Finding Collisions for Round-Reduced Romulus-H

Marcel Nageler, Felix Pallua and Maria Eichlseder

Graz University of Technology, Graz, Austria
marcel.nageler@iaik.tugraz.at, felix.pallua@student.tugraz.at,
maria.eichlseder@iaik.tugraz.at

Abstract. Romulus-H is a hash function that currently competes as a finalist in the NIST Lightweight Cryptography competition. It is based on the Hirose DBL construction which is provably secure when used with an ideal block cipher. However, in practice, ideal block ciphers can only be approximated. The security of concrete instantiations must be cryptanalyzed carefully; the security margin may be higher or lower than in the secret-key setting. So far, the Hirose DBL construction has been studied with only a few other block ciphers, like IDEA and AES. However, Romulus-H uses Hirose DBL with the SKINNY block cipher where no dedicated analysis has been published so far.
In this work, we present the first third-party analysis of Romulus-H. We propose a new framework for finding collisions in hash functions based on the Hirose DBL construction. This is in contrast to previous work that only focused on free-start collisions. Our framework is based on the idea of *joint differential characteristics* which capture the relationship between the two block cipher calls in the Hirose DBL construction. To identify good joint differential characteristics, we propose a combination of a MILP and CP model. Then, we use these characteristics in another CP model to find collisions. Finally, we apply this framework to Romulus-H and find practical collisions of the hash function for 10 out of 40 rounds and practical semi-free-start collisions up to 14 rounds.

Keywords: Hash functions · Differential cryptanalysis · MILP · SMT · Romulus-H

1 Introduction

Hash functions are among the most versatile and widely-used cryptographic schemes. However, compared to other cryptographic primitives such as keyed block ciphers, estimating the security margin of hash functions appears even more challenging. Most applications rely on standardized hash functions such as the NIST standards SHA-2 and SHA-3, which have undergone substantial scrutiny. Both are however not well-suited for more constrained platforms, which may struggle with the large state size of the Keccak permutation or the software-oriented design of SHA-2. For this reason, the National Institute of Standards and Technology (NIST) is currently running the Lightweight Cryptography (LWC) standardization project, a cryptographic competition for lightweight authenticated encryption schemes (AEAD) and hash functions.

One of the 10 finalists in the NIST LWC competition is the Romulus cipher suite, published by Iwata et al. at FSE 2020 [IKMP20]. Romulus comprises multiple authenticated encryption modes for different use-cases and the hash function Romulus-H. All members of the Romulus cipher suite internally use the tweakable block cipher SKINNY [BJK+16].

The hash function Romulus-H follows the MDPH construction [Nai19]. This construction combines the Hirose double block-length (DBL) mode [Hir06], a construction to build a $2n$-bit compression function from an n-bit block cipher, with MDP [HPY07], an extension of the Merkle-Damgård construction that prevents length extension attacks. MDPH is proved

to be indifferentiable from a random oracle up to $n - \log(n)$ bits when instantiated with an n-bit ideal block cipher [Nai19, GIM22]. Despite its status as a finalist, no dedicated cryptanalysis of the Romulus-H hash function has been published so far.

Related work. Much of the published corpus on hash function cryptanalysis either focuses on the MD5/SHA-1/SHA-2 family or on the candidates of the SHA-3 competition, particularly the winner Keccak. Many of the techniques introduced in these contexts are hard to apply directly for collision search in the Hirose DBL construction. For example, the analysis of MD5 [WLF+05, WY05] and its successors SHA-1 [WYY05, SKP16, SBK+17] and SHA-2 [MNS11, MNS13, DEM15] strongly builds on the strategy of message modification [WLF+05, WY05]. These attacks consider differential characteristics with a dense, low-probability part that can be controlled and thus satisfied directly from the message input, while the later rounds have sparser differences with higher probability that can be satisfied mostly probabilistically. Inside-out approaches like the rebound attack [MRST09, LMS+15] similarly find many solutions for part of the characteristic, and some of these probabilistically satisfy the rest.

This is much more challenging in a double-block-length construction where such dense parts would typically appear twice, in both primitive calls, so the separation into controllable dense parts and probabilistic sparse parts does not work so well – at least not when looking for hash collisions, where the output and input chaining variable are further constrained. Besides such collisions of the full hash function, we may also look for collisions of the compression function applied to each block in a Merkle-Damgård hash function. Such compression function collisions are referred to as semi-free-start collisions (if the difference is induced only via the message block, while the input and output chaining value each have zero difference) and free-start collisions (if differences may appear in both the message block and the input chaining value). However, they cannot be trivially extended to attacks on the full hash function.

There are a few results on the Hirose double block length construction instantiated with other block ciphers. At FSE 2012, Wei et al. [WPS+12] showed that the IDEA block cipher does not lead to a secure construction when used with Hirose DBL. They search for free-start collisions by using differential characteristics with a difference in the chaining value equal to the initial difference between the two block cipher calls in the Hirose construction. This way, the analysis is very similar to analysis of the Davies-Meyer construction. However, free-start collisions cannot be easily extended to full collisions.

A similar approach has been applied to Hirose DBL instantiated with AES-256. Chen et al. [CHKM14] propose free-start collision attacks on Hirose DBL with up to 9 rounds of AES-256 by using a rebound attack [MRST09]. This attack was extended to 10 rounds by using a quantum version of the rebound attack [CKS21].

The SKINNY tweakable block cipher used in Romulus-H has been analyzed in an unkeyed setting in a different context, in the Matyas-Meyer-Oseas (MMO) construction. This hash function construction is single-block-length and thus has an output size of 128 bits for SKINNY; so it only offers 64 bits of security against collisions. Guo et al. describe collisions for 15 rounds of SKINNY-128-256-MMO in $2^{55.8}$ time and 19 rounds of SKINNY-128-384-MMO in 2^{35} time [GLLP22].

While the MDPH construction is provably secure when instantiated with an ideal block cipher, SKINNY only claims security in the much weaker related-key model. Therefore, we need dedicated analysis of the security of using SKINNY in the MDPH construction. Unfortunately, so far, no such dedicated analysis has been published.

Our contributions. We fill this gap in the analysis of Romulus-H with the first dedicated third-party cryptanalysis. Our contribution can be summarized as follows.

- We create a framework for analyzing hash functions based on Hirose DBL and the Merkle-Damgård construction. We propose so-called *joint differential characteristics*. These capture the relationship between the two block cipher calls in the Hirose DBL construction by using an additional connecting characteristic.
- To identify these joint differential characteristics, we propose a two-step search process based on MILP and CP models. Furthermore, we show efficient CP models for finding collisions based on joint differential characteristics.
- We apply this framework to Romulus-H hash function and provide optimizations based on the STK framework as well as on the slow diffusion of SKINNY.
- With these models, we obtain practical collisions for round-reduced variants of Romulus-H. Concretely, we show practical collisions for 10 out of 40 rounds of Romulus-H and practical semi-free-start collisions for 14 rounds.

Our results are summarized in Table 1. The first table rows show our bounds on the minimum number of active S-boxes (#S) in joint characteristics in general and in a simplified, equality-based model. Next, we show the estimated probability of the best characteristics we found. These estimations are under the Markov assumption, which is clearly not satisfied in the unkeyed setting. The bold values are optimal when restricted to a fixed cellwise characteristic with the minimum number of active S-boxes; we observe a large gap between an activity-based bound and actual characteristics, i.e., many S-boxes use suboptimal transitions. Yet, our results show that our tool-based approach can make efficient use of the available degrees of freedom and find solutions for characteristics with very low probability.

Table 1: Results and bounds on the number of active S-boxes based on different models.

Rounds	6	7	8	9	10	11	12	13	14	15	16
#S (joint)	11	16	25	33	41	46	54	59	69	73	74
#S (equal)	11	16	25	33	42	50	59	67	76	77	96
Best prob.	$2^{\mathbf{-28}}$	—	$2^{\mathbf{-95.2}}$	$2^{\mathbf{-163.3}}$	$2^{-194.8}$	—	$2^{-302.9}$	—	$2^{-397.5}$	$2^{-349.9}$	$2^{-478.3}$
Semi-coll.	✓	—	✓	✓	✓	—	✓	—	✓	—	—
Collision	✓	—	✓	✓	✓	—	—	—	—	—	—

Outline. In Section 2, we recall the specification of Romulus-H and SKINNY, as well as background on automated differential cryptanalysis. In Section 3, we propose our framework for collision-finding in the Hirose DBL construction and propose the concept and modelling of joint characteristics. In Section 4, we apply the framework to Romulus-H and present some cipher-specific optimizations as well as our results. We conclude in Section 5.

2 Background

In this chapter we outline necessary preliminary knowledge including the definition of Romulus-H, and the SKINNY block cipher. Furthermore, we recall differential cryptanalysis, how it can be automated and how it can be applied to hash functions.

