, we apply it to Gimli, Ascon, and Xoodoo. For Gimli, we improve the efficiency of searching for a valid 8-round colliding differential trail compared to the previous MILP model (CRYPTO 2020). Based on this differential trail, a practical semi-free-start collision attack on the intermediate 8-round Gimli-Hash is thus successfully mounted. For Ascon, we check several differential trails reported at FSE 2021. Specifically, we find that a 4-round differential used in the forgery attack on Ascon-128's iteration phase has been proven invalid. As a consequence, the corresponding forgery attack is also invalid. For Xoodoo, as an independent interest, we develop a SAT-based automatic search toolkit called XoodooSat to search for 3- and 4-round differential trail cores of Xoodoo. Our toolkit finds two more 3-round differential trail cores of weight 48 that were missed by the designers which enhances the security analysis of Xoodoo. Then, we verify tens of thousands of 3-round differential trails and two 4-round differential trails extended from the so-called differential trail cores. We find that all these differential trails are valid, which effectively demonstrates that there are no contradictions in the conditions imposed by the round differentials of the DTs in the trail core.

Keywords: Cryptographic Permutation · SAT · Automatic Verification · Differential Characteristic Search · Semi-free-start Collision Attacks

1 Introduction

With the rapid development of the Internet of Things (IoT), more and more light mobile devices appear in people's daily life, such as smart cards, wireless sensors, and Radio Frequency IDentification (RFID) tags. As these devices have limited memory and computing resources, it is infeasible to directly apply traditional encryption algorithms such as the Advanced Encryption Standard (AES) [DR02] in such scenarios. Therefore, lightweight cryptographic algorithms (LWC) have attracted more and more attention. Especially,

in August 2018, the National Institute of Standards and Technology (NIST) launched a competition [NIS18] to solicit, evaluate, and standardize LWC algorithms, including authenticated encryption with associated data (AEAD) and lightweight hash functions that are suitable for use in constrained environments. Three candidates, Ascon [DEMS21], Xoodyak [DHP+20], and Gimli [BKL+19], have attracted great attention because of their clean designs and high efficiency. On February 7$^{\text{th}}$, 2023, Ascon was announced as the ultimate winner.

Ascon, Xoodyak, and Gimli are all designed based on cryptographic permutations, which is an increasingly popular paradigm for designing LWCs. These permutation-based ciphers usually have high performance in both software and hardware implementations, but simultaneously their novel designs make it difficult for cryptanalysts to fully understand their security properties. As a result, we always would tend to study the security properties of the underlying permutations to deepen our understanding of the whole ciphers.

Similar to the classical block ciphers, these permutations are also iterated algorithms consisting of simple round functions. Naturally, some cryptanalytic methods originated for block ciphers have been borrowed to evaluate the security of permutations. One of the most important attacks among them is the differential cryptanalysis introduced by Biham and Shamir at CRYPTO 1990 [BS91]. In a differential attack, the attacker seeks a fixed input difference α_0 that propagates through a r-round primitive (the primitive could be a block cipher or a permutation) to a fixed output difference α_r with a high probability p, the differential is thus represented by (α_0, α_r). To find a proper differential (α_0, α_r) with a high probability for the primitive, we examine the differential property of the i-th $(1 \leq i \leq r)$ round and try to find a local differential for this round denoted by (a_{i-1}, a_i) whose probability is denoted by p_i. With the tacit assumption that differentials of two consecutive rounds are independent, these local differentials for all rounds could be chained into one so-called differential trail (DT) $(\alpha_0, \alpha_1, \ldots, \alpha_r)$, whose probability is computed by $p = \prod_{i=1}^{r} p_i$. For the sake of simplicity, we generally refer to all these underlying assumptions as the Markov assumption in this paper.

With the Markov assumption, many methods such as the so-called automated tools have been invented to search for useful or even optimal DTs. When using automated tools, the differential propagation rules for components of a primitive in each round are modeled by some specific constraints. All solutions satisfying these constraints are expected to be valid DTs. Based on these constraints, additional constraints, such as describing whether the corresponding Sboxes in the DTs are active or not, would also be added to the constraint pool. The set formed by all these constraints is denoted by $\mathcal{C}$ in this paper. In general, a constraint representing the number of active Sboxes which is the so-called *objective function* denoted by $\mathcal{O}$ is also imposed. Different automated tools handle $(\mathcal{C}, \mathcal{O})$ differently. There are three types of automated tools in the literature that are often used for the search: (a) Boolean satisfiability problem (SAT) [MP13], where the constraints in $\mathcal{C}$ and the objective function are modeled by the corresponding clausal normal forms (CNF). An extension of the SAT called satisfiability modulo theories (SMT) [GD07] is also available, which generalizes the SAT to more complex formulas involving *e.g.*, the integers and / or bit vectors. (b) Mixed Integer Linear Programming (MILP) [MWGP11] where $(\mathcal{C}, \mathcal{O})$ are described by a set of inequalities (including equations). (c) Constraint programming (CP) [GMS16], where users could use more flexible formulas to describe $(\mathcal{C}, \mathcal{O})$.

Although the Markov assumption is generally considered reasonable for block ciphers, unfortunately, sometimes it would not hold for some permutations. At CRYPTO 2020 [LIM20], Liu, Isobe, and Meier pointed out that the 6-round and 2-round DT used for attacking Gimli-Hash and Ascon-Hash, respectively, found by MILP in [ZDW19] is invalid. That means, although these DTs seem legal under the Markov assumption, no conforming right pairs (pairs that propagate following the predefined DT) can be found in

practical cryptanalysis. No sophisticated key schedule algorithms or round subkeys are considered as one of the reasons resulting in these incompatibilities.

To guarantee the existence of at least one conforming right pair following the DT. Liu *et al.* [ZDW19] developed an improved MILP model that simultaneously takes into account the propagations of a DT $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ but without considerations of differential probability and message pair $(x_0, x_1, \ldots, x_r)$. By carefully analyzing the relationship between α_i and x_i, their model traces the hybrid path $((\alpha_0, x_0), (\alpha_1, x_1),$
$\ldots, (\alpha_r, x_r))$. Later, Sadeghi, Rijmen, and Bagheri proposed another MILP model to verify a differential [SRB21]. Unlike the Liu *et al.*'s model that traced the difference and the value, the Sadeghi *et al.* approach directly traced the two encrypted values as $((x_0, x_0'), \ldots, (x_r, x_r'))$ and assigned the input and output differences as $\alpha_0 = x_0 \oplus x_0', \alpha_r = x_r \oplus x_r'$ for differential (α_0, α_r).

Both methods require analyses of different operations of primitives to construct the numerous inequalities. For example, Liu *et al.*'s model requires constructing three-part models including difference and value transition model and connection model which is used to describe the relations between difference and value in the nonlinear layer. However, their model becomes more complex with an increasing number of models, resulting in longer run times. Sadeghi *et al.*'s model constructs two value transition models at the same time, and adds some linear constraints to ensure that the XOR of two value transitions satisfy the given DT.

At CRYPTO 2021, Liu *et al.* proposed an algebraic perspective on differential(-linear) cryptanalysis [LLL21]. This novel algebraic perspective pointed out that the output difference of a Boolean function is a special Boolean function of the input difference and input value. For a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ representing a certain output bit of a primitive, the output difference of f with respect to the input difference Δ at a point X is

$$\mathcal{D}_\Delta f(X) = f(X) \oplus f(X \oplus \Delta).$$

Liu *et al.* defined a new Boolean function f_Δ as

$$f_\Delta(X, x) = f(X \oplus x\Delta),$$

where x is an auxiliary binary variable. Then Liu *et al.* gave the following formula

$$\mathcal{D}_x f_\Delta = D_\Delta f, \tag{1}$$

where $\mathcal{D}_x f_\Delta$ is the partial derivative of f_Δ with respect to x. In this paper, we present two more generic and efficient SAT models from an algebraic perspective inspired by Liu *et al.*'s work [LLL21]. The contributions are fivefold as follows.

An efficient SAT model to fast verify DTs. Once the input and output DT (Δ, ∇) is fixed (*i.e.*, they are considered as constants), the output difference is completely determined by X. Therefore, to check whether $\mathcal{D}_x f(X \oplus x\Delta) = \nabla \in \mathbb{F}_2$ holds equals to determine whether a solution X exists for this Boolean equation. In this way, if a right pair $(X, X \oplus \Delta)$ is found, then it confirms the validity of the DT.

A novel SAT model to search DTs and find pairs simutenously. According to Equation 1, the output difference of a Boolean function can be represented as a Boolean expression $\mathcal{D}_x f(X \oplus x\Delta)$, meaning that if the input value X and parts of input and output DT are set as free variables (*i.e.*, we do not specify their values), any solution (X, Δ, ∇) satisfying $\mathcal{D}_x f(X \oplus x\Delta) = \nabla$ is a valid differential with a right pair $(X, X \oplus \Delta)$.

Both of them are SAT problems that can be solved with off-the-shelf SAT solvers. To construct the Boolean equation, we only need to simulate the update of the target cipher to obtain the expression of $\mathcal{D}_x f(X \oplus x\Delta)$, which is easy to handle by symbolic

computation. In this paper, we take `SageMath` [The22] as the symbolic computation tool and the `Bosphorus` [CSCM19] as the SAT solver, which supports Boolean equations as its input.

Applications to Ascon. We examine some differentials proposed in previous forgery and collision attacks on Ascon-AEAD and Ascon-Hash. A 2-round DT that was used in the improved 2-round collision attack on Ascon-Hash, two 3-round DTs and one 4-round DT used in the forgery attacks on the finalization or iteration phases of Ascon-128 [GPT21] as well as a 5-round truncated DT in [DEMS21] are all proved valid. Besides, a 4-round differential leveraged in the forgery attack on Ascon-128 reported in [GPT21] is proven invalid since our verification model proves no right pair exists. Thus, this forgery attack is accordingly invalid.

Applications to Xoodoo. We develope a toolkit called XoodooSat [1] for exhaustively searching all 3-round *differential trail cores* up to weight 50 and two 4-round *differential trail cores*[2] with weight of 80. Each differential trail core actually corresponds to exponentially many real 3- and 4-round DTs expanded by the omitted χ operation. To examine the validity of these DTs, we randomly select tens of thousands of 3-round DTs and two 4-round DTs extended from the differential trail cores, and we find all of them are valid.

