Exploring SAT for Cryptanalysis: (Quantum) Collision Attacks against 6-Round SHA-3

Jian Guo[1], Guozhen Liu[1], Ling Song[2], and Yi Tu[1]

[1] Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
[2] Jinan University, Guangzhou, China
guojian@ntu.edu.sg, guozhen.liu@ntu.edu.sg, songling.qs@gmail.com, tuyi0002@e.ntu.edu.sg

Abstract. In this work, we focus on collision attacks against instances of SHA-3 hash family in both classical and quantum settings. Since the 5-round collision attacks on SHA3-256 and other variants proposed by Guo *et al.* at JoC 2020, no other essential progress has been published. With a thorough investigation, we identify that the challenges of extending such collision attacks on SHA-3 to more rounds lie in the inefficiency of differential trail search. To overcome this obstacle, we develop a SAT-based automatic search toolkit. The tool is used in multiple intermediate steps of the collision attacks and exhibits surprisingly high efficiency in differential trail search and other optimization problems encountered in the process. As a result, we present the first 6-round classical collision attack on SHAKE128 with time complexity $2^{123.5}$, which also forms a quantum collision attack with quantum time $2^{61.4}$, and the first 6-round quantum collision attack on SHA3-224 and SHA3-256 with quantum time $2^{96.15}$ and $2^{102.65}$, both with negligible requirement of classical and quantum memory. The fact that classical collision attacks do not apply to 6-round SHA3-224 and SHA3-256 shows the higher coverage of quantum collision attacks, which is consistent with that on SHA-2 observed by Hosoyamada and Sasaki at CRYPTO 2021.

Keywords: SHA-3, SAT-based Automatic Search Tool, Collision Attacks, Quantum Cryptanalysis

1 Introduction

The Keccak hash function [BDPVA13], designed by Bertoni *et al.* in 2008, was standardized as the Secure Hash Algorithm-3 (SHA-3) [Dwo15] in 2015 by the National Institute of Standards and Technology (NIST) of the U.S. The SHA-3 family has four instances with fixed digest lengths, namely, SHA3-224, SHA3-256, SHA3-384 and SHA3-512, and two eXtendable-Output Functions (XOFs) SHAKE128 and SHAKE256.

Being one of the most important cryptographic hash functions at present, SHA-3 (Keccak) received intensive security analysis. The most relevant security criteria for cryptographic hash functions include preimage resistance and collision resistance. Preimage attacks of SHA-3 were investigated in [NRM11, MPS13,

GLS16,LSLW17,LS19,Raj19,LHY21,HLY21]. The best-known practical attacks reach 3 rounds of SHAKE128 and SHA3-224 [GLS16, LS19][3] while the best-known theoretical ones can reach 4 rounds of all its instances [MPS13, Raj19, HLY21]. With marginal time complexity gains over bruteforce, theoretical preimage attacks cover up to 7/8/9 rounds for Keccak-224/256/512, respectively [CKMS14, Ber10, MPS13].

More relevant to this research are the collision attacks on SHA-3 (Keccak) with reduced number of rounds. In [DDS12,DDS14], Dinur *et al.* presented practical collision attacks on 4 rounds of Keccak-224 and Keccak-256. The actual collisions were found by combining a 3-round differential trail and a 1-round connector (which connects the differential trail to valid initial values). The same authors also presented practical collision attacks on 3-round Keccak-384/Keccak-512, and theoretical collision attacks on 5/4-round Keccak-256/Keccak-384 using internal differentials [DDS13]. Following the framework proposed by Dinur *et al.* in [DDS12], Qiao *et al.* introduced 2-round connectors by prepending a fully linearized round to the 1-round connectors and obtained actual collisions for 5-round SHAKE128 [QSLG17]. Further, these connectors were improved in [SLG17,GLL+20] to consume fewer degrees of freedom by using partial linearization. Consequently, 3-round connectors became possible and practical collision attacks on 5-round SHA3-224 and SHA3-256 were obtained.

Collision attack in quantum settings. In the previous works, collision attacks of SHA-3 were studied only in classical settings. Recently, quantum collision attacks are attracting more attention and showing unexpected efficiencies.