2.1 The Romulus-H hash function

Romulus-H is a hash function proposed by Iwata et al. at FSE 2020 [IKMP20]. Together with 3 authenticated encryption modes for different use-cases, Romulus-H forms the Romulus

suite. This suite of cryptographic algorithms is currently a finalist of the NIST Lightweight Cryptography competition.

Romulus-H is based on the Merkle-Damgård construction with permutation (MDP) as depicted in Figure 1. Compared to the original Merkle-Damgård construction, this construction applies a permutation before processing the final message block to prevent length extension attacks. Romulus-H uses an XOR with the value 2 as the permutation. The combination of MDP with Hirose DBL is called MDPH [Nai19].

To build a compression function h, SKINNY-128-384+ is used in the Hirose DBL (double block length) construction as shown in Figure 2. This compression function takes a 256-bit chaining value h and a 256-bit message block M_i as input and outputs a new 256 bit chaining value $h' = \mathrm{CF}(h, M_i)$. The output is calculated as follows. First, the chaining value is split into 2 separate 128-bit values: $h_\mathrm{L} \parallel h_\mathrm{R} = h$. Then, h' is calculated as follows:

$$\begin{aligned} h'_\mathrm{L} &= E(h_\mathrm{L}, h_\mathrm{R} \parallel M_i) \oplus h_\mathrm{L} \\ h'_\mathrm{R} &= E(h_\mathrm{L} \oplus 1, h_\mathrm{R} \parallel M_i) \oplus h_\mathrm{L} \oplus 1 \\ h' &= h'_\mathrm{L} \parallel h'_\mathrm{R}\,, \end{aligned}$$

where $E(M_i, TK)$ denotes encryption of the plaintext M_i under tweakey TK.

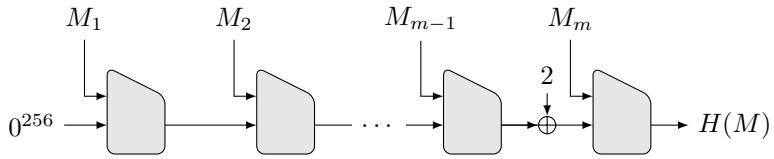


Figure 1: The Merkle-Damgård construction with permutation (MDP) used in Romulus-H.

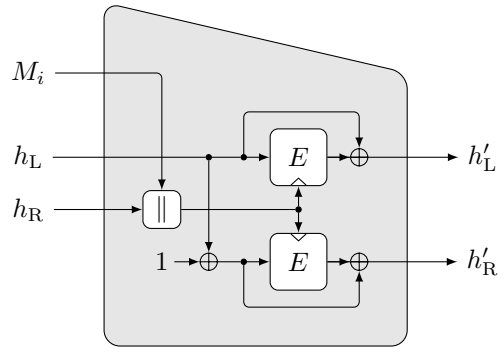


Figure 2: The compression function of Romulus-H based on Hirose's DBL construction.

2.2 The SKINNY family of block ciphers

SKINNY is a family of tweakable block ciphers proposed by Beierle et al. [BJK+16]. It features a 64-bit or 128-bit state and a tweakey of size 128, 256, or 384 bits. Romulus-H uses a state size of 128 bits and a tweakey of 384 bits. The state is organized as a 4×4 matrix of 8-bit elements. The tweakey is stored in 3 4×4 matrices of 8-bit elements called $TK1$, $TK2$, and $TK3$. However, instead of SKINNY-128-384, Romulus-H uses the round-reduced variant SKINNY-128-384+ with 40 instead of 56 rounds.

SKINNY's round function is depicted in Figure 3 and consists of the following steps.

SubCells: An 8-bit S-box is applied to all 16 cells of the state in parallel. This S-box was chosen to meet certain minimum security requirements while minimizing the

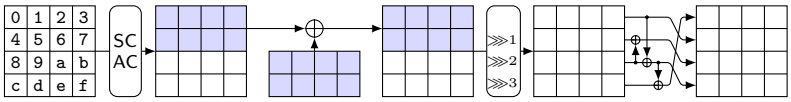


Figure 3: SKINNY's round function.

complexity of a hardware implementation. Consequently, the DDT (differential distribution table) contains entries with probabilities ranging from 2^{-2} to 2^{-7}. Also, it is highly structured as evident from Figure 4. In this figure, each pixel corresponds to an entry in the DDT with darker pixels corresponding to higher probability.

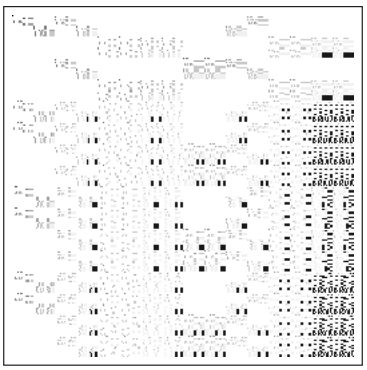


Figure 4: Differential distribution table of the SKINNY S-box. White pixels indicate impossible transitions while darker pixels indicate transitions of higher probability.

AddConstants: A round constant is XORed onto the leftmost column of the state.

AddRoundTweakey: The top two rows of the three tweakey states $TK1$, $TK2$, and $TK3$ are XORed onto the state.

UpdateRoundTweakey: The cells of $TK1$, $TK2$, and $TK3$ are permuted according to

$$(0, \ldots, 15) \rightarrow (9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7)$$

where the indices are taken row-wise. This ensures that the upper half of the initial three tweakey states apply to the even-numbered rounds, while the lower half apply to the odd-numbered rounds. Finally, the top two rows of $TK2$ and $TK3$ are updated with two separate LFSRs of period 15. Note that the LFSRs are exact inverses of each other.

ShiftRows: The cells of the state are rotated row-wise similar to AES but to the right.

MixColumns: Each column of the state is multiplied with the following binary matrix:

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

This mixing layer is chosen to be lightweight and can be implemented with only three XORs and some rewiring.

2.3 Differential Cryptanalysis

Differential cryptanalysis is one the main techniques for evaluating the security of cryptographic primitives. It was proposed by Biham and Shamir in 1990 to attack the DES block cipher [BS90]. The core idea is to find differential characteristics which trace the difference between two executions of a cryptographic primitive and hold with a particular probability. In SPN designs, this probability is determined by the active S-boxes, i.e., S-boxes with nonzero input difference.

In the context of hash function cryptanalysis, differential characteristics are particularly useful for finding collisions; i.e., we are interested in vanishing characteristics with output difference 0. Besides the classic XOR-differences, other difference notions such as modular differences (subtraction modulo 2^n) and signed differences [WY05, WLF+05] are useful. Additionally, different levels of determination of differences can be used to describe different phases of the search: Similar to truncated differential analysis [Knu94], we may consider parts of the characteristic undetermined, or even refine the notion and impose additional constraints on the values [DR06].

Automating Differential Cryptanalysis. The process of finding differential characteristics has increasingly been automated. One common tool is Mixed-Integer Linear Programming (MILP), a modelling approach based on linear constraints and linear objectives, which is often used to find patterns of active S-boxes[MWGP11, ENP19, ZHWW20, MR22]. Additionally, SMT and SAT solvers are often used to find differential characteristics based on these S-box patterns [AK18, MAS15]. In the context of hash function cryptanalysis, automation is mostly driven by custom dedicated tools, particularly when searching for actual collisions [MNS11, SBK+17]. For preimage attacks, a few results using SAT solvers have been shown [HMRS12].

3 Framework for Finding Collisions in the Hirose DBL

In this section, we outline our general framework for finding collisions in hash functions based on MDPH. This framework is built on what we call joint differential characteristics and performs the following 4 steps.

1. Find a joint cellwise characteristic with few active S-boxes.
2. Find many joint bitwise characteristics based on the cellwise characteristic.
3. Filter the bitwise characteristics by searching for an assignment in the semi-free-start collision setting. A semi-free start collision is a choice of chaining value h and distinct messages M_1, M_2 such that $\mathrm{CF}(h, M_1) = \mathrm{CF}(h, M_2)$.
4. Find a full collision by generating many prefixes until a collision can be found. A full collision is a choice of two distinct messages M^1, M^2 such that $H(M^1) = H(M^2)$

3.1 Joint Differential Characteristics

To apply differential analysis to the Hirose DBL construction we define the concept of joint differential characteristics. We define this concept based on a few observations. We can use the same differential characteristic for both invocations of SKINNY. There is a connection between the two SKINNY calls within a single compression function which we describe using the *connecting difference*. In the first few rounds, this connecting difference is mostly zero because the initial difference affects only a single bit. We can keep track of this connecting difference by using a *connecting differential characteristic*. After some time, it does not pay to keep track of the connection difference. Then, we can leave this

difference unknown as done in truncated differential cryptanalysis. Therefore, we define a *joint characteristics* as a (main) differential characteristic δ for the tweakable SKINNY block cipher and a *connecting characteristic* τ, i.e., a truncated differential characteristic with zero difference in the tweakey as both SKINNY calls within a compression function use the same tweakey. We depict the idea and notation based on the example of a single S-box in Figure 5.