Applications to Gimli. We are able to search for a valid 6-round Semi-Free-Start colliding DT and a right pair simultaneously in just 9.74 seconds, which took Liu *et al.* [LIM20] approximately 4 hours. In order to establish a SFS collision attack on the intermediate 8-round Gimli-Hash, Liu *et al.* designed a conditional 8-round DT,*i.e.*, some round difference bits are known. However, their MILP model cannot search such a DT and right pair following this conditional DT in practical time. We apply our search model to search a DT and find a right pair simutenously following Liu *et al.*'s conditional 8-round DT, and it takes us only about one minute. This is a significant improvement compared to Liu *et al.*'s model. With this pair, we successfully mount a practical SFS collision attack on the intermediate 8-round Gimli-Hash.

It is interesting to note that for large-state-size permutation such as Keccak-f, our approach still shows excellent performance. We verify one 4-round DT of Keccak-f[1600] and one 4-round DT of Keccak-f[800] in [GLST22], and confirm that all are valid. For better comparing our method with [SRB21], we construct a SAT model to verify one 4-round DT of Keccak-f[800] and Keccak-f[1600] using the their verification method, which take us approximately 80 seconds and 210 seconds, respectively. However, our method is more efficent, we are able to verify the same 4-round DTs in only 7.89 seconds and 21.59 seconds, respectively.

Table 1 summarizes the best verification times achieved for DTs over various cryptographic primitives. All of our times are solved by `Bosphorus` (v3.0). All experiments [3] are conducted on a server with Inter(R) Xeon(R) CPU E5-4650 v3 @ 2.10GHz 12 Core, 65G RAM, and Ubuntu 18.04.5.

Paper outline. The rest of this paper is organized as follows. In Section 2, we give some concepts used in our work. We describe the full details of our verification and search approach in Section 3. We present the application of our approach on Gimli in Section 5, on Ascon in Section 4 and on Xoodoo in Section 6, respectively. Finally, we conclude our paper in Section 7. Details of experimental data, including the solving times, DCs, and right pairs are given in Appendix.

[1] Please refer to https://github.com/HuinaLi/XoodooSat.

[2] Trail cores, which are equivalence classes for differential trails, group together trails with the same weight in Xoodoo.

[3] The source codes and results are presented at https://github.com/HuinaLi/AlgSAT_a-SAT-Method-for-Search-and-Verification

Table 1: Comparison of our solving times with previous works.

Primitive	Rnd	In Attack	DC from	Validity	Our Time	Pre. Time
Gimli-Hash	6	SFS collision	Tab.3	Valid	9.74s	4h [LIM20]
	8	SFS collision	Tab.5	Valid	66.71s	-† [LIM20]
Ascon-Hash	2	Collision	[GPT21]	Valid	0.02s	-
Ascon-128A	3	Forgery(final.)	[GPT21]	Valid	0.07s	-
	3	Forgery(iter.)	[GPT21]	Valid	0.31s	-
Ascon-128	3	Forgery(final.)	[GPT21]	Valid	0.08s	-
	3	Forgery(iter.)	[GPT21]	Valid	81s	-
	4	Forgery(final.)	[GPT21]	Valid	194s	-
	4	Forgery(iter.)	[GPT21]	Invalid	0.05s	-
	5	-	[DEMS21]	Valid	3894s	-
Xoodoo	3	-	Tab.21	Valid	1.37s	-
	3	-	Tab.25	Valid	1.62s	-
	3	-	Tab.27	Valid	1.08s	-
	3	-	Tab.23	Valid	0.12s	-
	4	-	Tab.29	Valid	1.24s	-
	4	-	Tab.31	Valid	343s	-
Keccak-f[800]	4	Collision	[GLST22]	Valid	7.86s	79s‡[SRB21]
Keccak-f[1600]	4	Collision	[GLST22]	Valid	21.59s	210s‡[SRB21]

†The MILP model in [LIM20] could not return any results in practical time.
‡Using the verification method of [SRB21] to automatically verify DCs but with the help of SAT.

2 Preliminaries

In this section, we give some related terms and properties used in our work.

2.1 Differential Cryptanalysis

In a differential attack, the attacker seeks a fixed input difference α_0 that propagates through an r-round primitive (the primitive could be a block cipher or a permutation) to a fixed output difference α_r with a high probability p, the differential is thus represented by (α_0, α_r).

If there exists an ordered pair $(x, x \oplus \alpha_0)$ satisfying $f(x) \oplus f(x \oplus \alpha_0) = \alpha_r$, then it is said to follow the differential (α_0, α_r). In this case, we call (α_0, α_r) a valid differential, and $(x, x \oplus \alpha_0)$ is called a right pair.

Usually, finding a differential and computing its probability is difficult, so we tend to study the differential properties of every round of the cipher. Let $f = f^{r-1} \circ f^{r-1} \circ \cdots \circ f^0$ be an r-round iterative cipher and α_i, α_{i+1} be the input and output difference of $f^i, 0 \leq i < r$. $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ is called a DT of the cipher f. If there is a vector of variables $(x_0, x_1, \ldots, x_r)$ that satisfies $f^i(x_i) \oplus f^i(x_i \oplus \alpha_i) = \alpha_{i+1}$ for all $0 \leq i < r$, we say that $\{(x_0, x_1, \ldots, x_r), (x_0 \oplus \alpha_0, x_1 \oplus \alpha_1, \ldots, x_r \oplus \alpha_r)\}$ is a right pair following the DT.

Following the Markov cipher assumption [LMM91] where round functions are treated as independent functions, a DT $(\alpha_0, \alpha_1, \ldots, \alpha_r)$, whose probability is computed by $p = \prod_{i=1}^{r} p_i \geq 0$ is valid. However, the independence of round functions might not always hold, especially for permutations without round keys. That means that some differential attacks on certain ciphers might be false since the differentials or DTs used in the attacks might be invalid *i.e.*, there is no right pair following the differential. Therefore, it is necessary to check the validity of DTs of a permutation derived under the Markov assumption.

2.2 SAT-based Cryptanalysis

Given a Boolean formula $f(x_1, x_2, \ldots, x_n)$, the Bool-ean Satisfiability problem (SAT) is to determine whether there is any assignment of values to these Boolean variables which makes the formula true. The SAT problem is satisfiable if a valid assignment exists, otherwise it is unsatisfiable. Most of the previously introduced SAT-based cryptanalysis methods [SWW18, SWW21, GLST22] encode directly the cryptanalysis problem as a SAT instance under the Markov assumption and then invoke the off-the-shelf SAT solver to solve it.

There are many off-the-shelf SAT solvers available which have been introduced into cryptanalysis, such as the `CryptoMiniSat` [SNC09] and `CaDiCaL` [Bie19]. Usually, modern SAT solvers based on *conflict-driven clause learning* (CDCL) [MLM21] support the CNF as their input which uniquely defines a Boolean formula. A formula in CNF consists of clauses joined by *conjunctions* ($\wedge$), where each clause is a *disjunction* ($\vee$) of *literals*, each literal represents a positive or negative variable, *e.g.*, x_i or $\neg x_i$.

However, for cryptanalysts, ANF which consists of $\oplus$ and $\wedge$ is more friendly and preferred to use since the output bits of a cryptographic primitive are naturally written as ANFs of its input bits. Unfortunately, compared with CNF solvers, ANF solvers on huge polynomial systems often use more memory that might be infeasible on many computing platforms.

To fill this vacancy, Davin *et al.* proposed an ANF simplification and solving tool, called `Bosphorus` [CSCM19], which bridges between ANF and CNF solving techniques. The `Bosphorus` supports the ANFs as its input, which could take advantage of the algebra of polynomials naturally. It first uses many optimized mathematical algorithms, including XL [CKPS00], Brickenstein's ANF-to-CNF conversion [BD09], Gauss-Jordan elimination, *etc.*, to simplify ANFs and converted these highly optimized ANFs to CNFs. Afterwards, the SAT solver `Cryptominisat` within `Bosphorus` is invoked to solve those CNFs. The `Bosphorus` can be roughly seen as a SAT solver that supports the ANFs as input.

2.3 A Brief Introduction to Ascon

Ascon [DEMS21] has been announced by NIST as the final winner in the lightweight cryptography standardization competition. The Ascon family consists of AEAD and Hash schemes. Ascon-AEAD adopts a MonkeyDuplex [BDPA11] mode with stronger keyed initialization and keyed finalization phases. Ascon-Hash takes a sponge structure [BDPVA07] and the compressing process. Both schemes operate on a state of 320 bits which they update with two permutations p^a and p^b whose rounds are respectively a and b. The number of rounds a and the number of rounds b are tunable security parameters.

The round function of permutation consists of three steps p_C, p_S, and p_L, denoted by $p = p_L \circ p_S \circ p_C$, operating on a 320-bit state S arranged into five rows, *i.e.*, $S = w_0 \| w_1 \| w_2 \| w_3 \| w_4$, each row is a 64-bit register word. The bits of each 64-bit word are denoted by $S[64i + k], 0 \leq i < 5, 0 \leq k < 64$, where i is the index of row, $S[64i]$ indicates the most significant bit (MSB).

Addition of Constants (p_C). p_C adds a round constant c_i to register word w_2 of the state S in round i. The round constants c_i is shown in Table 2.

Substitution Layer (p_S). p_S operates the state S with 64 parallel applications of the 5-bit Sbox to each bit-slice of the five registers w_0, w_1, w_2, w_3, w_4. We suppose that the input of the single 5-bit Sbox is $(x_0, x_1,$

Table 2: Constants c_i used in the ASCON Permutation

Round i	Constant c_i
0	00000000000000000f0
1	00000000000000000e1
2	00000000000000000d2
3	00000000000000000c3
4	00000000000000000b4
5	00000000000000000a5
6	0000000000000000096
7	0000000000000000087
8	0000000000000000078
9	0000000000000000069
10	000000000000000005a
11	000000000000000004b

$x_2, x_3, x_4)$, and the output is $(y_0, y_1, y_2, y_3, y_4)$. The ANF of the single Sbox is given by

$$\begin{aligned} y_0 &= x_4x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1x_0 \oplus x_1 \oplus x_0 \\ y_1 &= x_4 \oplus x_3x_2 \oplus x_3x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1 \oplus x_0 \\ y_2 &= x_4x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus 1 \\ y_3 &= x_4x_0 \oplus x_4 \oplus x_3x_0 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0 \\ y_4 &= x_4x_1 \oplus x_4 \oplus x_3 \oplus x_1x_0 \oplus x_1 \end{aligned} \tag{2}$$

Linear Diffusion Layer (p_L). p_L provides diffusion within each 64-bit register word $w_i, 0 \leq i < 5$ as follows,

$$\begin{aligned} w_0 &\leftarrow \Sigma_0(w_0) = w_0 \oplus (w_0 \ggg 19) \oplus (w_0 \ggg 28) \\ w_1 &\leftarrow \Sigma_1(w_1) = w_1 \oplus (w_1 \ggg 61) \oplus (w_1 \ggg 39) \\ w_2 &\leftarrow \Sigma_2(w_2) = w_2 \oplus (w_2 \ggg 1) \oplus (w_2 \ggg 6) \\ w_3 &\leftarrow \Sigma_3(w_3) = w_3 \oplus (w_3 \ggg 10) \oplus (w_3 \ggg 17) \\ w_4 &\leftarrow \Sigma_4(w_4) = w_4 \oplus (w_4 \ggg 7) \oplus (w_4 \ggg 41) \end{aligned}$$

Moreover, an r-round DT of ASCON permutation $(\beta_0, \alpha_1, \beta_1, \ldots, \alpha_r)$ is represented in the following form.

$$\beta_0 \xrightarrow{p_S \circ p_C} \alpha_1 \xrightarrow{p_L} \beta_1 \xrightarrow{p_S \circ p_C} \alpha_2 \xrightarrow{p_L} \cdots \xrightarrow{p_S \circ p_C} \alpha_r, \tag{3}$$

where β_i is the input difference of i-th $p_S \circ p_C$, α_{i+1} is the output difference of the i-th p_S, $0 \leq i < r$, and $\beta_i[j]$ is the j-th bit of $\beta_i, 0 \leq j < 320$.