The generic security margin of collision attacks in quantum settings has been investigated with the recent progress in post-quantum security of cryptographic schemes and primitives. Several quantum collision algorithms [BHT98,CNPS17] were introduced to provide security bounds for generic hash functions. However, the quantum collision attack against exact hash functions was not published until 2020 [HS20]. In this work, Hosoyamada and Sasaki demonstrated that differential trails of low probability that couldn't be utilized in classical collision attacks were exploited to mount quantum collision attacks of more rounds. Later, the authors extended their quantum collision search algorithms to other hash functions and proposed the first quantum collision attacks on SHA-2 at CRYPTO 2021 [HS21]. Additionally, results of quantum rebound attacks on AES hashing modes [DSS+20] and quantum multi-collision distinguishers [BGLP] on dedicated hash functions were also presented.

Challenges. There are two major challenges in mounting quantum collision attacks on SHA-3. The first is to search for differential trails that are more suitable for quantum collision attacks, *i.e.*, trails that cover as many rounds as possible with the bound on the probability relaxed to 2^{-n}. As a consequence, the search space expands drastically which calls for more advanced and efficient searching techniques. The second challenge lies in connecting the differential trail

[3] The preimage attack on 3-round Keccak-256 in [LHY21] has a time complexity 2^{65}, but no concrete preimage is given.

with the initial state. When differential trails with lower probability are used, more conditions are imposed on the internal state which should be satisfied by the connector. Thus, to avoid being the bottle neck of the whole attack, connectors must be constructed in more efficient way than before.

SAT-based cryptanalysis. Great attention from the cryptographic community has been paid on automatic tools based on Boolean Satisfiability Problem (SAT), Mixed Integer Linear Programming (MILP), Satisfiability Modulo Theories (SMT), and other related methods for linear and differential trail search. Since the performance of automatic search is determined by the power of corresponding mathematical problem solvers, the efficiency is not particularly satisfactory when cryptographic ciphers with large state sizes are analyzed. Practically, most of the previous related works focus on lightweight block ciphers where the automatic tools showed incredible strength.

The SAT problem decides whether a set of constraints could be satisfied by giving valid assignments to variables. In the research line of SAT-based cryptanalysis, Mouha and Preneel searched differential trails of ARX ciphers with SAT method in [MP13]. Based on SAT, Sun *et al.* [SWW18] put forward an automatic search method for ciphers with Sboxes to obtain differential trails of more accurate as well as high probability. In [SWW21], Sun *et al.* proposed a new encoding method to convert the Matsui's bounding conditions into Boolean formulas, which could reduce clauses and speed up the SAT solving phase. Besides, Morawiecki and Srebrny presented preimage attack on 3-round KECCAK hash functions by developing a SAT toolkit [MS13].

Our Contributions. Inspired by Hosoyamada and Sasaki's findings from [HS20, HS21] that collision attacks in quantum settings can take advantage of differential trails of low probability, we develop an automatic trail search toolkit based on SAT and propose advanced collision attacks on SHA-3 in both classical and quantum settings. The results of our work and the comparison with previous works are listed in Table 1. Main contributions are summarized in the following.

1. **The SAT-based automatic trail search toolkit** To facilitate differential trail search of the underlying permutation $Keccak\text{-}f$ of `SHA-3`, an SAT-based automatic search toolkit is developed. The toolkit is not only simple to implement but also provides more flexibility and better efficiency in generating various differential trails compared to dedicated trail search strategies in previous works [DVA12, MDA17, LQT19]. It's interesting to note that for cryptographic primitives of large state size like $Keccak\text{-}f$, automatic tools that are based on mathematical problems such as the most widely applied MILP are unlikely to provide advantage in trail search. That's why specialized search techniques were proposed for `SHA-3`. Surprisingly, the SAT-based automatic toolkit fills the vacancy and shows excellent performance in trail search of the large-state $Keccak\text{-}f$.
2. **Advanced collision attack algorithms for `SHA-3`** Augmented with the SAT-based automatic search tool, the collision attack methods used in [DDS12,