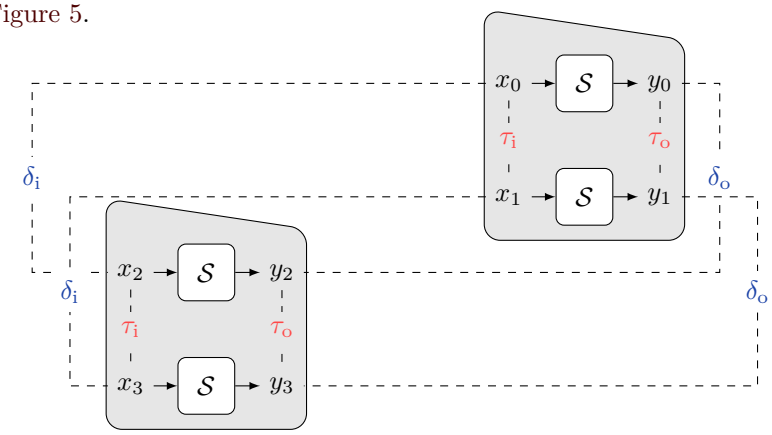


Figure 5: The idea of joint differential characteristics illustrated on a single S-box.

For each S-box transition $(\delta_i, \tau_i) \to (\delta_o, \tau_o)$ in a block cipher with s-bit S-boxes, we have the following: $(\delta_i, \delta_o) \in \mathbb{F}_2^s$, $(\tau_i, \tau_o) \in \mathbb{F}_2^s \cup \{?\}$, and $\tau_i = ? \implies \tau_o = ?$.

Joint probability. We calculate the overall probability of a joint characteristic as the product of the probabilities of each S-box transition. To accurately evaluate the joint transition $(\delta_i, \tau_i) \to (\delta_o, \tau_o)$ through an S-box, we cannot rely only on the DDT, but need additional information. For this purpose, we define two additional tables for the S-box $\mathcal{S}$, the DDT3 and DDT4, depending on whether we want to know τ_o or not:

$$\begin{aligned}\mathrm{DDT3}[\delta_i, \tau_i \delta_o] &= \#\{x \in \mathbb{F}_2^s : \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \delta_i) = \delta_o \wedge \mathcal{S}(x \oplus \tau_i) \oplus \mathcal{S}(x \oplus \tau_i \oplus \delta_i) = \delta_o\} \\ \mathrm{DDT4}[\delta_i, \tau_i, \delta_o, \tau_o] &= \#\{x \in \mathbb{F}_2^s : \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \delta_i) = \delta_o \wedge \mathcal{S}(x \oplus \tau_i) \oplus \mathcal{S}(x \oplus \tau_i \oplus \delta_i) = \delta_o \wedge \\ &\qquad \mathcal{S}(x) \oplus \mathcal{S}(x \oplus \tau_i) = \tau_o \wedge \mathcal{S}(x \oplus \delta_i) \oplus \mathcal{S}(x \oplus \delta_i \oplus \tau_i) = \tau_o\}\,.\end{aligned}$$

Additionally, we use XDDT, XDDT3, and XDDT4 to denote the underlying solution sets, i.e., the sets of input values to the S-box (x in the equation above) that lead to the desired transition. Then, the transition probability p of $(\delta_i, \tau_i) \to (\delta_o, \tau_o)$ for an s-bit S-box is

$$p = \begin{cases} (\mathrm{DDT}[\delta_i, \delta_o] \cdot 2^{-s})^2 & \text{if } \tau_i = ? \\ \mathrm{DDT}[\delta_i, \delta_o] \cdot 2^{-s} & \text{if } \tau_i = 0 \text{ and } \tau_o = 0 \\ \mathrm{DDT3}[\delta_i, \tau_i, \delta_o] \cdot 2^{-s} & \text{if } \tau_i \in \mathbb{F}_2^s \text{ and } \tau_o = ? \\ \mathrm{DDT4}[\delta_i, \tau_i, \delta_o, \tau_o] \cdot 2^{-s} & \text{otherwise.} \end{cases} \tag{1}$$

Maximum Probability of the DDT3 and DDT4. The largest entries in the DDT3 and DDT4 (for nonzero τ_i) equal the differential uniformity of the S-box. This is a special case of the following fact:

$$\begin{aligned}\mathrm{DDT4}[\delta_i, \delta_i, \delta_o, \delta_o] &= \mathrm{DDT}[\delta_i, \delta_o] \\ \mathrm{DDT3}[\delta_i, \delta_i, \delta_o] &= \mathrm{DDT}[\delta_i, \delta_o]\,.\end{aligned}$$

While the DDT4 of an 8-bit S-box is a table with 2^{32} entries, it is very sparse. For example, the DDT4 of the 8-bit SKINNY S-box only contains $2^{16.8}$ nonzero entries. Therefore, we can store it efficiently as a sparse matrix. The DDT3 is also relatively sparse; for SKINNY there are $2^{16.2}$ nonzero entries (out of 2^{24}).

3.2 MILP Model for Joint Cellwise Characteristics

To find joint characteristics and conforming inputs, we follow a multi-step approach. One of the main challenges is the complexity of the DDT4, which is challenging for automated solvers. For this reason, we want to minimize the number of S-boxes we need to model with the full DDT4, and first search for cellwise characteristics; that is, we first only determine the activity pattern of the joint characteristics, not the precise bit-level differences. This way, we obtain (a) upper bounds on the probability of the best joint characteristics and (b) starting points to search for good (though not necessarily optimal) bit-level differential characteristics.

We define a MILP model to search for optimal or near-optimal joint cellwise characteristics and associated bounds. To calculate cellwise bounds, we consider the maximum differential probability given by these tables. We consider three different settings to evaluate the advantage of joint characteristics:

- **Plain setting**: We ignore any connection between the upper and lower characteristic (i.e., $\bar{\tau} = ?$ is completely undetermined) and consider the same characteristic $\bar{\delta}$ independently for both SKINNY calls. For each active S-box $\bar{\delta}_i \neq 0$, we pay (at least) the maximum differential probability of the S-box DDT twice. In other words, we consider only the first case in Equation 1. For the cellwise model, we need only one variable per S-box, as the same optimal characteristic $\bar{\delta}$ can be used for upper and lower part and $\bar{\tau}$ is not needed; for the bitwise model, we only need the DDT.

- **Equality setting**: We propagate the fixed input difference of $\bar{\tau}$ with probability 1, i.e., we deduce in which cells $\bar{\tau}$ is definitely zero. Then, for each active S-box $\bar{\delta}_i \neq 0$, we pay the S-box DDT transition either once (if $\bar{\tau}_i = 0$, second case in Equation 1) or else twice as in the independent setting. For the cellwise model, we use two variables per S-box for $\bar{\delta}$ and $\bar{\tau}$, but only the $\bar{\delta}$ variables are actual decision variables while $\bar{\tau}$ can be deduced deterministically. For the bitwise model, we still only need the DDT.

- **Joint setting**: We use the full definition of joint characteristics from Subsection 3.1, with the transition probabilities of Equation 1. The connection characteristic $\bar{\tau}$ is modelled as a *partial characteristic*: Differences can be either *known* ($\tau \in \mathbb{F}_2^s$ for bitwise characteristics or $\bar{\tau} \in \{0, \mathsf{x}\}$ for cellwise characteristics, where x denotes an active, nonzero difference) or *unknown* ($\bar{\tau} = ?$). In an S-box with a known, nonzero input difference $\bar{\tau}_i \neq 0$, we can choose to either know or *forget* the output difference. For the cellwise model, we need corresponding decision variables to these states; for the bitwise model, we need the DDT, DDT3, and DDT4.

MILP model for the plain setting. The objective of this model is to minimize the number of active S-boxes in the main characteristic $\bar{\delta}$ in order to maximize the differential probability in the two "independent" SKINNY calls. We introduce the following binary decision variables for R-round SKINNY-128-384: $X[r, i, j]$ is the S-box activity of $\bar{\delta}$ in round $r \in \mathcal{R}' = \{0, 1, \ldots, R\}$ (where $X[R, \cdot, \cdot]$ is the output), row $i \in \mathcal{I} = \{0, 1, 2, 3\}$, and column $j \in \mathcal{J} = \{0, 1, 2, 3\}$; $Y[r, i, j]$ is the activity pattern of $\bar{\delta}$ after AddRoundTweakey for $r \in \mathcal{R} = \{0, 1, \ldots, R-1\}$; and $K[r, i, j]$ is the activity pattern of the round subtweakey. This round subtweakey is the XOR of the 3 round tweakeys $TK1[r]$, $TK2[r]$, $TK3[r]$; to correctly model potential cancellations in this addition that are consistent with their LFSR update functions, we follow the designers' MILP model [BJK+16]: we use variables $L[i, j]$ to denote the lane activity. If $L[i, j] = 0$, then $K[0, i, j]$ and all corresponding permuted versions of this cell for $r \geq 1$ must be 0. If $L[i, j] = 1$, then at least $r - 1$ of the r corresponding cells must be active, as cancellations can occur only once per 30 rounds (in the TK2 setting).