2.4 A Brief Introduction to XOODOO

XOODYAK has been announced by NIST as one of ten finalists for LWC algorithms. XOODOO presented by Daemen *et al.* [DHAK18] in ToSC 2018, is a 384-bit underlying permutation of XOODYAK, the differential nature of the former directly influences the strength of the latter against differential attacks. The state of XOODOO is organized as a 3-dimensional array. Each bit of the array is located by (x, y, z) coordinate where $0 \leq x < 4$, $0 \leq y < 3$ and $0 \leq z < 32$. The state can be broken down into lanes or columns, or planes as shown in Figure 1.

A lane is a 32-bit word, denoted by $S_{y,x}$. A column is operated on 3 bit of y coordinate, indexed by (x, z). A plane is represented as A_y. A state is made up of 12 lanes or 128

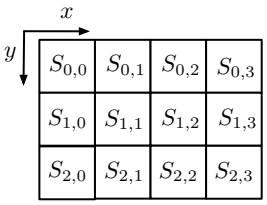

Figure 1: Illustration of the Xoodoo state

columns or 3 planes. Similar to Gimli, in its internal convention, the bits of each 32-bit lane are denoted by $S[32(x+4y)+z]$, where $S[32(x+4y)]$ indicates the LSB.

Xoodoo consists of the iteration of a round function R with 12 rounds, which has the similar design approach as Keccak-p with five step mappings *i.e.*, the linear steps θ, ρ_{west}, ι, ρ_{east}, and the non-linear step χ, denoted by $R = \rho_{east} \circ \chi \circ \iota \circ \rho_{west} \circ \theta$.

Mixing Layer θ. θ is a column parity mixer if the parity of a column is 1, we call it an *odd* (resp. *even*) column that operates as follows. Moreover, $\lll, +$ represent the logic operations rotate left, XOR, respectively.

$$\begin{aligned} P &\leftarrow A_0 + A_1 + A_2 \\ E &\leftarrow P \lll (1,5) + P \lll (1,14) \\ A_y &\leftarrow A_y + E \qquad \text{for } y \in \{0,1,2\} \end{aligned}$$

Diffusion Layer ρ_{west} and ρ_{east}. ρ_{west} and ρ_{east} operate plane A_1 and A_2 by cyclic shift with offsets $(1,0)$ and $(0,11)$ (resp. $(0,1)$ and $(2,8)$), respectively.

$$\begin{aligned} \rho_{west}: & A_1 \leftarrow A_1 \lll (1,0) \qquad A_2 \leftarrow A_2 \lll (0,11) \\ \rho_{east}: & A_1 \leftarrow A_1 \lll (0,1) \qquad A_2 \leftarrow A_2 \lll (2,8) \end{aligned}$$

Addition of Constants ι. ι adds a constant to lane $S_{0,0}$ for each round, which is a critical step in the round function. It can be used to remove symmetry of Xoodoo state. The round constants c_i in hexadecimal notation (see Table **??**), *i.e.*, the least significant bit is at $z = 0$.

Non-linear Layer χ. χ operates in parallel on 3-bit columns and as such forms a layer of 4×32 3-bit Sboxes. For 3-bit units, χ is involutive and hence this also holds for its inverse. χ has algebraic degree two and the ANF of the 3-bit Sbox is given by

$$\begin{aligned} b_0 &= a_0 \oplus (1 \oplus a_1)a_2 \\ b_1 &= a_1 \oplus (1 \oplus a_2)a_0 \\ b_2 &= a_2 \oplus (1 \oplus a_0)a_1 \end{aligned} \tag{4}$$

2.5 A Brief Introduction to Gimli

Gimli [BKL+17] is one of the second-round candidates of the NIST lightweight cryptography standardization process [NIS18], including an authenticated cipher Gimli-Cipher and a hash function Gimli-Hash. Both of them are built upon the Gimli permutation that applies 24 rounds to a 384-bit state. The state of Gimli permutation is organized as a 3×4 matrix of 32-bit words denoted by $S_{i,j}, 0 \le i < 3, 0 \le j < 4$.The j-th column is a sequence of 96 bits such as $S_j = \{S_{0,j}, S_{1,j}, S_{2,j}\}$, the i-th row is a sequence of 128 bits such that $S_i = \{S_{i,0}, S_{i,1}, S_{i,2}, S_{i,3}\}$ (see Figure 2). The bits of each 32-bit word are denoted by $S[32(j+4i)+k], 0 \le k < 32$, where $S[32(j+4i)]$ indicates the least significant bit (LSB). Each round is a sequence of three operations including a non-linear layer which is a 96-bit SP-box (SP) applied to each column, a linear mixing layer including Small-Swap

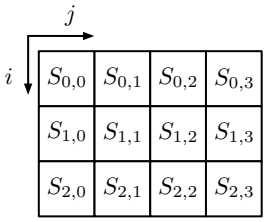

Figure 2: The matrix and indexes of the GIMLI state

(S_SW) and Big-Swap (B_SW) in every second round, and a constant addition (AC) in every fourth round. **SP-boxes** (SP)**.** Each SP-box operates on each column, *i.e.*, 96 bits as follows,

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

Small-Swap (S_SW). In the i-th round satisfying $i \bmod 4 = 0$, we apply Small-Swap operation as follows,

$$S_{0,0}, S_{0,1}, S_{0,2}, S_{0,3} \leftarrow S_{0,1}, S_{0,0}, S_{0,3}, S_{0,2}$$

Big-Swap (B_SW). In the i-th round satisfying $i \bmod 4 = 2$, we apply Big-Swap operation as follows,

$$S_{0,0}, S_{0,1}, S_{0,2}, S_{0,3} \leftarrow S_{0,2}, S_{0,3}, S_{0,0}, S_{0,1}$$

Addition of Constant (AC). When $i \in \{0, 4, 8, 12, 16, 20\}$, AC adds the round constant $\texttt{0x9e377900} \oplus (24 - i)$ to the first state word $S_{0,0}$.

The input state and the intermediate state after i rounds is represented as S^i and $S^{i+1}, 0 \leq i < 24$, respectively.

In addition, the input difference and the intermediate difference after i rounds be ΔS^i and $\Delta S^{i+1}, 0 \leq i < 24$, respectively.

Gimli-Hash. The GIMLI-HASH scheme is built upon the GIMLI using a sponge construction illustrated in Figure 3. Firstly, GIMLI-Hash initializes a 48-byte GIMLI state to all-zero, then reads sequentially through a variable-length input as a series of 16-byte input blocks after padding, *i.e.*, $m_0, m_1, \ldots, m_n$. The block size is the so-called absorbing rate, *i.e.*, 128 bits. The remaining c bits of the state are called the capacity which is not directly affected by message bits, nor are they taken as output. After all message blocks are fully processed, a 32-byte hash value h can be obtained. More details of GIMLI-Hash are given in [BKL+19].

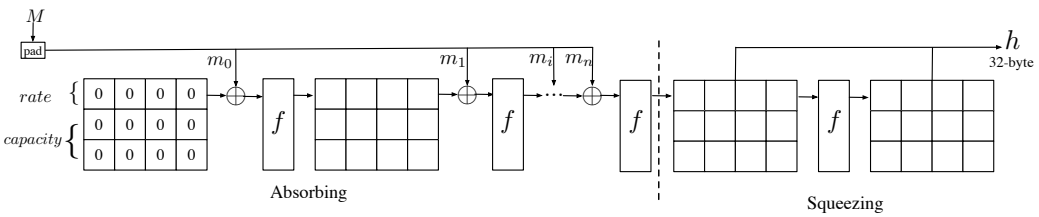

Figure 3: The illustration of the GIMLI-Hash

3 Verification of a Differential or Differential Characteristic from Algebraic Perspective

In this section, based on Liu et al.'s differential-linear cryptanalysis from an algebraic perspective [LLL21], we introduce a new approach to efficiently verify a differential or DT. The basic idea is to transform Equation 1 into a SAT problem which is handy to solve by SAT solvers.

3.1 SAT Model for Verifying a Differential or Differential Trail

Given a cryptographic primitive $E : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we denote its n output bits as n ANFs by $(f_0, f_1, \ldots, f_{n-1})$. We introduce our SAT model for verifying a differential and a DT in two cases, respectively.

Simple case. In the simple case, suppose we can derive the ANFs of all output bits of E. According to Equation 1, to verify a given differential $(\Delta, \nabla) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, the input value X is set as free variables. We need to compute the ANFs of $(f_0(X \oplus x\Delta), f_1(X \oplus x\Delta), \ldots, f_{n-1}(X \oplus x\Delta))$. The output difference $\nabla = (\nabla_0, \nabla_1, \ldots, \nabla_{n-1}) \in \mathbb{F}_2^n$ is thus $(\mathcal{D}_x f_0(X \oplus x\Delta), \mathcal{D}_x f_1(X \oplus x\Delta), \ldots, \mathcal{D}_x f_{n-1}(X \oplus x\Delta))$. Verifying the differential (Δ, ∇) is equivalent to checking if the following equation set is solvable.

$$\begin{cases} \nabla_0 = \mathcal{D}_x f_0(X \oplus x\Delta) \\ \nabla_1 = \mathcal{D}_x f_1(X \oplus x\Delta) \\ \quad \vdots \\ \nabla_{n-1} = \mathcal{D}_x f_{n-1}(X \oplus x\Delta) \end{cases} \tag{5}$$

Note that $\mathcal{D}_x f_i(X \oplus x\Delta)$, where $0 \le i < n - r$, are the ANFs of X. Equation 5 is naturally a SAT problem that can be solved with a SAT solver.