Table 1: Summary of collision attacks against the SHA-3 family

Target	Type	Rounds	Time Complexity	Reference
SHA3-224	Classical	5	Practical	[GLL+20]
	Quantum	**6**	$2^{96.15}$	Sect. 4.4
SHA3-256	Classical	5	Practical	[GLL+20]
	Quantum	**6**	$2^{102.65}$	Sect. 4.3
SHA3-384	Classical	4	2^{147}	[DDS13]
SHA3-512	Classical	3	Practical	
SHAKE128	Classical	5	Practical	[GLL+20]
	Classical	**6**	$2^{123.5}$	Sect. 4.2
	Quantum	**6**	$2^{61.4}$	
SHAKE256	-	-	-	-

DDS14, QSLG17, SLG17, GLL+20] are improved in multiple ways. Collision attacks proposed in those works primarily consist of two phases, i.e., a phase of differential trail search that ensures collision on the digest bits, also referred to as the *colliding trail* search phase in our work, and a second phase of constructing "connectors" that generates message pairs satisfying the constraints imposed by the padding rule and initial value of SHA-3 and the input difference of the colliding trail at the same time. Both phases are considerably improved utilizing our automatic tool.

- Colliding trail search algorithms that generate colliding trails of *any rounds*, *any digest length*, and *high probability* are presented. In other words, search space of colliding trails is covered efficiently which has been impossible in previous works.
- Improved connector construction algorithms are proposed. Differential trails of the connectors (which are called *connecting differential trails* in the rest of the paper) can not only be directly generated but also produce sufficient degrees of freedom which has been the bottleneck in extending the collision attacks to more rounds.

3. **The first 6-round collision attacks on SHA-3** With the novel automatic search tool and the improved algorithms, we finally extend the 5-round collision attacks on SHA-3 instances to 6-round. In detail, 6-round classical collision attacks on SHAKE128 with complexity $2^{123.5}$, 6-round quantum collision attacks on SHA3-224 and SHA3-256 with complexity $2^{96.15}$ and $2^{102.65}$ respectively, are mounted . To the best of our knowledge, this is the first time that quantum collision attacks are mounted on SHA-3 and one more round is covered compared with previous results in classical setting.

Organization. The rest of the paper is organized as follows. In Section 2, an overview of the SAT-aided collision attacks on SHA-3 instances is provided. In Section 3, specifications of SHA-3 hash functions and implementations of the SAT-based automatic search toolkit are presented. Section 4 exhibits the first 6-

round collision attacks on SHA-3 in both classical and quantum settings. Section 5 concludes the paper. Details of differential trails and message pairs are given in the supplementary material.

2 Overview of SAT-based Collision Attacks against SHA-3

In this section, limitations of previous collision attacks are discussed. Subsequently, the SAT-based automatic trail search toolkit that can be conveniently applied to all kinds of cryptanalytic scenarios are introduced. Basic ideas used to extend previous collision attacks by one round in both classical and quantum settings are also presented.

2.1 Limitations of Previous Collision Attacks

As depicted in Figure 1, the collision attacks on SHA-3 and KECCAK instances take a 3-stage analytic framework, *i.e.*,

- at stage 1, prepare n_{r_2}-round colliding trails of high probability that ensure d-bit digest collision. ΔS_I and ΔS_O stand for the input and output difference of colliding trails.
- at stage 2, construct n_{r_1}-round connectors that promise a subspace of message pairs which meet both the message difference $\Delta\overline{M}$ imposed by the sponge construction[4] and the input difference ΔS_I of the colliding trails.
- at the last stage, exhaustively enumerate the messages pairs generated with the connectors until one message pair that collides in digest bits is found.

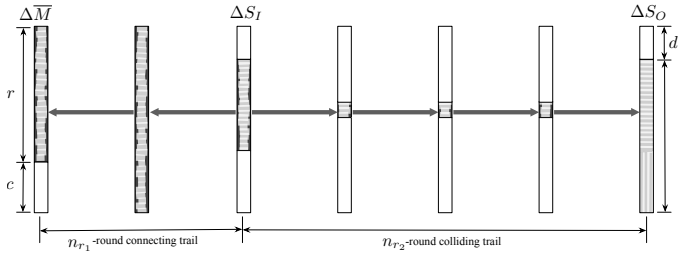

Figure 1: Overview of $(n_{r_1}+n_{r_2})$-round collision attack on SHA-3

[4] In this attack model, collision messages of 1-block are generated. The constraints imposed by the sponge construction include (1) the c-bit capacity, *i.e.*, c continuous "0" bits, and (2) 2-bit padding "11" which is concatenated with a "01" or "1111" string at the tail of the message block.