Based on these variables, a cellwise differential model of the round update function is quite straightforward using only 2-input cellwise XORs. We model all XORs without helper variables to minimize the solving time [ENP19].

Since we want to find collisions, we additionally constrain the input and output differences accordingly. In the DBL construction, both SKINNY instances are used with a feed-forward. Thus, we obtain a free-start collision of the compression function if the input and output difference match, i.e., $X[0,i,j] = X[R,i,j]$ for all $i \in \mathcal{I}, j \in \mathcal{J}$. In this paper, we focus on semi-free-start collisions and full hash collisions, so we require a zero input difference in the chaining value, $X[0,i,j] = 0$ for all i,j. Additionally, we are limited to the TK2 setting, as only the 2 tweakey input states corresponding to the message block M can induce differences (see Figure 2).

MILP model for the equality setting and joint setting. For both settings, the objective is to minimize the joint S-box transition cost and thus maximize the joint differential probability. We introduce additional binary decision variables for the connecting characteristic $\bar{\tau}$: $T[r,i,j]$ is the S-box activity of $\bar{\tau}$ in round r; $U[r,i,j]$ denotes whether this activity is known ($U=0$) or unknown ($U=1$) at the S-box output; and $F[r,i,j]$ marks whether a known, active input difference $\bar{\tau}_\mathrm{i} = \mathsf{x}$ to an S-box is forgotten, i.e., $\bar{\tau}_\mathrm{o} = \mathsf{?}$. Thus, $U[r,i,j]$ denotes whether the S-box output is unknown, while $U[r,i,j] - F[r,i,j]$ denotes whether the input is unknown.

The cost $C[r,i,j] \in \{0,1,2\}$ of an S-box transition depends on all these decision variables and is measured in multiples of the differential weight of the best non-zero DDT transition. It is nonzero whenever the main characteristic is active, $\bar{\delta}_\mathrm{i} \neq 0$, or the connecting characteristic is active with known output difference, $\bar{\tau}_\mathrm{i} = \bar{\tau}_\mathrm{o} = \mathsf{x}$ (see Equation 1). It reaches the maximum cost of 2 when the main characteristic is active, $\bar{\delta}_\mathrm{i} \neq 0$, and the connecting characteristic has an unknown input difference, $\bar{\tau}_\mathrm{i} = \mathsf{?}$. This can be expressed in linear constraints as

$$\begin{aligned} \forall i \in \mathcal{I}, j \in \mathcal{J}, r \in \mathcal{R}: \quad & C[r,i,j] \geq X[r,i,j] \\ & C[r,i,j] \geq T[r,i,j] - U[r,i,j] \\ & C[r,i,j] \geq 2 \cdot X[r,i,j] + U[r,i,j] - F[r,i,j] - 1\,. \end{aligned}$$

For the joint setting, we initialize

$$\forall\, i \in \mathcal{I}, j \in \mathcal{J}: \qquad U[0,i,j] = F[0,i,j], \qquad T[0,i,j] = \begin{cases} 1 & i = j = 0, \\ 0 & \text{else.} \end{cases}$$

For simplicity, we only describe the joint setting; the equality setting is functionally equivalent to the joint setting with $U[0,0,0] = 1$, i.e., considering the initial fixed difference of 1 as unknown, which will propagate to all following (potentially) active cells in $\bar{\tau}$. Similarly, the model is reduced to the plain setting if we set $U[0,i,j] = 1$ for all i,j.

We can model the effect of the round function for $\bar{\tau}$ in a similar way as $\bar{\delta}$ based on 2-input XORs, using a simple TK0 setting as both calls use the same tweakey, so there are no connecting differences in the tweakey schedule. Regarding the known/unknown status, we can choose to forget $\bar{\tau}_\mathrm{o}$ in any S-box with active, known input $\bar{\tau}_\mathrm{i}$:

$$\forall i \in \mathcal{I}, j \in \mathcal{J}, r \in \mathcal{R}: \qquad F[r,i,j] \leq T[r,i,j]$$

For the XOR operations, if either of the inputs is unknown, then the output is unknown; else, it is known. For example, if the S-box input in cell (i,j) of round $r+1$ is derived as the XOR of S-box output cells (i',j') and (i'',j'') in round r (via ShiftRows and MixColumns),

then we add

$$\begin{aligned} U[r, i', j'] &\leq U[r+1, i, j] - F[r+1, i, j] \\ U[r, i'', j''] &\leq U[r+1, i, j] - F[r+1, i, j] \\ U[r, i', j'] + U[r, i'', j''] &\geq U[r+1, i, j] - F[r+1, i, j]\,. \end{aligned}$$

We discuss the bounds obtained with this model in Subsection 4.1 and use the best cellwise characteristics as starting points for collision-finding.

3.3 SMT Model for Finding Joint Bitwise Characteristics

To find a joint bitwise characteristic, we use the Z3 SMT solver [dMB08]. Concretely, we create variables for all active S-box input and output differences in the main characteristic and variables for all active and known differences in the connecting characteristic.

To model the linear layer, we use XOR constraints of the SMT solver. We only need to model those differences that are active in the truncated joint characteristic. Note that we only model the connecting differential characteristic as long as there is a known difference.

To model the S-boxes, we consider the relevant DDTs. For a given S-box, we distinguish based on the values of the activity indicators of the joint cellwise characteristic, i.e., $\bar{\delta}$, $\bar{\tau}_\mathrm{i}$, $\bar{\tau}_\mathrm{o}$. If both characteristics are inactive, i.e., $\bar{\delta} = \bar{\tau}_\mathrm{i} = \bar{\tau}_\mathrm{o} = 0$, we do not need to model anything. If only the main characteristic is active and the connecting difference is inactive or unknown, i.e., $\bar{\delta} = \mathsf{x}$ and $\bar{\tau}_\mathrm{i} \in \{0, ?\}$, then we need to model the DDT. To do this, we list all possible transitions $\{\delta_\mathrm{i}, \delta_\mathrm{o} : \mathrm{DDT}(\delta_\mathrm{i}, \delta_\mathrm{o}) > 0\}$ in a DNF. We then convert this DNF to a small CNF using the `espresso` logic minimizer and pass it to the Z3 solver. If only the connecting difference is active and known, i.e., $\bar{\tau}_\mathrm{i} = \bar{\tau}_\mathrm{o} = \mathsf{x}$ and $\bar{\delta} = 0$, we use an analogous model. The remaining cases are handled similarly. For the case of the connecting difference turning unknown, i.e., $\bar{\tau}_\mathrm{i} = \mathsf{x}$, $\bar{\tau}_\mathrm{o} = ?$, we model the DDT3. Finally, if all differences are nonzero and known, i.e., $\bar{\delta} = \bar{\tau}_\mathrm{i} = \bar{\tau}_\mathrm{o} = ?$, we model the DDT4.

While the model is functional, we perform a small modification to improve efficiency and to find better characteristic. To do this, we define a cutoff value w_c, and only consider the transitions $\{\delta_\mathrm{i}, \delta_\mathrm{o} : \mathrm{DDT}(\delta_\mathrm{i}, \delta_\mathrm{o}) \geq w_\mathrm{c}\}$ in a DNF. We do this analogously for the DDT3 and DDT4 with cutoff values $w_{\mathrm{c,DDT3}}$, and $w_{\mathrm{c,DDT4}}$, respectively. This serves two purposes. First of all, the resulting model will be simpler as fewer transitions need to be considered. And second, the resulting differences will have higher probability as all transitions worse than the cutoff are left out.

We want to use the characteristics this model generates to find semi-free-start collision and ultimately full collisions. To do this, we search for assignments of the characteristic with arbitrary or fixed chaining value hf for semi-free-start collisions or full collisions, respectively. One factor that influences whether such an assignment exists is the probability and the degrees of freedom in the given setting. In the semi-free-start setting, we can freely the message M_1 and the chaining value h. While when searching for full collisions, we only freely control the message M_1; the chaining value h is determined pseudorandomly based on the preceding message blocks. As an attacker, this chaining value h can thus only be influenced by generating many different preceding message blocks. We can only expect to find an assignment (i.e., a (semi-free-start) collision) if the degrees of freedom exceed the probability of differential characteristic.