Example 1. Take the 5-bit Sbox of Ascon as an example (the ANFs of the Sbox is presented later in Equation 2 in Section 2.3). Let the input value be $X = (x_0, x_1, x_2, x_3, x_4)$ and the ANFs of the output bits are denoted by $(f_0, f_1, f_2, f_3, f_4)$. To verify whether (Δ, ∇) is a valid differential where $\Delta = (1, 1, 1, 0, 0)$ and $\nabla = (1, 0, 0, 0, 0)$, $X \oplus x\Delta = (x_0 \oplus x, x_1 \oplus x, x_2 \oplus x, x_3, x_4)$, we compute the expressions of the 5-bit output difference according to Equation 5 as follows.

$$\begin{cases} D_x f_0(X \oplus x\Delta) = x_0 \oplus x_2 \oplus x_4 \oplus 1 \\ D_x f_1(X \oplus x\Delta) = x_1 \oplus x_2 \\ D_x f_2(X \oplus x\Delta) = 0 \\ D_x f_3(X \oplus x\Delta) = x_3 \oplus x_4 \oplus 1 \\ D_x f_4(X \oplus x\Delta) = x_0 \oplus x_1 \oplus x_4 \end{cases}$$

Since the output difference is $\nabla = (1, 0, 0, 0, 0)$, we obtain the following five equations.

$$\begin{cases} x_0 \oplus x_2 \oplus x_4 \oplus 1 = 1 \\ x_1 \oplus x_2 = 0 \\ 0 = 0 \\ x_3 \oplus x_4 \oplus 1 = 0 \\ x_0 \oplus x_1 \oplus x_4 = 0 \end{cases}$$

In Example 1, the five equations are easy to solve even by hand. But most of the time, the equations are much more complicated. We regard them as a SAT problem and

use `Bosphorus` to solve these ANFs to decide whether (Δ, ∇) is valid or not by observing whether a solution $(x_0, x_1, x_2, x_3, x_4)$ would be returned.

Complicated case. If the state size of a cryptographic primitive is large, it is computationally infeasible to compute the exact ANFs of the output bits. Inspired by the DATF technique [LLL21], we take advantage of the variable substitutions to simplify the form of the ANFs while retaining the variable x. Suppose E consists of $E = E_{r-1} \circ \cdots \circ E_1 \circ E_0$ where the ANFs of E_i are available, and the output bits of E_i is denoted by $(f_0^{i+1}, f_1^{i+1}, \ldots, f_{n-1}^{i+1})$. To verify a differential (Δ, ∇), we first focus on E_0 and compute the ANFs of $(f_0^1(X \oplus x\Delta), f_1^1(X \oplus x\Delta), \ldots, f_{n-1}^1(X \oplus x\Delta))$. Subsequently, we introduce $2n$ transitional variables a_j^1, b_j^1, where $0 \leq j < n$ to perform the variable substitutions as follows.

$$\begin{cases} f_j^1(X \oplus x\Delta) = b_j^1 \oplus a_j^1 x \\ a_j^1 = \mathcal{D}_x f_j^1(X \oplus x\Delta) \\ b_j^1 = \mathcal{D}_x f_j^1(X \oplus x\Delta)x \oplus f_j^1(X \oplus x\Delta) \end{cases}, \ 0 \leq j < n \tag{6}$$

Based on Equation 6 (which are the ANFs of transitional variables a^1, b^1 and x), we compute the outputs of E_1, *i.e.*, perform similar variable substitutions by introducing $2n$ new transitional variables $a_j^2, b_j^2, 0 \leq j < n$.

$$\begin{cases} f_j^2(b^1 \oplus a^1 x) = b_j^2 \oplus a_j^2 x \\ a_j^2 = \mathcal{D}_x f_j^2(b^1 \oplus a^1 x) \\ b_j^2 = \mathcal{D}_x f_j^2(b^1 \oplus a^1 x)x \oplus f_j^2(b^1 \oplus a^1 x) \end{cases}, \ 0 \leq j < n \tag{7}$$

Repeat this process until the simplified forms of the ANFs of

$$(f_0^r(b^{r-1} \oplus a^{r-1}x), f_1^r(b^{r-1} \oplus a^{r-1}x), \ldots, f_{n-1}^r(b^{r-1} \oplus a^{r-1}x)) \tag{8}$$

is obtained. Likewise, we omit the subscript of f_j^r, and write Equation 8 as $f^r(b^{r-1} \oplus a^{r-1}x)$. Finally, we add constraints on the overall output difference $\nabla = (\nabla_0, \nabla_1, \ldots, \nabla_{n-1})$ with

$$\mathcal{D}_x f_j^r(b^{r-1} \oplus a^{r-1}x) = \nabla_j, 0 \leq j < n.$$

In this way, we get a set of ANFs that determines whether (Δ, ∇) is a valid differential. Obviously, it is also a SAT problem.

3.2 Fast verification for differentials or Differential Trails.

It is easy to adapt the above verification model for a r-round DT $(\Delta^0, \Delta^1, \ldots, \Delta^r)$ where $\Delta^i, 0 < i \leq r$ is the output difference of the $(i-1)$-th round and Δ^0 is the initial input difference. Similarly, ignoring the ANFs of the intermediate difference, we verify the validity of a differential that uses a differential (Δ^0, Δ^r) rather than a specific DT. We introduce our verification Algorithm 1, and successfully apply it to verify DTs of Ascon and Xoodoo in Section 4 and Section 6, respectively.

3.3 Simultaneously searching for Differential trails and right pairs.

In addition to verifying a given differential (Δ, ∇) or a DT $(\Delta^0, \Delta^1, \ldots, \Delta^r)$, our algorithm can also search simultaneously for DTs and find the right pairs. The only distinction is that we do not add constraints on ∇ or $\Delta^i, i \geq 1$ and let parts of these unknown differences be free variables. On the other hand, if the values of some inner variables are given in advance, *e.g.*, when dealing with a conditional DT (parts of difference are unknown), we can fix those variables accordingly as additional constraints. In this way, every solution to

the SAT problem is a valid DT and a right pair. It is especially useful for scenarios where DTs of a specific form, such as collision DTs used in collision attacks, are needed. We apply Algorithm 1 to search for collision DTs of Gimli in Section 5 and it shows outstanding performance.

Algorithm 1 Fast Verification of Differential Trails

Require: An unknown message $X = (x_0, \ldots, x_{n-1})$, the primitive $E = E_{r-1} \circ \cdots \circ E_0$, the number r of rounds, a given DC $(\Delta^0, \Delta^1, \ldots, \Delta^r)$, an auxiliary binary variable x.
Ensure: The value of X or "Invalid".
Initialize the input variable vector $f^0 = X \oplus x\Delta^0$ and allocate a set $Q = \emptyset$;
for i from 0 to $r-1$ **do**
Compute the output of E_i according to the ANF of E_i , $f^{i+1} \leftarrow E_i(f^i)$.
Add $\mathcal{D}_x f^{i+1} = \Delta^{i+1}$ to Q. ▷ For verifying a differential, only when $i = r-1$ we execute this step
Introduce transitional variables a^{i+1}, b^{i+1}, let $f^{i+1} = a^{i+1}x \oplus b^{i+1}$. ▷ The substitution rule is used after the nonlinear operations by default.
Add $a^{i+1} = \mathcal{D}_x f^{i+1}$ and $b^{i+1} = \mathcal{D}_x f^{i+1} x \oplus f^{i+1}$ to Q.
end for
Convert Q into a SAT problem by invoking *cnfwrite*() in `Bosphorus`.
if The SAT problem is feasible **then**
return X
else
return "Invalid"
end if

3.4 Obtaining and Solving the SAT Model

We exploit `SageMath` [The22] to obtain the ANFs of the output bits of a cipher. `SageMath` is a popular tool in cryptanalysis. For example, in [SHW+14], Sun *et al.* took `SageMath` to generate inequalities for a convex hull. `SageMath` also offers good support for calculating Boolean equations (represented by ANFs) over a ring and field. By simulating the round functions of the target cipher with variable substitutions, a set of ANFs linking the input and output differences are established.

`Bosphorus` supporting ANFs as its input, and is able to internally transform them into CNF before solving them. After solving the SAT problem, we examine the returned solution. If no pair exists, the target differential or DT is invalid; otherwise, a confirming right pair will be derived.

3.5 Discussion on Our New Verification Approach

Similar to previous verification algorithms such as [LIM20], our new verification algorithm also traces the propagation of both the values and differences on the target primitive. However, there are some essential differences between our new verification algorithms and [LIM20].

Firstly, the relations between the value transitions and difference transitions are very different. The verification algorithm in [LIM20] derived the relations between the values and differences for the nonlinear functions (the difference transitions and the value transitions are dependent only on the nonlinear operation). For example, to the nonlinear functions of Gimli, the authors of [LIM20] derived four types of Boolean relations between the value and difference transitions. More importantly, their manual analysis is not universal, for different cryptographic primitives we need to analyze their nonlinear operations separately, such as

S-boxes. Instead, the fundamental theory of our algorithm is the algebraic perspective of differential-linear cryptanalysis proposed recently in [LLL21]. The Boolean expressions of the output difference of a cryptographic primitive can be explicitly presented. As shown in Section 3.1, after setting the initial input of the primitive (*i.e.*, the input is $X \oplus x\Delta$), we do not need to worry about the relations between the values and differences over any operation in the process. All we need to do is to simulate the update function by symbolic computations which is friendly to almost all kinds of cryptographic primitives.

Secondly, both our algorithm and [LIM20] try to find a solution for a target differential or DT rather than to prove something to be optimal. Unlike their transformation of the relation into a MILP problem[4], in our algorithm, we choose to use SAT to solve this problem from scratch, since the relations derived from our algorithm are an inherited SAT model with ANF forms. By invoking the `Bosphorus`, we can directly simplify and solve this SAT model. As a result, we find the efficiency of our algorithm is significantly higher than [LIM20]. For example, an 8-round SFS colliding DT and the conforming colliding pair were obtained in only 66.71 seconds using our algorithm, while any of such DTs could not be found by [LIM20] in practical time.

4 Application to Ascon

In this section, we show how to leverage our verification method introduced in Section 3 to verify some DTs proposed in previous forgery and collision attacks on Ascon-AEAD and Ascon-Hash, respectively.

Verify a DT for 2-round Ascon-Hash. Gerault *et al.* [GPT21] used the CP tool to find a new 2-round DT which could also be used in the collision attack on the 2-round Ascon-Hash. Since the DT proposed in [ZDW19] has been proven invalid, a similar case may also occur for this DT. Therefore, it is necessary to check its validity of it.