A continuous series of investigations [DDS12,DDS14,QSLG17,SLG17,GLL+20] have been conducted on collision attacks against SHA-3. Both the colliding trail search phase and the connector construction phase have been intensively inspected. At first glance, it seems that there is no room for further improvements. Actually, no essential progress has ever been published since the last work [SLG17] presented five years ago. The lack of new results can be explained from two aspects, *i.e.*, the constrained and low-efficiency colliding trail search algorithms, and the quick consumption of degrees of freedom from the connectors by (full) linearization.

2.1.1 Difficulty in generating colliding trails of more rounds. Due to the huge state size of $Keccak\text{-}f$, trail search of any kind, be it the general differential trail or the colliding trail, is a difficult task. In previous collision attacks, the strategy adopted to search colliding trails is quiet simple, *i.e.*,

1. General 3-round differential trails obtained from dedicated trail search algorithms, *e.g.*, [DVA12, MDA17, LQT19], are extended forward by one round and exhaustively searched for possible d-bit collision. Due to limited computation power and the enormous search space, only a small fraction of the trails are checked.
2. When sufficient many 3-round trails with digest collision are collected, extend them backward by one round to determine satisfactory output differences for connectors, which are also the input differences ΔS_I of the n_{r_2}-round colliding trails.

There are two problems about this colliding trail search strategy. On one hand, the exhaustive colliding trail search, especially the backward extension, drain computing resources significantly. In practice, sophisticated implementation techniques and even GPU resources [SLG17] are introduced to speed up the colliding trail search. However, without dramatical increase in computing power, it's unlikely that the search efficiency can be improved further. On the other hand, the colliding trails are limited by the results of general differential trail search. For example, the 5-round practical collision attack on SHA3-256 [GLL+20] is not possible until new results on general 3-round trails [LQT19] are published. Particularly, even with ultimate computing power, better colliding trails won't be possible unless results of general trail search are updated. Then it comes to the common trail search problem again which is a challenging task.

2.1.2 Quick consumption of degrees of freedom in connector construction. The connector construction is comprised of two parts. In the first part, as depicted in Figure 1, connecting trails whose input difference (*i.e.*, $\Delta\overline{M}$) and output difference (*i.e.*, ΔS_I) are partially or fully fixed are constructed. In the second part, data structures that output a subspace of message pairs following the connecting trail are generated. Essentially, as long as the connecting trail is determined, systems of equations (*i.e.*, the data structures) on messages are

listed in which the *degree of freedom* (DF for short) are quickly consumed[5]. As conditions of both $\Delta\overline{M}$ and ΔS_I are strict, the sophisticated Target Difference Algorithms (TDA for short) are devised to determine the exact connecting trails. When we try to extend the connector by one more round, $\Delta\overline{M}$ and ΔS_I are so heavy that connecting trails are hardly generated. Even if the TDA generates connecting trails, data structures become impossible to construct as almost all DF is consumed to meet conditions of the heavy connecting trail. Therefore, developing new connecting trail search methods to generate lighter connecting trails would be a feasible way to save DF and possibly allow to extend collision attacks for more rounds.

Summary. Limitations of collision attacks lie in inefficiency of differential trail search, more specifically, the lack of effective search techniques for trails of special requirements.

2.2 SAT-based Automatic Trail Search Toolkit

Automatic search has long been introduced to evaluate robustness of symmetric primitives. However, it's not the case of $Keccak\text{-}f$ permutation. Indeed, the initial attempts with MILP method failed to generate good trails due to the large $Keccak\text{-}f$ state. Researchers have to develop dedicated techniques to investigate the propagation properties of $Keccak\text{-}f$. On the other hand, automatic search based on other mathematical problems, such as SAT and SMT, is not properly studied. In this work, SAT-based automatic search shows productivity in generating trails involved in collision attacks on `SHA-3`. In line with limitations of the state-of-the-art collision attacks, we introduce in brief how the SAT-based tool facilitates collision attacks on `SHA-3`.