When applying this model to Romulus-H in Section 4, we find that this is not a sufficient condition. We would expect a probability of $\geq 2^{-512}$ to be sufficient to find semi-free-start collisions and a probability of $\geq c \cdot 2^{-256}$ to be sufficient to find full collisions within c trials. However, we identify many characteristics with sufficient probability where no valid assignment exists. This is because we estimate the probability under the assumption of a Markov cipher, i.e., under the assumption that each round is totally independent of the previous round. This is not the case for most block ciphers and definitely not the case for

SKINNY. First of all, each round of SKINNY only mixes 64 bits of tweakey material into the 128 bit state. Furthermore, the key schedule is relatively simple as it only consists of cellwise permutations and LFSRs. Therefore, in our application to Romulus-H this only serves as a necessary condition.

3.4 SMT Model for Finding Collisions

So far, we have found a cellwise joint characteristic and a compatible bitwise joint characteristic. Now we want to use the bitwise joint characteristic to find a semi-free-start collision, i.e., an assignment for h and M such that $\mathrm{CF}(h, M) = \mathrm{CF}(h, M \oplus \Delta M)$.

A naive way to find such an assignment would be to model the whole compression function twice and restrict the difference between them based on the main differential characteristic and the difference within each compression function based on the connecting differential characteristic. Such a model is easy to construct; however, it creates a lot of unnecessary variables and constraints. For example, many variables must be equal due to a zero difference in the joint characteristic. Furthermore, we can even use the fixed nonzero differences in the characteristic to eliminate variables.

Instead of this naive model, we only define variables for the execution of $\mathrm{CF}(h, M)$ and add conditions such that the differential characteristic is followed as done in previous work [MZ06]. We optimize our model based on the main differential characteristic and the connecting characteristic. We outline these optimizations in the paragraphs below.

Optimizations based on main differential characteristic. We model each S-box based on the XDDT. Concretely, we model all possible values (x, y) such that $y = \mathcal{S}(x)$ and $x \in \mathrm{XDDT}[\delta_\mathrm{i}, \delta_\mathrm{o}]$. Similar to the model for finding characteristics, we use the espresso logic minimizer to generate a small CNF. Note that we also apply this model to the inactive S-boxes, where $\mathrm{XDDT}[0, 0]$ corresponds to the whole input space. By using this model, all the S-box transitions are guaranteed to happen.

For the linear layer, we only need to ensure that the values we choose for the S-boxes are also consistent with the linear layer. As the linear layer transitions happen with probability 1, we don't need any extra constraints. To build the model, we use XOR constraints of the SMT solver. Note that we also need to model the round constants.

This model is already functional and fast. However, we can further improve it by considering the connecting difference.

Further optimizations based on connecting characteristic. So far, we have seen that we only need to model one compression function with some extra constraints on the S-boxes. Now, we explain how we can use the connecting difference to reduce the number of S-boxes and linear layers we need to model this one compression function.

Consider an S-box transition with differences $\delta_\mathrm{i} \to \delta_\mathrm{o}$ and connecting differences $\tau_\mathrm{i} \to \tau_\mathrm{o}$. If both connecting differences are known, i.e., $\tau_\mathrm{i} \neq ?$ and $\tau_\mathrm{o} \neq ?$, then we only need to model a single S-box assignment instead of two. Concretely, we model (x, y) such that $y = \mathcal{S}(x) \land x \in \mathrm{XDDT4}[\delta_\mathrm{i}, \tau_\mathrm{i}\delta_\mathrm{o}, \tau_\mathrm{o}]$. As before, we use the `espresso` logic minimizer to create an optimized CNF for this assignment. Note that if $\tau_\mathrm{i} = \tau_\mathrm{o} = 0$, then the XDDT4 equals $\mathrm{XDDT}(\delta_\mathrm{i}, \delta_\mathrm{o})$ and we get the same model as before, but we only need to create it once instead of twice.

If only one of the connecting differences is known, i.e., $\tau_\mathrm{i} \neq ?$ and $\tau_\mathrm{o} = ?$, then we create two models for the two S-box assignments (x, y_0) such that $y_0 = \mathcal{S}(x)$ and $x \in \mathrm{XDDT}(\delta_\mathrm{i}, \delta_\mathrm{o})$ and (x, y_1) such that $y_1 = \mathcal{S}(x \oplus \tau_\mathrm{i})$ and $(x \oplus \tau_\mathrm{i}) \in \mathrm{XDDT}(\delta_\mathrm{i}, \delta_\mathrm{o})$. These are modeled as two separate minified CNFs.

If both connecting differences are unknown, i.e., $\tau_\mathrm{i} = ?$ and $\tau_\mathrm{o} = ?$, then we need to use the model outlined above. That is, we model the two separate S-box assignments (x_0, y_0)

and (x_1, y_1) according to the XDDT.

To model the linear layer, we can use similar observations. In particular, we only need to include the linear layer for those values that we actually use in the S-box model above. Concretely, if the connecting difference τ at an output of the linear layer is known ($\tau \neq ?$), then we only need to model this calculation once. Otherwise, we need to model it twice.

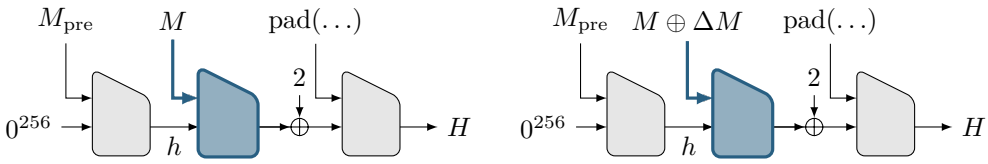
3.5 Extension to Full Collisions

While (semi-)free-start collisions show potential weaknesses in a hash function, we ultimately want to find full collisions. That is, we want to find two messages M^1 and M^2 such that $H(M^1) = H(M^2)$ and $M^1 \neq M^2$.

For our search, this means that the input chaining value is constrained, either to 0 (if we target the first message block) or to the chaining value produced by a fixed prefix. We still want to take advantage of the (limited) degrees of freedom in the chaining value, so we repeatedly pick a random prefix M_{pre}. This prefix leads to a fixed chaining value $h = \text{CF}(0^{256}, M_{\text{pre}})$ as defined by the MDPH construction. Then, we use this chaining value as a starting point for a collision in the next compression function call. Consequently, we search for messages of the following form:

$$M^1 = M_{\text{pre}} \parallel M\,,$$
$$M^2 = M_{\text{pre}} \parallel (M \oplus \Delta M)\,.$$

To model this setting, we can reuse the SMT model from the previous subsection with only a minor modification. Instead of leaving the chaining value h free, we use the fixed chaining value $h = \text{CF}(0^{256}, M_{\text{pre}})$. This way, the assignment for M will not only be a semi-free-start collision but a full collision for the hash function. The setup for the colliding messages in the analyzed construction is depicted in Figure 6.



(a) Processing of M^1. (b) Processing of M^2.

Figure 6: Processing of the colliding messages in the MDP construction.

We do not expect that this process is successful on the first try. Therefore, we generate many random prefix values M_{pre} until we find an assignment for M. Note that this process can be performed in parallel on many cores.

4 Application to Romulus-H

Now, we apply the framework to find hash collision from the previous section to Romulus-H. We introduce dedicated optimizations based on the STK-framework and SKINNY and discuss the results.

4.1 Results for Joint Cellwise Characteristics

We use the unmodified MILP model to derive upper bounds on the number of active S-boxes in joint cellwise characteristics.

Figure 7: Joint cellwise characteristic for 14 rounds of Romulus-H with 69 active S-boxes.

The results of these models are listed in Table 2. Based on these bound for the number of S-boxes, we can also find a bound on the best joint differential characteristics However, we do not expect these bounds to be very tight because the concrete transitions for each S-box have a big range of probability from 2^{-2} to 2^{-7}. For a valid bound, we need to assume all S-box transitions use the best case of 2^{-2} which will not be the case in practice.

Based on these results, we see that the idea of joint differential characteristics gains importance as the number of S-boxes grows. In particular, for 16 rounds, only 74 active S-boxes are needed in the joint settings while 96 are needed in the equality setting.

Table 2: Bounds on the number of active S-boxes based on different models.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Plain	0	0	4	4	4	16	22	34	44	54	60	66	78	86	86	106
Equal	0	0	2	2	2	11	16	25	33	42	50	59	67	76	77	96
Joint	0	0	2	2	2	11	16	25	33	41	46	54	59	69	73	74

Figure 7 shows a joint characteristic in the joint setting with a cost of 69 active S-boxes.

4.2 Finding Joint Bitwise Characteristics for Romulus-H

Next, we use the cellwise characteristics from the previous step to find bitwise characteristics for Romulus-H. For this purpose, we introduce an optimized model for the tweakey schedule which applies to all STK constructions. Furthermore, we discuss different trade-offs and approaches for obtaining minimized CNFs of the large DDT4 tables.