With `SageMath`, the SAT model is easily constructed. A right pair following this DT is returned in less than one second. Therefore, we confirm that this characteristic is valid.

Check the differentials and DTs in forgery attacks on Ascon-128. The authors [GPT21] proposed several forgery attacks against the finalization and iteration phases of Ascon-AEAD. First, they constructed a CP model to search for forgery DTs with different constraints for different phases. Using these DTs, they improved forgery attacks against the finalization phase, as well as the iteration phase of 3- and 4-round Ascon-128. Again, we need to check these DTs to see if they are valid.

We apply our approach to verify these forgery DTs. For forgery attacks against the finalization phase of 3-round Ascon-128 with 2^{-32} differential provablity (DP), and 4-round Ascon-128 with 2^{-100} DP, we prove that all of these DTs are valid. For forgery attacks against the iteration phase of 3-round Ascon-128 with 2^{-231} DP, we confirm that this DT is also valid and the right pair is obtained.

Additionally, we apply our algorithm to check the 4-round forgery DT in the iteration phase, and our program immediately returns "Invalid". This means that this 4-round DT is invalid. We are interested in what results in its invalidity. To find the contradictions hidden among the DT and message value, we separate the whole 4-round DT into two parts (every part contains 2 rounds) and verify them separately. In the first two rounds, we can obtain the right message pair, but in the second two rounds, our program immediately returns "Invalid" in less one second. Therefore, there are some contradictions hidden in the third and fourth rounds.

Moreover, we are curious whether the corresponding 4-round forgery differential (rather than the DT) is valid. To check it, we remove the restrictions on the internal differences while fixing the input and output difference and run our algorithm again. Surprisingly, the

[4]Actually, their model can also be transformed into a SAT problem with extra works.

loose model is still "Invalid", which means this 4-round differential is impossible for any right pairs, *i.e.*, it is invalid.

5-round results. We verified a 6-round truncated collision-producing DT for Ascon-128 identified in [DEMS21]. However, we are not able to find any solution in practice time (≥ 1 month). Therefore, we only checked the first 5 rounds of 6-round truncated DT. The result shows that `Bosphorus` can return a valid 5-round DT and a right pair simultaneously in about one hour.

4.1 Explaining the Contradiction

In this section, we discuss why the 4-round forgery DC in the iteration phase is invalid. To find the contradictions hidden among the DC and value, we separate the whole 4-round DC into two parts (every part contains two rounds) and verify both of them separately. In the first two rounds, we can obtain the right pair, but in the second two rounds, `Bosphorus` returns "UNSAT" immediately. Therefore, there are some contradictions hidden in the last two rounds.

For a better understanding of what causes the contradiction, we extract all the set ANFs (*i.e.*, all the conditions between the input value and DT of the last two rounds). Suppose that $X = (x_0, x_1, \ldots, x_{319})$ is the input difference of the third round and $(\beta_2, \alpha_3, \beta_3, \alpha_4)$ is the DT of the last two rounds. We can obtain the following set of ANFs, where $f_S^r[i]$ is the ANF of the i -th output bit after the r-th nonlinear operation p_S:

$$\begin{cases} f_S^3 = p_S \circ p_C(X + \beta_2 x) \\ D_x f_S^3[i] = \alpha_3[i] \\ D_x f_S^4[i] = \alpha_4[i] \\ a_i^1 = D_x f_S^3[i] & , 0 \leq i < 320 \\ b_i^1 = D_x f_S^3[i] x \oplus f_S^3[i] \\ x_{320+i} = a_i^1 \\ x_{640+i} = b_i^1 \end{cases} \tag{9}$$

The value of some bits of X and some conditions between X and (β_2, α_3) can be deduced according to $f_S^3 = p_S \circ p_C(X + \beta_2 x)$ and $D_x f_S^3[i] = \alpha_3[i], 0 \leq i < 320$ which is shown in Figure 4.

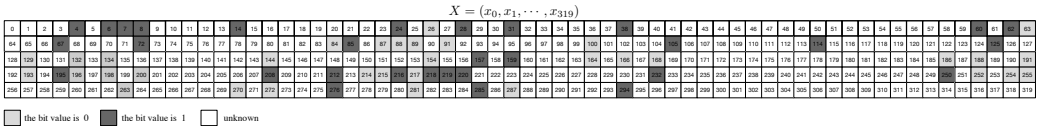

Figure 4: The conditions implied in the third round

In particular, we can get the following constraints,

$$\begin{cases} x_{152} \oplus x_{664} = 0 \\ x_{152} \oplus x_{856} \oplus 1 = 0 \end{cases}$$

Based on the above conditions, we could explain the contradiction. A set of ANFs is extracted using $D_x f_S^4 = \alpha_4$ that leads to a contradiction as follows,

$$\begin{cases} \mathcal{D}_x f_S^4[216] = \alpha_4[216] = 0 \\ \mathcal{D}_x f_S^4[216] = x_{645} \oplus x_{664} \oplus x_{700} \oplus 1 \\ \mathcal{D}_x f_S^4[152] = \alpha_4[152] = 0 \\ \mathcal{D}_x f_S^4[152] = x_{839} \oplus x_{846} \oplus x_{856} \oplus 1 \\ \mathcal{D}_x f_S^4[197] = \alpha_4[197] = 0 \\ \mathcal{D}_x f_S^4[197] = x_{645} \oplus x_{681} \oplus x_{690} \end{cases}$$

Observe the above equation, if we set the 152-th bit of the input value, *i.e.*, x_{152} is 1, then x_{664} is 1 according to $x_{152} \oplus x_{664} = 0$, x_{645} is 1 computed using $\mathcal{D}_x f_S^4[216] \oplus x_{664} \oplus x_{700} \oplus 1$. However, $\mathcal{D}_x f_S^4[197]$, which is derived from our formula for the 197-th bit of the output difference can be calculated from $x_{645} \oplus x_{681} \oplus x_{690}$ and determine the value of $\mathcal{D}_x f_S^4[197]$ as 1. Therefore, there is a contradiction. On the other hand, if we set x_{152} is 0, then x_{856} is 1 according to $x_{152} \oplus x_{856} \oplus 1 = 0$. However x_{856} is 0 computed using $\mathcal{D}_x f_S^4[152] \oplus x_{839} \oplus x_{846} \oplus 1$, so there is also a contradiction.

Therefore, the above conditions cannot hold simultaneously that means there is no right pair following the last two DC, the 4-round forgery DC in [GPT21] is invalid as well.

5 Application to Gimli

A Practical SFS Collision Attack on 8-Round Gimli-Hash

The Semi-Free-Start (SFS) collision attack is one of the four types of collision attacks, where the cryptanalyst can choose the initial chain value, *i.e.*, IV as well as a pair of different messages, *i.e.*, M_1, M_2 such that $H(IV, M_1) = H(IV, M_2)$ [SKP16]. For the first step of the SFS collision attack on Gimli-Hash, we need to find a special DT whose input and output differences are both active only in the rate part. In other words, we need to achieve an inner collision in the capacity part of the Gimli-Hash state. In the second step, by introducing one more pair of message blocks that has the same difference in the rate part, a real SFS collision is successfully converted.

In [LIM20], Liu *et al.* proposed an SFS collision attack on the intermediate 8 rounds of Gimli-Hash. In this attack, they firstly gave a conditional 8-round DT pattern illustrated in Figure 5. The input difference is only injected in $\Delta S^1_{0,3}$ and the difference of several internal state words is conditioned. Later, they constructed a MILP model and expect to search for a specific 8-round DT instance according to the conditional DT and make an inner collision in the capacity simultaneously. However, it is difficult for their MILP model to find such an 8-round DT instance. The Gurobi solver does not output "INFEASIBLE" or any solution for an acceptable time. Thus, their SFS collision attack is in fact unsuccessful.

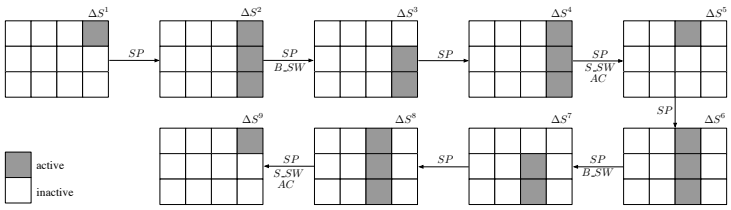

Figure 5: Semi-free-Start collision attack on the intermediate 8-round Gimli-Hash

In our attack, we aim to search for a valid 8-round DT instance according to the conditional DT in [LIM20]. Once we obtain a right pair, we can launch an SFS collision attack. Our approach, as described in Section 3 can easily be adapted to simultaneously

search for DT and find the right pair. Based on the 8-round conditioned DT pattern shown in Figure 5, we present an SFS collision attack.

Firstly, we let active input difference bits be free variables. In [LIM20], the difference $\Delta S^1_{0,3}$ of the conditional DT is active, which means that at least one of the 32 bits of $\Delta S^1_{0,3}$ is nonzero. Therefore, in our attack model, $\Delta S^1_{0,3}$ is represented by 32 unknown binary variables, denoted by $d_0, \ldots, d_{31}$, where the rest of the difference bits of $\Delta S^1_{i,j}$ are zero. For the rest of the round difference, we only add exact ANFs of inactive bits. For example, if $\Delta S^2_{i,j} = 0$, the following 32 ANFs can be obtained

$$\mathcal{D}_x f^1_{32(j+4i)+k} = 0, 0 \leq k < 32$$

where $f^1_{32(j+4i)+k}$ is the ANF of $32(j+4i)+k$-th output bit of the first round function. Next, we take `SageMath` to generate all related ANFs that satisfy the condition of this attack model. We use `Bosphorus` to solve these ANFs. Consequently, a valid 8-round DT and an inner collision are successfully found at the same time. Finally, a real collision can be found by introducing one more pair of message blocks $(M, M \oplus \Delta S^9)$ to absorb the difference in the rate part. Compared with the algorithm in [LIM20], our method shows a much higher efficiency. The SAT solver returns a feasible solution in about one minute.

Applications to 6-round Gimli-Hash. Liu *et al.* used his MILP model to find a valid 6-round SFS collision DT according to the DT pattern in [ZDW19], which cost them about 4 hours. Our verification algorithm is more efficient than their MILP model since it took only 24.11 seconds for us to find a colliding DT and a right pair that satisfies the same DT pattern.

6 Application to Xoodoo

Verification of the DTs for Xoodoo In [BDKA21], Borders *et al.* proved that all 3-round DTs extended from trail cores with weight less than and equal to 50 are all valid, since the differential propagations over any consecutive two rounds are independent.