2.2.1 SAT-based colliding trail search. With the SAT-based automatic search tool, differential trails that (1) satisfy the d-bit digest collision, (2) cover more rounds, (3) follow any specific differential pattern, and (4) meet any probability constraint can be effectively generated. The search space is expanded to the extent that efficiency of automatic search tool outperforms dedicated search strategy. Moreover, as the new method does not rely on general differential trails, colliding trail search will not be limited by progress of general trails any more. Cryptanalysts are also free from devising and implementing sophisticated trail search algorithms. We emphasize that colliding trails of low probability, *e.g.*, with complexity near or even beyond the birthday bound, are easily generated, which was never possible with previous search strategy. Such colliding trails are utilized to mount collision attacks in quantum settings.

[5] The practical algorithms are much more complex. We just describe in this abstract way to express basic ideas.

2.2.2 SAT-based connecting trail search. Similar to the case of colliding trail search, the SAT-based connecting trail search method is effortlessly implemented. Good connecting trails that (1) follow the fixed input and output differences of the connectors and (2) provide adequate DF for messages are generated with the SAT-based tool. The idea of generating connecting trails with SAT gives some kind of insights to the constrained-input constrained-output (CICO) problem [BPVA+11] of sponge constructions. As the input and output differences of connecting trails are partially or fully fixed, this is generally a difficult problem. Except for the sophisticated approach used in [DDS12, DDS14, QSLG17, SLG17, GLL+20], there is no other progress on constructing connecting trails. The SAT-based connecting trail search method presents the first general solution for the problem of bypassing the constraints imposed by the sponge construction in collision attacks on `SHA-3`.

2.3 Improved (Quantum) Collision Attacks on `SHA-3`

With the SAT-based automatic tool, collision attacks on `SHA-3` instances that cover one more round are mounted in both quantum and classical settings.

2.3.1 6-round collision attacks on `SHAKE128`. With the SAT-based tool, 4-round colliding trails of 256-bit digest collision are generated. Although one round is extended compared to trails used in previous works, the 4-round colliding trails are of low probability. To mount valid collision attacks, one round of the colliding trails is merged into the connectors, *i.e.*, the 6-round collision attacks consist of a 3-round connecting trail and a 3-round colliding trail. Due to the low probability, the 3-round connectors can only be partially constructed, *i.e.*, only a fraction of the third round conditions are treated while the other constraints are left for the brute force stage. Ultimately, a theoretical 6-round collision attack on `SHAKE128` are mounted with a complexity $2^{123.5}$ which is slightly better than the generic attack.

2.3.2 6-round quantum collision attacks on `SHA3-224` and `SHA3-256`. The identical 4-round colliding trail is used to mount 6-round collision attacks on `SHA3-224` and `SHA3-256`. Constrained by the great amount of DF consumed, it becomes impossible to construct even partial 3-round connectors for these instances. Therefore, for `SHA3-224` and `SHA3-256`, only 2-round connectors are feasible. 6-round collision attacks on `SHA3-224` and `SHA3-256` cannot be mounted in classical setting as complexity of the 4-round colliding trail exceeds the birthday bound. Fortunately, colliding trails of low complexity can be employed to mount quantum collision attacks. In a nutshell, 6-round quantum collision attacks on `SHA3-224` and `SHA3-256` with complexity $2^{96.15}$ and $2^{102.65}$ are presented.

3 SHA-3 and SAT-based Automatic Search Toolkit

In this section, we describe notations used in the collision attacks and specifications of the SHA-3 family hash functions. Afterwards, the SAT-based automatic search toolkit developed for $Keccak\text{-}f$ permutation is presented.