4.2.1 Optimized model for the tweakey schedule

We want to create a model for all possible round tweakey differences that are compatible with the cellwise joint characteristic. In our setup, we only need to model the $TK2$ and $TK3$ lanes of each active tweakey lane. The $TK1$ lanes always have zero difference, as these are given by the chaining value in the MDPH mode of operation and we are looking for semi-free-start collisions.

We want to model the LFSRs of the tweakey schedule with as few variables and as few XOR constraints as possible. Therefore, we take inspiration by the technique outlined by Soos et al. [SNC09]. They propose to choose the reference bits, i.e., the actual variables the solver solves for, in the middle of the output stream of the LFSR. This way, fewer XOR constraints are needed as the output bits we calculate using XORs are closer to the actual variables of the model.

Additionally, we use the fact that the LFSRs for $TK2$ and $TK3$ are exact inverses of each other and combine that with the known cancellations in the cellwise characteristic in the tweakey schedule. If there is a difference cancellation between $TK2$ and $TK3$ in the tweakey schedule, we know that the two differences in this cell must be equal. Therefore, we can create a single set of 8 variables as a reference state to model the difference for the two active tweakey lanes $TK2$ and $TK3$. Then, based on these 8 variables, we create all the linear expressions that we need to model the $TK2$ and $TK3$ lane. By using these ideas, we only need 8 variables instead of 16 to model one $TK2$ lane and the associated $TK3$ lane.

4.2.2 Optimized CNF encoding of the DDT4

Creating a small representation of the DDT4 in CNF formulation is challenging because it is so large. In particular, just using the espresso logic minimizer does not lead to results, even after multiple days. Instead, we can use the SAT solver Lingeling, which supports CNF minimization with time limits, to minimize the CNF [Bie17].

Concretely, our approach is performed in 4 steps. First, we create a DNF of all entries above or equal to the cutoff value $w_{\mathrm{c,DDT4}}$. Second, we convert this DNF to a CNF by using espresso with the right configuration. We use the `.phase 0` option to swap the set of ones and zeros after reading the input. This way we can specify the much small DNF, and it is internally converted to a CNF of reasonable size. Instead of optimizing this CNF, we use the `-Decho` command line flag to disable all optimization steps and just receive a reasonably sized CNF. Third, we note that a DDT transition above the cutoff is a necessary condition for a DDT4 transition above the cutoff. Hence, we can add a CNF of $\{\delta_{\mathrm{i}}, \delta_{\mathrm{o}}, : \mathrm{DDT}(\delta_{\mathrm{i}}, \delta_{\mathrm{o}}) \geq w_{\mathrm{c,DDT4}}\}$ and $\{\tau_{\mathrm{i}}, \tau_{\mathrm{o}}, : \mathrm{DDT}(\tau_{\mathrm{i}}, \tau_{\mathrm{o}}) \geq w_{\mathrm{c,DDT4}}\}$. These two CNFs allow Lingeling to eliminate many clauses, as many clauses of the DDT4 CNF are already covered by one of the DDT CNFs.

When we apply this model to the formulation of $\{\delta_{\mathrm{i}}, \tau_{\mathrm{i}}, \delta_{\mathrm{o}}, \tau_{\mathrm{o}} : \mathrm{DDT4}(\delta_{\mathrm{i}}, \tau_{\mathrm{i}}, \delta_{\mathrm{o}}, \tau_{\mathrm{o}}) \geq w_{\mathrm{c,DDT4}}\}$ with cutoff $w_{\mathrm{c,DDT4}} = 8$ we see a drastic decrease in the number of required clauses. Instead of listing the nearly 2^{32} entries below the cutoff, we only list the $2^{15.7}$ entries above or equal to the cutoff thanks to the `.phase 0` option. When running espresso with all optimization steps disabled, we receive a CNF with $146\,317 = 2^{17.2}$ clauses within a few seconds. Then, we add 217 clauses for each of the two DDTs. Finally, we run

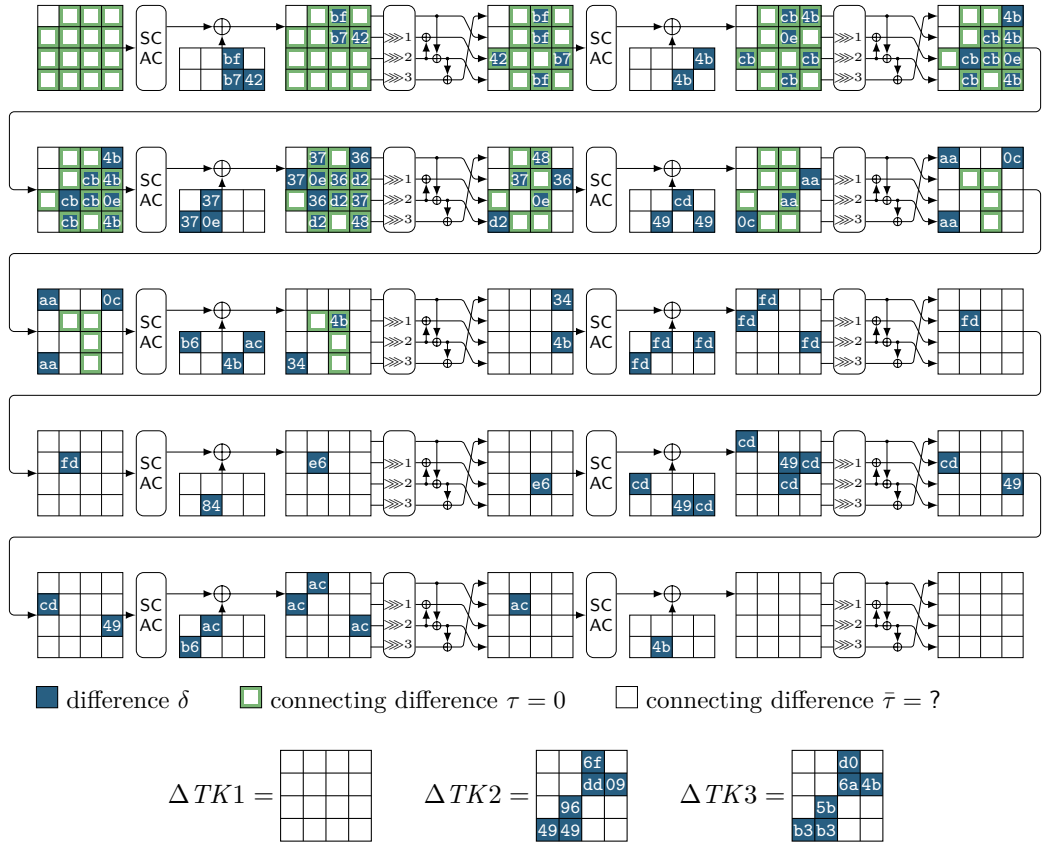


Figure 8: Joint characteristic for 10 rounds of Romulus-H with $p = 2^{-233.8}$.

Lingeling in optimization mode and receive a CNF with only $31\,578 = 2^{14.9}$ clauses after a few hours.

4.2.3 Results for Bitwise Characteristics and Semi-Free-Start Collision

We apply this model to the best cellwise joint characteristic for 10 rounds and find that choosing the right cutoff values is crucial for identifying good characteristics quickly. When using $w_c = 4$, $w_{c,DDT3} = w_{c,DDT4} = 8$, the SMT solver reports that no such characteristics exist after a few seconds. When using $w_c = 4$, $w_{c,DDT3} = w_{c,DDT4} = 4$, we find a new joint characteristic about every 20 seconds on a laptop. The identified joint characteristics have probabilities ranging from 2^{-219} to 2^{-229}.

To speed up the search, we also apply the model to the best cellwise joint characteristic for 10 rounds in the equality model. Then, we do not need to model the heavy DDT4. When using a DDT cutoff of $w_c = 4$, the Z3 solver needs about 250 milliseconds for each joint characteristic. While these have lower probabilities on average, we can generate many more of them within the same time. After generating 355 joint characteristics, we find a satisfiable one with probability $p = 2^{-233.8}$ which is depicted in Figure 8. The green border indicates which S-boxes have a connecting difference that is known to be zero.