However, this method is based on some specific conditions, such as the sets of input values and the sets of output values following the given differential over Sboxes are affine subspaces, while our verification approach is quite generic.

In this section, we use our approach to examine these 3-round DTs with weight equal to 36 trying to find at least one right pair for each of them. Although in [BDKA21], Borders *et al.* have shown that any 3-round DTs with weights up to 50 are valid, our verification provides interesting experimental confirmation of their theory. In addition, our approach presents the right pairs for optimal DTs, which gives us more insight about the differential property of Xoodoo.

Given an r-round DT, since the linear layers before and after the χ operations in the first and last rounds, respectively, do not influence its validity, we omit these linear layers in our verification. Consequently, r rounds of Xoodoo can be represented as $R^r = \chi \circ (\iota \circ \rho_{west} \circ \theta \circ \rho_{east} \circ \chi)^{r-1} = \chi \circ (\lambda \circ \chi)^{r-1}$ where $\lambda = \iota \circ \rho_{west} \circ \theta \circ \rho_{east}$ represents the linear operation in the round function.

Let $(\beta_0, \alpha_1, \ldots, \alpha_r)$ be an r-round DT of Xoodoo where β_i, α_{i+1} is the input and output differences of the χ operation of the i-th round, $0 \leq i < r$.

Firstly, the input of the r-round Xoodoo is set as $X \oplus x\beta_0$, where $X = (x_0, x_1, \cdots, x_{383})$ is a 384 binary variable representing one value of the input pair, x is an auxiliary Boolean variable, and β_0 is the input difference. From the symbolic computation, we will accordingly obtain the intermediate states before and after the i-th χ, denoted by f^i and f^i_χ, respectively. Secondly, we add constraints like $\mathcal{D}_x f^i = \beta_i$ and $\mathcal{D}_x f^i_\chi = \alpha_i$ to regulate differential transmission as our predefined DT $(\beta_0, \alpha_1, \ldots, \beta_r)$. After all, these constraints are collected

as a SAT model. If this SAT model is solvable, a solution will be returned. $(X, X \oplus \beta_0)$ is thus a right pair for the DT. Otherwise, this DT is invalid.

Since the χ operation is actually 128 parallel 3-bit Sboxes and all differential propagations over these 3-bit Sboxes have the same weight, *i.e.*, 2, their search algorithm omitted the first and last χ. In other words, the weight of an r-round DT $(\beta_0, \alpha_1, \ldots, \beta_{r-1}, \beta_r)$ is totally determined by the inner state differences, *i.e.*, $(\alpha_1, \ldots, \beta_{r-1})$. $(\alpha_1, \ldots, \beta_{r-1})$ is named the *differential trail core* of $(\beta_0, \alpha_1, \ldots, \beta_{r-1}, \beta_r)$. Since every nonzero input difference of the Xoodoo 3-bit Sbox or its inverse has 4 types of output differences, if the input and output differences of a differential trail core have w_i and w_o activate Sboxes, respectively, this differential trail core can be extended at most to $2^{2(w_i+w_o)}$ DTs. It is too costly to examine all these extended 3-round DTs, so we randomly select 2^{14} 3-round DTs from each of the four differential trail cores with the lowest weight 36. Finally, these 4×2^{14} 3-round DTs are all valid. This implies that there are no contradictions in the conditions imposed by the round differentials of the DTs in the trail core.

Similarly to the 3-round cases, 4-round DT can also be determined by their differential trail core. In [DMA22], the authors have proven the theoretical lower bound of 4-round DT with weight 80, but did not provide any concrete 4-round DTs in the literature.

To obtain some 4-round DT instances, we independently search for 4-round trail cores with the help of our XoodooSat toolkit. We successfully find two 4-round trail cores of weight 80. We verify two 4-round DTs extended from two 4-round trail cores, and all these two 4-round DTs are also valid.

Applications to Keccak. Since Xoodoo and Keccak share lots of similarities, it is smooth for us to adapt the verification algorithm to examine the differentials for Keccak. We verify one 4-round DT of Keccak-f[1600] and one 4-round DT of Keccak-f[800] (see Table 8 and Table 9 in [GLST22]) with the weight of 133 and 95, respectively, and confirm that all of them are valid.

7 Conclusion

In this paper, we presented an efficient and quite generic automatic verification method from an algebraic perspective. Our method can additionally be used as a right pair generation tool to quickly correct message pairs satisfying the given DT. We demonstrate the power of our approach by verifying the validity of the DTs of Gimli, Ascon, Xoodoo, and Keccak and directly searching for a valid DT of Gimli. We successfully mounted a SFS collision attack on the intermediate 8-round Gimli-Hash by searching for a valid DT as well as finding an inner message pair only in about one minute.

Additionally, our method can also serve as a useful tool to observe the interaction between the linear and non-linear layers in cryptographic algorithms. We found that the published forgery attacks of Ascon-128 are invalid because the 4-round forgery DT in the iteration phase is invalid, which means that there are some contradictions in the conditions imposed by the round differentials of the DT.

Actually, our tool is suitable also for many other cryptographic primitives, such as keyed permutations. According to $D_x f_\Delta = D_\Delta f$, the set of ANFs of the output difference ∇ can be represented as $D_x f(X \oplus x\Delta, K)$. Once we have fixed the input difference Δ, the output difference is therefore completely determined by the input value X and the key value K. Therefore, to check whether the differential (Δ, ∇) is valid, it is equivalent to seeing whether there is a solution (X, K) for this set of ANFs. However, it seems infeasible for higher-degree primitives, which will lead to a considerable complexity SAT model, SAT solver cannot return any solutions in practice time. In the future, we will consider improving our approach to verify longer DCs as well as higher-degree round functions.

References

[AST+17] Ahmed Abdelkhalek, Yu Sasaki, Yosuke Todo, Mohamed Tolba, and Amr M. Youssef. MILP Modeling for (Large) S-boxes to Optimize Probability of Differential Characteristics. *IACR Trans. Symmetric Cryptol.*, 2017(4):99–129, 2017.

[BD09] Michael Brickenstein and Alexander Dreyer. PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. *J. Symb. Comput.*, 44(9):1326–1345, 2009.

[BDKA21] Nicolas Bordes, Joan Daemen, Daniël Kuijsters, and Gilles Van Assche. Thinking Outside the Superbox. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference*, volume 12827, pages 337–367, Cham, 2021. Springer.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *Selected Areas in Cryptography - 18th International Workshop, SAC 2011*, volume 7118, pages 320–337, Berlin, Heidelberg, 2011. Springer.

[BDPVA07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007, 2007.

[Bie19] Armin Biere. CADICAL at the SAT Race 2019. 2019. https://github.com/arminbiere/cadical.

[BKL+17] Daniel J. Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke Todo, and Benoît Viguier. Gimli : A Cross-Platform Permutation. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference*, volume 10529, pages 299–320, Cham, 2017. Springer.

[BKL+19] Daniel J. Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke Todo, and Benoît Viguier. Gimli 20190329, 2019. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-2-Candidates.

[BM22] Emanuele Bellini and Rusydi H. Makarim. Functional Cryptanalysis: Application to reduced-round Xoodoo. *IACR Cryptol.*, 2022:134, 2022. https://eprint.iacr.org/2022/134.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques*, volume 1807, pages 392–407, Berlin, Heidelberg, 2000. Springer.

[CSCM19] Davin Choo, Mate Soos, Kian Ming Adam Chai, and Kuldeep S. Meel. Bosphorus: Bridging ANF and CNF Solvers. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019*, pages 468–473, 2019.

[DDS12] Itai Dinur, Orr Dunkelman, and Adi Shamir. New Attacks on Keccak-224 and Keccak-256. In *Fast Software Encryption - 19th International Workshop, FSE 2012*, volume 7549, pages 442–461, Berlin, Heidelberg, 2012. Springer.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

[DHAK18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of Xoodoo and Xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

[DHAK21] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. XooTools software, 2021. https://github.com/KeccakTeam/Xoodoo.

[DHP+20] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.

[DMA22] Joan Daemen, Silvia Mella, and Gilles Van Assche. Tighter trail bounds for Xoodoo. *IACR Cryptol.*, 2022:1088, 2022. https://eprint.iacr.org/2022/1088.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, Berlin, Heidelberg, 2002.

[GD07] Vijay Ganesh and David L. Dill. A decision procedure for bit-vectors and arrays. In *Computer Aided Verification, 19th International Conference*, volume 4590, pages 519–531, Berlin, Heidelberg, 2007. Springer.

[GLST22] Jian Guo, Guozhen Liu, Ling Song, and Yi Tu. Exploring SAT for Cryptanalysis: (Quantum) Collision Attacks against 6-Round SHA-3. *IACR Cryptol.*, 2022:184, 2022. https://eprint.iacr.org/2022/184.

[GMS16] David Gérault, Marine Minier, and Christine Solnon. Constraint Programming Models for Chosen Key Differential Cryptanalysis. In *Principles and Practice of Constraint Programming - 22nd International Conference*, volume 9892, pages 584–601, Cham, 2016. Springer.

[GPT21] David Gérault, Thomas Peyrin, and Quan Quan Tan. Exploring Differential-Based Distinguishers and Forgeries for ASCON. *IACR Trans. Symmetric Cryptol.*, 2021(3):102–136, 2021.

[HLJ+20] Xichao Hu, Yongqiang Li, Lin Jiao, Shizhu Tian, and Mingsheng Wang. Mind the Propagation of States - New Automatic Search Tool for Impossible Differentials and Impossible Polytopic Transitions. In *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security*, volume 12491, pages 415–445, Cham, 2020. Springer.

[IMM18] Alexey Ignatiev, António Morgado, and João Marques-Silva. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference*, volume 10929, pages 428–437, Cham, 2018. Springer.

[LIM20] Fukang Liu, Takanori Isobe, and Willi Meier. Automatic Verification of Differential Characteristics: Application to Reduced Gimli. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference*, volume 12172, pages 219–248, Cham, 2020. Springer.

[LLL21] Meicheng Liu, Xiaojuan Lu, and Dongdai Lin. Differential-Linear Cryptanalysis from an Algebraic Perspective. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference*, volume 12827, pages 247–277, Cham, 2021. Springer.

[LMM91] Xuejia Lai, James L. Massey, and Sean Murphy. Markov Ciphers and Differential Cryptanalysis. In *Advances in Cryptology — EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques*, volume 547, pages 17–38, Berlin, Heidelberg, 1991. Springer.