3.1 Notations

Most of the notations to be used in this paper are listed below.

c	Capacity of a sponge function
r	Rate of a sponge function
d	Length of the digest in bits
p	Number of fixed bits in the initial state due to padding
n_r	Number of rounds
$Keccak\text{-}f$	The underlying permutation of SHA-3 hahs functions
$\theta, \rho, \pi, \chi, \iota$	The five operations of the round function of $Keccak\text{-}f$. A subscript i denotes the operation at the i-th round, *e.g.*, χ_i denotes the χ layer at the i-th round where $i = 0, 1, 2, \cdots$
λ	Composition of θ, ρ, π and its inverse denoted by λ^{-1}
RC_i	Round constant of the i-th round, where $i = 0, 1, 2, \cdots$
$\mathtt{R}^i(\cdot)$	$Keccak\text{-}f$ permutation reduced to the first i rounds
$\mathtt{S}(\cdot)$	5-bit Sbox operating on each row of $Keccak\text{-}f$ state
$\delta_{in}, \delta_{out}$	5-bit input and output differences of an Sbox
DDT	Differential distribution table, and $\mathtt{DDT}(\delta_{in}, \delta_{out}) = \ \lvert\{x : \mathtt{S}(x) + \mathtt{S}(x + \delta_{in}) = \delta_{out}\}\rvert$, where $\lvert \cdot \rvert$ denotes the size of a set
α_i	Input difference of the i-th round, where $i = 0, 1, 2, \cdots$
β_i	Input difference of χ in the i-th round, where $i = 0, 1, 2, \cdots$
w_i	Propagation weight (*weight* for short) of the i-th round
$w(\beta_i)$	Weight of β_i, where β_i is the input difference of χ
$w^{rev}(\alpha_i)$	Minimal reverse weight of α_i
DF	Degree of freedom of the solution space of connectors
$\overline{M}$	Padded message of M. Note that $\overline{M}$ is of one block in our attacks.
$M_1 \lvert\lvert M_2$	Concatenation of strings M_1 and M_2
x_i	Bit value vector before λ of each round, where $i = 0, 1, 2, \cdots$
y_i	Bit value vector before χ of each round, where $i = 0, 1, 2, \cdots$
E_{y_i}	System of equations on y_i of each round, where $i = 0, 1, 2, \cdots$

3.2 Description of SHA-3 family

The SHA-3 family [Dwo15] consists of a subset of Keccak [BDPVA13] hash functions that are built upon the sponge construction [BDPVA07, GJMG11] with an internal permutation called $Keccak\text{-}f$.

3.2.1 Specification of *Keccak-f* permutation. The underlying permutation *Keccak-f* takes a large state size of 1600 bits and there are 24 iterative rounds in total. Each round of *Keccak-f* is comprised of five operations, namely, the four linear operations denoted by θ, ρ, π and ι, and one solely nonlinear operation denoted by χ. The 1600-bit state is organized as a 3-dimensional array of bits, i.e., $5\times5\times64$, denoted with $A[5][5][64]$. Each of the state bits indexed by the coordinate (i, j, k) in the state array is denoted by $A[i][j][k]$ where $0 \leq i, j < 5$, and $0 \leq k < 64$. The 5 step mappings of the *Keccak-f* round are specified with the following transformations.

θ: $A[i][j][k] \leftarrow A[i][j][k] \oplus \sum_{j'=0}^{4} A[i-1][j'][k] \oplus \sum_{j'=0}^{4} A[i+1][j'][k-1]$.

ρ: $A[i][j] \leftarrow A[i][j] \lll T(i,j)$, where $T(i,j)$s are constants.

π: $A[j][2i+3j] \leftarrow A[i][j]$.

χ: $A[i][j][k] \leftarrow A[i][j][k] \oplus (A[i+1][j][k] \oplus 1) \cdot A[i+2][j][k]$.

ι: $A[0][0] \leftarrow A[0][0] \oplus RC_{i_r}$, where RC_{i_r} is the i_r-th round constant.

The multiplication used in χ operation is in $\mathsf{GF}(2)$. As ι won't affect the differences in our attacks, we ignore it in the rest of the paper unless otherwise stated.