We also apply the model to the best cellwise joint characteristic for 14 rounds in the equality model. Because a cutoff of $w_c = 4$ leads to an unsatisfiable model, we use $w_c = 2$. In this configuration, Z3 needs about 600 milliseconds to generate a new joint

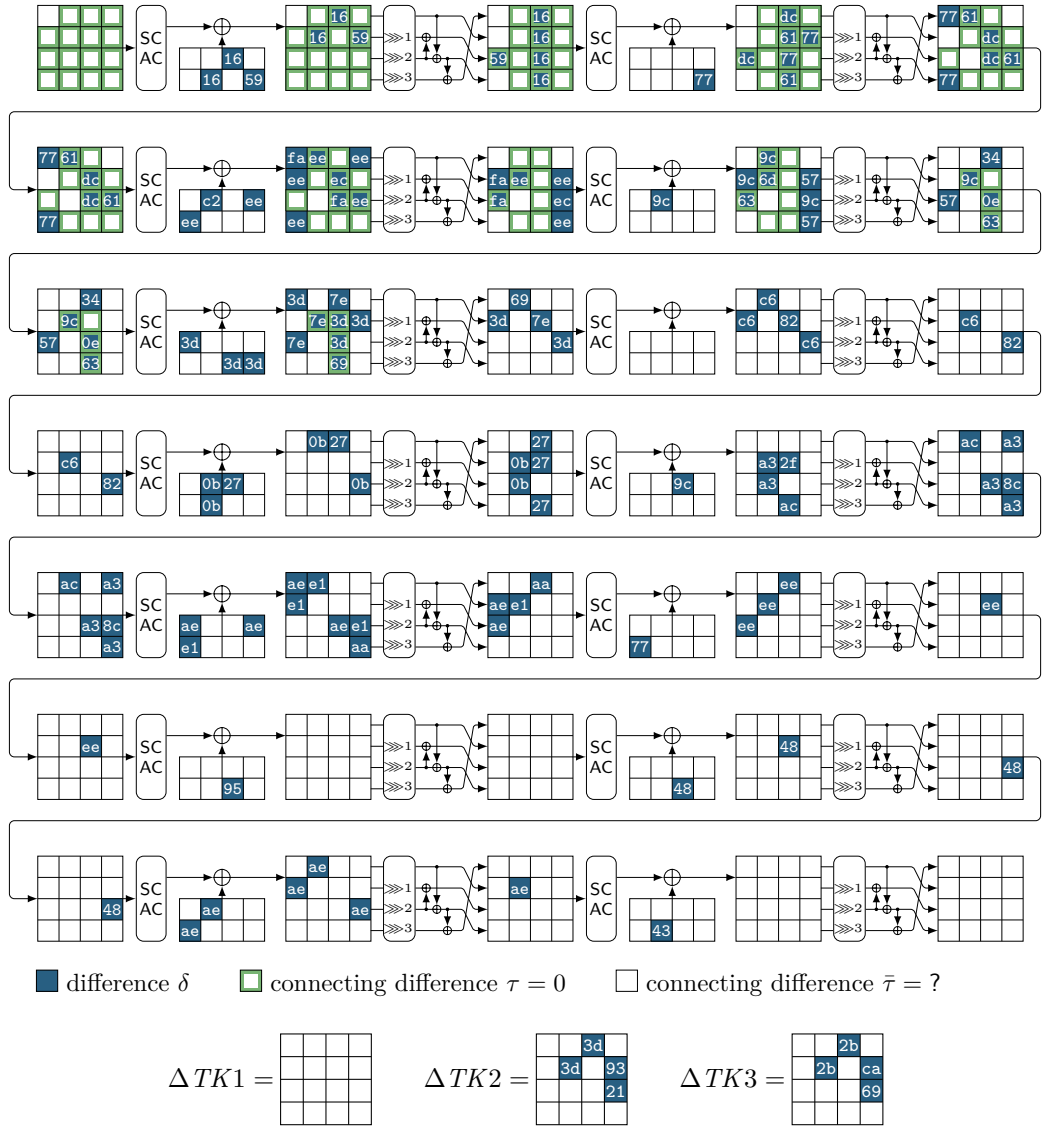


Figure 9: Joint characteristic for 14 rounds of Romulus-H with $p = 2^{-419.66}$.

characteristic. After generating 405 joint characteristics, we find a satisfiable one with probability $p = 2^{-419.66}$ which is depicted in Figure 9. Interestingly, we did not find any additional satisfiable characteristics, even after generating many thousands more.

For this joint characteristic, we can find valid assignments to obtain a semi-free-start collision for 14 rounds of Romulus-H. Concretely, we find a chaining value h and two messages M_1, M_2 such that $\mathrm{CF}(h, M_1) = \mathrm{CF}(h, M_2)$, where CF denotes 14-round

compression function:

$$\begin{aligned} h &= \texttt{4039b0e35651a1ab68979475e7c469b4e961b61a8f4ac9362e36a6cb44eec973}, \\ M_1 &= \texttt{2c367b0d018a8f536cf8ea100401d344734f8e890c5bfc45b2c693b1c19b8959}, \\ M_2 &= \texttt{2c36460d01b78fc06cf8ea310401d344734fa5890c70fc8fb2c693d8c19b8959}, \\ M_1 \oplus M_2 &= \texttt{00003d00003d00930000002100000000000002b00002b00ca0000006900000000}. \end{aligned}$$

Finding this semi-free-start based on the characteristic takes about 60 seconds on a laptop. The main bottleneck which prevents extending this results to more rounds is finding characteristics that are actually satisfiable. We generated thousands of characteristics for 15 rounds; none of them where satisfiable. Based on the consideration of degrees of freedom and probabilities there might be solutions for 15 rounds as well. However, there seem to be many hidden contradictions because SKINNY is not a Markov cipher.

4.3 Bounding Joint Differential Characteristics

We have established that there is a link between satisfiability and the probability of a differential characteristic. This raises the question of whether we can extend our attacks by explicitly searching for characteristics with high probability.

To find the best joint differential characteristics compatible with a given cellwise trail, we use a technique similar to the one in CryptoSMT [Kö]. The main idea is to define an upper limit on the overall log-probability $w_{\max}$ and then find a value x such that the model with $w_{\max} = x$ is satisfiable while the model with $w_{\max} = x - 1$ is not. To build this model, we use the cellwise characteristics obtained from the equality model as a basis. This makes our model easier, as we do not have to consider the large DDT3 and DDT4. Concretely, we split the DDT into sub-tables w-DDT where for each weight $w \in \{0, 2, 3, 4, 5, 6, 7\}$. Each table $w - DDT$ corresponds to those entries in the DDT with associated probability $2^{w-1} < p \leq 2^w$. Then we create a separate CNF for each w-DDT. For each S-box, we then create indicate variables v_w for each value of w and model $v_w \Rightarrow (\delta_\mathrm{i}, \delta_\mathrm{o} \in w\text{-DDT})$. Additionally, we need to make sure that at least one of the indicate variables is true; so we add the following constraint: $\bigvee_w v_w$. With this model we might slightly overestimate the probability of our differential characteristic (a characteristic with probability slightly worse 2^x might be satisfiable with $w_{\max} = x$) because we are grouping transitions with non-power-of-two probability with the next higher power of two. However, this effect is limited as of the 11 469 transitions only 170 have a probability that is not a power of two.

Results. The probabilities of the best characteristics identified with this model are listed in Table 1 on page 3. The probabilities typeset in bold font correspond to tight bounds were either all transitions used a power-of-two entry in the DDT or it has been verified with a more exact model that creates a separate class for each possible transition probability. The other probabilities are the best characteristics identified but not necessarily optimal for the given cellwise characteristic. For 7, 11, and 13 rounds, we could not apply any of the SMT models, because some tweakey lanes contain no cancellation which was not considered in the code.

4.4 Finding Collisions for Romulus-H

While finding semi-free-start collisions can be done as outlined for the general framework, when searching for full collisions we are much more limited by the fixed chaining value h which is determined by the prefix M_{pre}.

To find semi-free-start collisions, we use the approach from Subsection 3.4 unmodified. However, as the number of rounds increases, many of the bitwise joint characteristics are not satisfiable. Therefore, we generate many characteristics in a loop and immediately

check satisfiability in a semi-free start setting. This way, we are only left with valid characteristics that we use to search for full collisions.

When searching for full collisions, we can perform some optimizations based on the unkeyed first round of SKINNY. Remember that each prefix M_{pre} leads to a fixed chaining value h_{L} at the input of SKINNY block cipher. Therefore, we have limited degrees of freedom in the first few rounds. The first SubCells layer is completely tweakey-independent (and thus message-independent). Any active S-box in the connecting differential characteristic will only be valid for a subset of chaining values.

Similarly, we can optimize based on the second round where the tweakey dependency is still low. The second SubCells layer only depends on 64 bits of tweakey material which is only slowly mixed with the state. Remember that tweakey addition is performed on the first and second row only. According to the definition of MixColumns, the second row only influences the third row after mixing, while the third row influences rows 1, 2, and 4. Now, consider the characteristic in Figure 10. The three highlighted S-box transitions in the second round only depend on a single tweakey byte. Consequently, only a small fraction of all h_{L} values permit an assignment of the free tweakey values such that the characteristic is followed in the first two rounds. In the example from Figure 10, this fraction is 2^{-11}. Based on these observations, we can efficiently filter the chaining values. Then, we only need to call the SAT solver for a small fraction of chaining values. This filtering step for chaining values is crucial for finding collisions in reasonable time.