[MLM21] João Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-Driven Clause Learning SAT Solvers. In *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 133–182. IOS Press, Ohmsha, 2021.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. *IACR Cryptol.*, 2013:328, 2013. https://eprint.iacr.org/2013/328.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In *Information Security and Cryptology - 7th International Conference, Inscrypt 2011*, volume 7537, pages 57–76, Berlin, Heidelberg, 2011. Springer.

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT Solvers to Cryptanalysis of Hash Functions. In *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference*, volume 4121, pages 102–115, Berlin, Heidelberg, 2006. Springer.

[NIS18] NIST. The NIST lightweight cryptography project, 2018. https://csrc.nist.gov/Projects/lightweight-cryptography.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security*, volume 8873, pages 158–178, Berlin, Heidelberg, 2014. Springer.

[Sin05] Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference*, volume 3709, pages 827–831, Berlin, Heidelberg, 2005. Springer.

[SKP16] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart Collision for Full SHA-1. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 9665, pages 459–483, Berlin, Heidelberg, 2016. Springer.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT Solvers to Cryptographic Problems. In *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584, pages 244–257, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[SRB21] Sadegh Sadeghi, Vincent Rijmen, and Nasour Bagheri. Proposing an MILP-based method for the experimental verification of difference-based trails: application to SPECK, SIMECK. *Des. Codes Cryptogr.*, 89(9):2113–2155, 2021.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang. More Accurate Differential Properties of LED64 and Midori64. *IACR Trans. Symmetric Cryptol.*, 2018(3):93–123, 2018.

[SWW21] Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the Search of Differential and Linear Characteristics with the SAT Method. *IACR Trans. Symmetric Cryptol.*, 2021(1):269–315, 2021.

[The21] The Keccak Team. Updated bounds on differential and linear trails in Xoodoo, 2021. https://keccak.team/2021/updated_bounds_xoodoo.html.

[The22] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5s)*, 2022. https://www.sagemath.org.

[ZDW19] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Collision Attacks on Round-Reduced Gimli-Hash/Ascon-Xof/Ascon-Hash. *IACR Cryptol.*, 2019:1115, 2019. https://eprint.iacr.org/2019/1115.

A Gimli

Table 3: The differential characteristic for SFS 6-round Gimli-Hash

ΔS^0			
00000000	c803ec98	00000000	c803ec98
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
ΔS^1			
00000000	00000000	00000000	00000000
00000000	a9580034	00000000	a9580034
00000000	98c803ec	00000000	98c803ec
ΔS^2			
00000000	a8c8203e	00000000	a8c8203e
00000000	a0106912	00000000	a0106912
00000000	319026b0	00000000	319026b0
ΔS^3			
00000000	800100f0	00000000	800100f0
00000000	000ae000	00000000	000ae000
00000000	9bc00080	00000000	9bc00080
ΔS^4			
00000000	00000080	00000000	00000080
00000000	00400000	00000000	00400000
00000000	80000000	00000000	80000000
ΔS^5			
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	80000000	00000000	80000000
ΔS^6			
00000000	80000000	00000000	80000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Table 4: Collision message pair for SFS 6-round Gimli-Hash

X			
e06d07af	445efb01	1a06fc49	47fefb01
46e37879	00d119d0	807e0345	00d11880
33682fd3	03332212	7e8d4676	8334b212
$X \oplus \Delta S^0$			
e06d07af	8c5d1799	1a06fc49	8ffd1799
46e37879	00d119d0	807e0345	00d11880
33682fd3	03332212	7e8d4676	8334b212

Table 5: The differential characteristic for intermediate 8-round Gimli-Hash: Round 1 to 9

ΔS^1			
00000000	00000000	00000000	81c18ba0
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
ΔS^2			
00000000	00000000	00000000	00000c50
00000000	00000000	00000000	e182408d
00000000	00000000	00000000	a081c18b
ΔS^3			
00000000	00000000	00000000	00000000
00000000	00000000	00000000	b4821adb
00000000	00000000	00000000	13068d32
ΔS^4			
00000000	00000000	00000000	361f001b
00000000	00000000	00000000	0035a72d
00000000	00000000	00000000	3a9b6b80
ΔS^5			
00000000	00000000	99e74180	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
ΔS^6			
00000000	00000000	004c0800	00000000
00000000	00000000	808967c3	00000000
00000000	00000000	8099e741	00000000
ΔS^7			
00000000	00000000	00000000	00000000
00000000	00000000	13ed158b	00000000
00000000	00000000	03101f8a	00000000
ΔS^8			
00000000	00000000	186bb8bd	00000000
00000000	00000000	dc0b0437	00000000
00000000	00000000	4e8c65a4	00000000
ΔS^9			
00000000	00000000	00000000	0806669c
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Table 6: Collision message pair for intermediate 8-round Gimli-Hash: Round 1 to 9

X			
8e57bfca	da90441d	9134941b	3a78650c
cd0c5da0	576ee7cd	7081c41a	df260717
2a98b7a5	02fd11bb	21954066	8e042b58
$X \oplus \Delta S^1$			
8e57bfca	da90441d	9134941b	bbb9eeac
cd0c5da0	576ee7cd	7081c41a	df260717
2a98b7a5	02fd11bb	21954066	8e042b58

B Ascon

B.1 Differential characteristics for Ascon-Hash and Conforming message pair for Forgery characteristics

Table 7: Conforming pair of 2-round differential characteristic for Ascon-Hash

DC	X	$X \oplus \beta_0$
[GPT21]	5c41069d791c645a	e70405b8a01771db
	bf65e7a5df0e6f0d	bf65e7a5df0e6f0d
	f2ced15c537c9796	f2ced15c537c9796
	36dcef2e451453f9	36dcef2e451453f9
	89b3344098ab8458	89b3344098ab8458

Table 8: Forgery characteristics for round-reduced Ascon-128A with a 3-round finalization in [GPT21]

β_0	α_1	α_2	α_3
0000000000000001	0000000000000000	0000000000000000	????????????????
0000000000000001	0000000000000000	0000000000000000	????????????????
0000000000000000	0000000000000001	8400000000000001	????????????????
0000000000000000	0000000000000000	8400000000000001	4010000000000000
0000000000000000	0000000000000000	0000000000000000	8461c20000000001

Table 9: Conforming pair of 3-round differential characteristic in Ascon-128A finalization phase [GPT21]

DC	X	$X \oplus \beta_0$
Table 8	30f78a1b80841d90	30f78a1b80841d91
	749dc43f87b6928d	749dc43f87b6928c
	a87e64223e33d3d3	a87e64223e33d3d3
	99e546e5528e8c4f	99e546e5528e8c4f
	67794a4a79d44a79	67794a4a79d44a79

Table 10: Forgery characteristics for round-reduced Ascon-128A with a 3-round permutation in [GPT21]

β_0	α_1	α_2	α_3
0040000400001004	0000000000000000	0240000402001004	2655811c3605b004
0000000000000000	0040000400001004	020080080a002024	2445011424009000
0000000000000000	0000000000000001	0000000000000000	0000000000000000
0000000000000000	0000000000000000	0000000000000000	0000000000000000
0000000000000000	0040000400001004	0a00800800002004	0000000000000000

Table 11: Conforming pair of 3-round differential characteristic in Ascon-128A permutation phase [GPT21]

DC	X	$X \oplus \beta_0$
Table 10	7c5db1d57b1562b1	7c1db1d17b1572b5
	c4ff06bf3619df87	c4ff06bf3619df87
	40078ce677be3196	40078ce677be3196
	7148974d0af4a995	7148974d0af4a995f
	6f85f592f9f630f0	6f85f592f9f630f0

Table 12: Forgery characteristics for round-reduced Ascon-128 with a 3-round finalization in [GPT21]

β_0	α_1	α_2	α_3
0000000000000001	0000000000000000	0000000000000000	????????????????
0000000000000000	0000000000000001	0200000000800000	????????????????
0000000000000000	0000000000000000	0000000002000009	????????????????
0000000000000000	0000000000000000	0000000002000008	b6010000050c0005
0000000000000000	0000000000000001	0200000000800000	0000000002008108

Table 13: Conforming pair of 3-round differential characteristic in Ascon-128 finalization phase [GPT21]

DC	X	$X \oplus \beta_0$
Table 12	df63a162860c7ade	df63a162860c7adf
	18201fba224f0c6d	18201fba224f0c6d
	a742e064fcafe921	a742e064fcafe921
	644c3786e3445133	644c3786e3445133
	06692cbc49174b50	06692cbc49174b50

Table 14: Forgery characteristics for round-reduced Ascon-128 with a 3-round permutation in [GPT21]

β_0	α_1	α_2	α_3
04000a0080014000	0000000000000000	80405826050100c0	f34a5fa78bdbc6dc
0000000000000000	04000a0080014000	054802b6010142c1	0000000000000000
0000000000000000	0000000000000000	a108588200010000	0000000000000000
0000000000000000	0000080000014000	a10858b6010142c0	0000000000000000
0000000000000000	04000a0080014000	a0081a9684034255	0000000000000000

Table 15: Conforming pair of 3-round differential characteristic in Ascon-128 iteration phase [GPT21]

DC	X	$X \oplus \beta_0$
Table 14	e3cd5bcd10216032	e7cd51cd90202032
	bed0df80d77d704a	bed0df80d77d704a
	704250acfea562a6	704250acfea562a6
	bc1556cc98493d47	bc1556cc98493d47
	404461840a57a8e8	404461840a57a8e8

Table 16: Forgery characteristics for round-reduced Ascon-128 with a 4-round permutation in [GPT21]

β_0	β_1	β_2	β_3
0000028000000000	0000000000000000	40008a4400402004	c1868824c0ca3030
0000000000000000	0000168000000005	4a00902d0280002b	8584d48c4ae22035
0000000000000000	0000000000000000	10001d5800000006	8082d4a448e20035
0000000000000000	0000000000000000	010214050a000004	40022a3085081140
0000000000000000	4000028500000001	4a8016a80800000f	0504729c47602140
β_4			
7d0b515048524344			
0000000000000000			
0000000000000000			
0000000000000000			
0000000000000000			

Table 17: Forgery characteristics for round-reduced Ascon-128 with a 4-round finalization in [GPT21]

β_0	α_1	α_2	α_3
0000000000000001	0000000000000001	0000201000000000	0200000000008000
0000000000000000	0000000000000001	0000201002000008	2000009004000000
0000000000000000	0000000000000000	0000000002000009	840120900308000d
0000000000000000	0000000000000000	0000000002000008	8605008005080005
0000000000000000	0000000000000000	0000000000000000	0200000004008100
α_4			
????????????????			
????????????????			
????????????????			
6011b00846802008			
856042820100c081			

Table 18: Conforming pair of 4-round differential characteristic in Ascon-128 finalization phase [GPT21]