3.2.2 Instances of `SHA-3` family. According to the bit length of digest, `SHA-3` contains 6 instances, *i.e.*, the four variants `SHA3`-224/256/384/512 that have a fixed hash length (where the numbers 224/256/384/512 stand for the hash size in bits) and the two variants `SHAKE128` and `SHAKE256` of extendable outputs. A multirate padding rule $\texttt{10}^*\texttt{1}$ is defined for all `SHA-3` instances. For the four standardized instances `SHA3`-224/256/384/512, a 2-bit string "`01`" is concatenated to the message before padded while the capacity is specified as $c = 2 \times d$. In regards to the two extendable variants, a 4-bit string "`1111`" is concatenated to the messages, and the capacity is 256 and 512 bits for `SHAKE128` and `SHAKE256` respectively. The digest size d of `SHAKE128` and `SHAKE256` can vary, and therefore the collision resistance level is given by $\min(d/2, 128)$ and $\min(d/2, 256)$ correspondingly.

3.3 SAT Implementation

In the following, the SAT solver employed, descriptions of the *Keccak-f* permutation and the related differential propagation, as well as the objective functions, are illustrated briefly.

CryptoMiniSAT We choose CryptoMiniSAT as the underlying SAT solver to implement our automatic trail search tools. Since proposed in [SNC09], the conflict-driven clause-learning(CLDL) SAT solver has been improved greatly with works in [SNC10, Soo14, Soo16, SBH+19, SDG+20, SSK+20]. Enhanced with a sequence of advanced search strategies such as Gauss-Jordan elimination and

target phases [QUE19], CryptoMiniSAT shows outstanding performances among other SAT solvers. Except for high performance, CryptoMiniSAT is selected to take advantage of its feature that an interface for XOR expressions is provided. In fact, most well-performed SAT solvers only understand constraints in *conjunctive normal form* (CNF for short). To exploit SAT solvers, cryptanalysts need to describe cryptographic primitives with CNFs which is generally challenging task. With XOR operation handled, CryptoMiniSAT allows attackers concentrate on problems emerged in collision attacks on `SHA-3` while providing high performance as well as simple implementation.

To implement SAT-based automatic trail search method, two kinds of constraints are fed into CryptoMiniSAT, namely, conditions imposed by (1) differential propagation over round functions (or in other words the description of round functions with CNFs), and (2) objective functions such as the number of active Sboxes and the propagation probability. Since CryptoMiniSAT provides very simple interface, the overall implementations are straightforward. Only basic ideas of listing all the constraints are described.

Round Function As depicted in the following model, two state differences, *i.e.*, α_r (the input difference of the r-th round) and β_r (the input difference of the χ operation of the r-th round) are introduced to the SAT implementation for a single r-th round.

$$\alpha_r \xrightarrow{\theta} c_r \xrightarrow{\pi\circ\rho} \beta_r \xrightarrow{\chi} \alpha_{r+1}$$

The 1600-bit difference α_r is represented by 1600 variables, *i.e.*, variable of each bit (whose coordinate is $\alpha_r[i][j][k]$ where $0 \leq i, j < 5$ and $0 \leq k < 64$) is indexed with $(320 \times j + 64 \times i + k)_{\alpha_r}$. This way we establish the mapping relationship between the 1600 variables and the corresponding state difference.

Recall that ρ and π are simply bit permutations. Therefore, differential propagations over the two linear operations are described through mapping the indexes of variables. For example, assuming that an active bit $c_r[i][j][k]$ is transformed to $\beta_r[i'][j'][k']$ through $\pi \circ \rho$, then the index mapping of the two variables is $(320 \times j + 64 \times i + k)_{c_r} \xrightarrow{\pi\circ\rho} (320 \times (2 \times i + 3 \times j) + 64 \times j + (k - T(i,j))\%64)_{\beta_r}$. These operations are described with easy index transformation of variables and no additional SAT computation is required.

By definition, θ operation updates each bit through XORing itself to two columns. Accordingly, θ is described with XOR clauses that could be directly understood by CryptoMiniSAT. That is, the XOR sums of 320 columns (denoted by $\alpha[i][k]$) are described with 320 variables each of which is indexed by $64 \times i + k$. As a result, the mapping of variable indexes induced by θ operation is captured with $(320 \times j + 64 \times i + k)_{c_r} = (320 \times j + 64 \times i + k)_{\alpha_r} \oplus (64 \times (i-1) + k)_{ColumnSum} \oplus (64 \times (i+1) + (k-1))_{ColumnSum}$. Here, the subscript $ColumnSum$ indicates the variables of column sums.