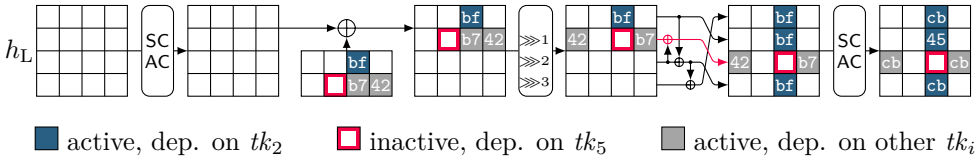


Figure 10: Filtering chaining values h_{L} in the first two rounds for full collisions. The three blue S-box transitions only depend on a single tweakey byte. The other tweakey byte in the same column (red highlight) does not affect the transitions.

Results. Based on the 10-round characteristic identified earlier (Figure 8), we found a full collision for 10 rounds. We needed just under an hour on a CPU with 88 cores to identify the result. Concretely, we found M_{pre}, M_1, and M_2 such that $H(M_{\text{pre}} \,\|\, M_1) = H(M_{\text{pre}} \,\|\, M_2)$, where H denotes Romulus-H based on 10-round SKINNY-128-384:

$$\begin{aligned} M_{\text{pre}} &= \texttt{55554654434b555559495a41504a4c414c415452474144524a4447515247594c}, \\ M_1 &= \texttt{b63a14a596b5216e97e6d7cc7b0b014d1d533b4f882a207504dd06463e1f98ed}, \\ M_2 &= \texttt{b63aa4a596b5211697e620cc50202a4d1d534a4f882a20fc04dd2d46dffe79ed}, \\ M_1 \oplus M_2 &= \texttt{0000b0000000007800000f7002b2b2b0000007100000000890000 2b00e1e1e100}. \end{aligned}$$

We did not find any full collisions for more than 10 rounds. The main problem seems to be that with more rounds the probability decreases drastically and for 12 rounds the best characteristic has probability $2^{-302.9}$ which is lower than the 256 degrees of freedom we have due to the message input. For 12 rounds further optimizations are necessary to have a chance of obtaining practical attacks with this approach.

5 Conclusion

In this work we provided the first dedicated analysis of Romulus-H. We developed a new framework for analyzing the hash functions based on the Hirose DBL construction and the

MDP mode of operation. This framework uses the idea of joint differential characteristics which capture the dependencies within the Hirose DBL constructions. We applied this framework to Romulus-H and found practical collisions up to 10 out of 40 rounds and practical semi-free-start collisions up to 14 rounds.

Our framework shows that the dependencies between the two block cipher calls in the Hirose DBL construction should be taken into account. These dependencies increase the number of rounds that can be attacked as fewer S-boxes need to be considered. Our results only apply to round-reduced version of Romulus-H and do not threaten the security of the full version. Based on these results, we gain valuable insights into the dependencies within the Hirose DBL construction.

These insights warrant future work on Romulus-H and other hash functions based on Hirose DBL. As our main bottleneck is finding satisfiable characteristics, we believe further research into why so many characteristics are not satisfiable might improve upon our results. A thorough understanding of these conflicts that lead to not satisfiable characteristics, would allow avoiding them much earlier. A big source of these conflicts likely is the limited key addition of SKINNY as we have seen when searching for full collisions. Another potential direction for future work is to explore different cellwise characteristics as a starting points.

In this work we have focused on practical attacks, which leads to the question whether we can extend the number of attacked rounds based on theoretical estimates. One idea is to apply the approach of Wei et al. to Romulus-H to find free-start collisions. Furthermore, a version of the Rebound attack that considers joint differential characteristics might give semi-free-start collisions for even more rounds. We conclude that our research on joint differential characteristics may serve as a starting point for several lines of research.

References

[AK18] Ralph Ankele and Stefan Kölbl. Mind the gap - A closer look at the security of block ciphers against differential cryptanalysis. In *SAC*, volume 11349 of *LNCS*, pages 163–190. Springer, 2018.

[Bie17] Armin Biere. Cadical, Lingeling, Plingeling, Treengeling and YalSAT entering the SAT competition 2018. *Proceedings of SAT Competition*, 14, 2017.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO (2)*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In *CRYPTO*, volume 537 of *LNCS*, pages 2–21. Springer, 1990.

[CHKM14] Jiageng Chen, Shoichi Hirose, Hidenori Kuwakado, and Atsuko Miyaji. A collision attack on a double-block-length compression function instantiated with round-reduced AES-256. In *ICISC*, volume 8949 of *LNCS*, pages 271–285. Springer, 2014.

[CKS21] Amit Kumar Chauhan, Abhishek Kumar, and Somitra Kumar Sanadhya. Quantum free-start collision attacks on double block length hashing with round-reduced AES-256. *IACR Trans. Symmetric Cryptol.*, 2021(1):316–336, 2021.

[DEM15] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Analysis of SHA-512/224 and SHA-512/256. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 612–630. Springer, 2015.

[dMB08] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[DR06] Christophe De Cannière and Christian Rechberger. Finding SHA-1 characteristics: General results and applications. In *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.

[ENP19] Maria Eichlseder, Marcel Nageler, and Robert Primas. Analyzing the linear keystream biases in AEGIS. *IACR Trans. Symmetric Cryptol.*, 2019(4):348–368, 2019.

[GIM22] Chun Guo, Tetsu Iwata, and Kazuhiko Minematsu. New indifferentiability security proof of MDPH hash function. *IET Inf. Secur.*, 16(4):262–281, 2022.

[GLLP22] Jian Guo, Shun Li, Guozhen Liu, and Phuong Pham. Rebound attacks on SKINNY hashing with automatic tools, 2022. https://ia.cr/2022/1057.

[Hir06] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In *FSE*, volume 4047 of *LNCS*, pages 210–225. Springer, 2006.

[HMRS12] Ekawat Homsirikamol, Pawel Morawiecki, Marcin Rogawski, and Marian Srebrny. Security margin evaluation of SHA-3 contest finalists through SAT-based attacks. In *CISIM 2012*, volume 7564 of *LNCS*, pages 56–67. Springer, 2012.

[HPY07] Shoichi Hirose, Je Hong Park, and Aaram Yun. A simple variant of the Merkle-Damgård scheme with a permutation. In *ASIACRYPT*, volume 4833 of *LNCS*, pages 113–129. Springer, 2007.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The Romulus and Remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[Knu94] Lars R. Knudsen. Truncated and higher order differentials. In *FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.

[Kö] Stefan Kölbl. CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives. https://github.com/kste/cryptosmt.

[LMS+15] Mario Lamberger, Florian Mendel, Martin Schläffer, Christian Rechberger, and Vincent Rijmen. The rebound attack and subspace distinguishers: Application to Whirlpool. *Journal of Cryptology*, 28(2):257–296, 2015.

[MAS15] Bodhisatwa Mazumdar, Sk Subidh Ali, and Ozgur Sinanoglu. Power analysis attacks on ARX: an application to salsa20. In *IOLTS*, pages 40–43. IEEE, 2015.

[MNS11] Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 288–307. Springer, 2011.

[MNS13] Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: New attacks on reduced SHA-256. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 262–278. Springer, 2013.

[MR22] Rusydi H. Makarim and Raghvendra Rohit. Towards tight differential bounds of ascon A hybrid usage of SMT and MILP. *IACR Trans. Symmetric Cryptol.*, 2022(3):303–340, 2022.

[MRST09] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In *FSE 2009*, volume 5665 of *LNCS*, pages 260–276. Springer, 2009.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *Inscrypt*, volume 7537 of *LNCS*, pages 57–76. Springer, 2011.

[MZ06] Ilya Mironov and Lintao Zhang. Applications of SAT solvers to cryptanalysis of hash functions. In *SAT*, volume 4121 of *LNCS*, pages 102–115. Springer, 2006.

[Nai19] Yusuke Naito. Optimally indifferentiable double-block-length hashing without post-processing and with support for longer key than single block. In *LATINCRYPT*, volume 11774 of *LNCS*, pages 65–85. Springer, 2019.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In *CRYPTO 2017*, volume 10401 of *LNCS*, pages 570–596. Springer, 2017.

[SKP16] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full SHA-1. In *EUROCRYPT 2016*, volume 9665 of *LNCS*, pages 459–483. Springer, 2016. arXiv:2015/967.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In *SAT*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[WPS+12] Lei Wei, Thomas Peyrin, Przemyslaw Sokolowski, San Ling, Josef Pieprzyk, and Huaxiong Wang. On the (in)security of IDEA in various hashing modes. In *FSE*, volume 7549 of *LNCS*, pages 163–179. Springer, 2012.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.

[ZHWW20] Hongluan Zhao, Guoyong Han, Letian Wang, and Wen Wang. MILP-based differential cryptanalysis on round-reduced Midori64. *IEEE Access*, 8:95888–95896, 2020.