DC	X	$X \oplus \beta_0$
	bd2445510dbd4c88	bd2445510dbd4c89
	5896c3af6f2ad294	5896c3af6f2ad294
Table 17	17f30c7ea871c0b0	17f30c7ea871c0b0
	e615b4b418a723b3	e615b4b418a723b3
	ce94413027760a9c	ce94413027760a9c

Table 19: The first 5 rounds differential characteristic (see Table 14-(a) in [DEMS21])

β_0	α_1	α_2	α_3
8000000000000000	0000000000000000	0000000000000000	0002000001824082
0000000000000000	8000000000000000	0100000000400000	9802a00000c64004
0000000000000000	0000000000000000	0000000001000004	1802800002c60006
0000000000000000	0000000000000000	0000000001000004	1800800002c60082
0000000000000000	8000000000000000	8100000000400000	8900200003004084
α_4	α_5		
2884024003c2a856	5a82d45841828c2a		
a4a4e8e000e0c182	c302ce434f290881		
74a062800e68cd21	1b2476214c4304cf		
1473caa04e4a3d61	9ba0b61b010c84c9		
a8f024000e847094	d2a2781b054708e6		

Table 20: Conforming pair of the first 5 rounds differential characteristic [DEMS21]

DC	X	$X \oplus \beta_0$
Table 19	f9434e1234f7d97e	79434e1234f7d97e
	acaebc0c445d988a	acaebc0c445d988a
	f5e5b6cc63c44934	f5e5b6cc63c44934
	d7c6c281c4dadfd3	d7c6c281c4dadfd3
	499d7613b65f59be	499d7613b65f59be

C Xoodoo

Table 21: No.1 Optimal differential characteristic for 3-round Xoodoo

β_0			
00000002	00000000	00000000	00000000
00000001	00000000	00000000	00000000
00000000	00000000	00000000	00000000
α_1			
00000002	00000000	00000000	00000000
00000001	00000000	00000000	00000000
00000000	00000000	00000000	00000000
α_2			
00000002	00000000	00000000	00000000
00000000	00000002	00000000	00000000
00000000	00000000	00000000	00000000
α_3			
00000002	00008040	00010080	00000000
00000000	00000000	00008044	00010080
00000000	04020000	08040000	00000000

Table 22: Conforming pair for No.1 Optimal 3-round differential characteristic of Xoodoo

X			
da809d9d	3c7886f7	2eda462a	d60e0b26
210da579	a33b2733	c476e74e	1ce39c19
080e8bee	78a93341	f523e2b0	ff95b517
$X \oplus \beta_0$			
da809d9f	3c7886f7	2eda462a	d60e0b26
210da579	a33b2733	c476e74e	1ce39c19
080e8bee	78a93341	f523e2b0	ff95b517

Table 23: No.2 Optimal differential characteristic for 3-round Xoodoo

β_0			
04000000	00000000	00000000	00000100
02000000	00000000	00000040	00000000
80000000	00000001	00000000	00000000
α_1			
04000000	00000000	00000000	00000100
02000000	00000000	00000040	00000000
80000000	00000001	00000000	00000000
α_2			
04000000	00000000	00000000	00000100
00000000	04000000	00000000	00000080
00000000	00000000	00040000	00080000
α_3			
04000000	00000000	00000000	00000100
00000100	00000000	08000000	00000000
00000020	00000040	00000000	00000000

Table 24: Conforming pair for No.2 Optimal 3-round differential characteristic of Xoodoo

X			
02003012	c70546a6	93480448	143ac404
80001012	00420009	41080000	20050401
04102000	c74546a6	d2401408	143fc107
$X \oplus \beta_0$			
06003012	c70546a6	93480448	143ac504
82001012	00420009	41080040	20050401
84102000	c74546a7	d2401408	143fc107

Table 25: No.3 Optimal differential characteristic for 3-round Xoodoo

β_0			
00000000	00000000	00000100	00000000
00000000	00000000	00000000	00000000
00000001	00000000	00000000	00000000
α_1			
00000000	00000000	00000100	00000000
00000000	00000000	00000000	00000000
00000001	00000000	00000000	00000000
α_2			
00000000	00000000	00000100	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00080000	00000000
α_3			
00000000	00000201	00000100	00402000
00402000	00000000	00000201	00000000
00000040	00100800	00000000	01000002

Table 26: Conforming pair for No.3 Optimal 3-round differential characteristic of Xoodoo

X			
563794c2	088415d9	29a33b86	40abecb4
d37ced1f	ead842d6	b6782c9e	5473208f
98dfd793	087c4616	950b8157	c0f590d1
$X \oplus \beta_0$			
563794c2	088415d9	29a33a86	40abecb4
d37ced1f	ead842d6	b6782c9e	5473208f
98dfd792	087c4616	950b8157	c0f590d1

Table 27: No.4 Optimal differential characteristic for 3-round Xoodoo

β_0			
00000000	00000000	00000000	00000000
00000000	00000000	00000080	00000000
00000001	00000000	00000000	00000000
α_1			
00000000	00000000	00000000	00000000
00000000	00000000	00000080	00000000
00000001	00000000	00000000	00000000
α_2			
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000100
00000000	00000000	00080000	00000000
α_3			
00804000	00000201	00000000	00000000
00000200	00804000	00000201	00000000
02000044	00100800	00000000	00000000

Table 28: Conforming pair for No.4 Optimal 3-round differential characteristic of Xoodoo

X			
fe518252	bd8ce600	ff7524d7	0a250b5e
ee299823	b23e3283	ecd1ba1d	6db528a5
cd66b57e	5c846c0a	11017b10	fa114ef8
$X \oplus \beta_0$			
fe518252	bd8ce600	ff7524d7	0a250b5e
ee299823	b23e3283	ecd1ba9d	6db528a5
cd66b57f	5c846c0a	11017b10	fa114ef8

Table 29: No.1 Differential characteristic for 4-round Xoodoo

β_0			
0000000e	0000000a	0000000e	0000000a
00000001	00000005	00000001	00000005
00000000	00000000	00000000	00000000
α_1			
0000000e	0000000a	0000000e	0000000a
00000007	00000005	00000007	00000005
00000000	00000000	00000000	00000000
α_2			
0000000c	0000000c	0000000c	0000000c
00000006	00000006	00000006	00000006
00000000	00000000	00000000	00000000
α_3			
00000008	00000008	00000008	00000008
00000004	00000004	00000004	00000004
00000000	00000000	00000000	00000000
α_4			
00000008	00000008	00000008	00000008
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Table 30: Conforming pair for No.1 4-round differential characteristic of Xoodoo

X			
a2cd7e45	cd20443d	dc07ec29	f54c2747
7bf33691	b3816090	e7a24d70	8fa6c550
5fb494b8	76f2590a	26a4b278	bb762ada
$X \oplus \beta_0$			
a2cd7e4b	cd204437	dc07ec27	f54c274d
7bf33690	b3816095	e7a24d71	8fa6c555
5fb494b8	76f2590a	26a4b278	bb762ada

Table 31: No.2 Differential characteristic for 4-round Xoodoo

β_0			
01000100	00000000	00010001	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
α_1			
01000100	00000000	00010001	00000000
00000000	00000000	00000000	00000000
01000100	00000000	00010001	00000000
α_2			
01080108	00000000	08010801	00000000
00000000	00000000	00000000	00000000
01080108	00000000	08010801	00000000
α_3			
41084108	00000000	00410041	00000000
00000000	00000000	00000000	00000000
41004100	00000000	08410841	00000000
α_4			
41084108	00000000	00410041	00000000
00000000	00000000	00000000	00000000
02000200	00000000	08020802	00000000

Table 32: Conforming pair for No.2 4-round differential characteristic of Xoodoo

X			
75bb818b	566f2118	19861633	020c4018
b165f93e	007bd8f6	c2b3a741	fe8eb1b6
a52299b9	44a359d1	1c1dca29	c3f32077
$X \oplus \beta_0$			
74bb808b	566f2118	19871632	020c4018
b165f93e	007bd8f6	c2b3a741	fe8eb1b6
a52299b9	44a359d1	1c1dca29	c3f32077

D Keccak

Table 33: Conforming pair for 4-round differential characteristic of Keccak-f[800] [GLST22]

		X		
6e0afffe	fd800d8a	4022d287	b3537a30	d6b65c78
be9ede92	f5464b9a	6cedf67b	38a53a33	0ed777f9
7b0a0f76	680cd690	bba798b6	349ffde5	7e57d84a
f45f264a	41a1e30f	c9101439	a10a3fb2	07e3ba49
a532921a	611a224e	e027aa10	36804867	fc42e2e6
		$X \oplus \beta_0$		
2e0afffe	d9800d8a	4022d287	b3537a30	d6b65c78
fe9ede92	f5464b9a	4cedf67b	78a53a33	0ed777f9
3b0a0f76	680cd690	9ba798b6	349ffde5	7e57d84a
f45f264a	45a1e30f	e9101439	a10a3fb2	07e3ba49
e532921a	611a224e	c027aa10	36804867	fc42e2e6

Table 34: Conforming pair for No.1 4-round differential characteristic of Keccak-f[1600] [GLST22]

		X		
315671d0d0071602	42fd1c41f82b838b	889cdc68f9955a8a	09687aad0335ab19	22c4588ab8ff3e66
3b9e4c0062cdcfc0	5cf91c118a6b9d29	6a02da273a3c06eb	f4a05ed9a49cae3a	aec18c639c6f7a8e
0aec05b00e3d51c0	5e6993b7cab8dc03	98fea95e6a2fb529	05058568b40ea8cf	cd8777e845686eae
9b85ebd1b9ab6716	fd65f99a6f278aa0	3c55b89a8c46af0f	050fad93c11d356c	2baf07920fdec3af
dfe99de2b880f360	55e35833d55f5c2a	f5b1745d702e2291	2db565d2c98140cd	29e6612766e261d6
		$X \oplus \beta_0$		
315671d0d0071606	42fd1c41f82b838b	889cdc68f9955a8a	09687aad0335ab1b	22c4588ab8ff3e6e
3b9e4c0062cdcfc4	5cf91c118a6b9d29	6a02da273a3c06eb	f4a05ed9a49cae38	aec18c639c6f7a86
0aec05b00e3d51c4	5e6993b7cab8dc03	98fea95e6a2fb529	05058568b40ea8cd	cd8777e845686ea6
9b85ebd1b9ab6712	fd65f99a6f278aa0	2c55b89a8c46af0f	050fad93c11d356e	2baf07920fdec3af
dfe99de2b880f364	55e35833d55f5c2a	e5b1745d702e2293	2db565d2c98140cd	29e6612766e261de