Practically, the three linear operations (*i.e.*, θ, ρ and π) are treated as a whole. The total index mapping of variables is described with $(320 \times (2 \times i + 3 \times j) + 64 \times j + (k - T(i,j))\%64)_{\beta_r} = (320 \times j + 64 \times i + k)_{\alpha_r} \oplus (64 \times (i-1) + k)_{ColumnSum} \oplus (64 \times (i+1) + (k-1))_{ColumnSum}$.

In regard to the only nonlinear operation χ which is generally considered as 5-bit Sbox, both the *difference distribution table* (DDT for short) and the operation itself are interpreted with truth tables. Specifically,

- The DDT is described with listing a truth table of 11 variables, including 10 variables that represent input and output difference and 1 variable marking compatibility of DDT entries. When fed into Logical Friday (refer to http://sontrack.com), 46 CNFs are generated to describe the DDT. Differential propagation over χ, *i.e.*, relationship between the input difference β_r and output difference α_{r+1}, is then depicted with simply writing CNFs for each Sbox.
- Similarly, variables that correspond to the input and output values of χ are connected with CNFs generated from χ truth table. Empirically, 11 variables are needed to construct truth tables and 29 CNFs are produced.

In summary, $1600 \times 2 + 320 = 3520$ variables are used to describe one round of $Keccak\text{-}f$ permutation in the SAT-based implementation. The relationship among variables are specified with methods illustrated above. Identical round description that is of different variable sets is implemented for each round. Multiple rounds are described by connecting each round, *i.e.*, (1) the input variables of each round are the output variables of its previous round and (2) the output variables of each round are the input variables of its next round.

Objective Function In the context of 6-round collision attacks on SHA-3, the number of active Sboxes and the propagation weight (*weight* for short)[6] are the two mainly considered objectives in implementation. To describe the objective function, constraints on integers (*i.e.*, number of active Sboxes and weights) should be transformed to CNFs. The sequential encoding method [Sin05] is employed to describe addition over integers, *e.g.*, $\sum_{i=0}^{n} x_i \leq w$ where $w \geq 1$. In this process, $(n \times (w+1) - w)$ auxiliary variables are introduced. More specifically,

- *Constraint on the Number of Active Sbox.* To describe the number of active Sboxes of each χ, 320 variables are introduced to indicate whether an Sbox is active or not. The sum of all the variables needs to satisfy a threshold weight (say w), *e.g.*, $\sum_{i=0}^{320} x_i \leq w$. Accordingly, $(320 \times (w+1) - w)$ extra variables are introduced to transform the constraint on the number of active Sboxes to CNFs.
- *Constraint on the Propagation Weight.* The DDT entries take 4 possible values (*i.e.*, 2, 4, 8, and 32), and the corresponding propagation weights belong to $\{0, 2, 3, 4\}$. As shown in Equation 1, four auxiliary variables denoted by (p_0, p_1, p_2, p_3) are introduced to represent the weight of each Sbox, meaning that $(320 \times 4 \times (w+1) - w)$ extra variables are added to describe constraints

[6] The *propagation weight* is defined as the opposite of the binary logarithm of the propagation probability. For example, if the propagation probability of a differential trail is 2^{-32}, the corresponding weight is 32.

on the weight of a whole state. Likewise, the weight constraint which is obtained through summing up all the variables is then transformed to CNFs.

$$(p_0, p_1, p_2, p_3) = \begin{cases} (1,1,1,1), & \texttt{DDT}(\delta_{in}, \delta_{out}) = 2; \\ (0,1,1,1), & \texttt{DDT}(\delta_{in}, \delta_{out}) = 4; \\ (0,0,1,1), & \texttt{DDT}(\delta_{in}, \delta_{out}) = 8; \\ (0,0,0,0), & \texttt{DDT}(\delta_{in}, \delta_{out}) = 32. \end{cases} \tag{1}$$

3.4 SAT-based Automatic Search Toolkit

In this section, we explain how to implement various trail search algorithms based on the SAT implementation. Let's first review some definitions and concepts introduced in [DVA12,BPVA+11]. The 6-round attack model presented in Section 4.1.3 is placed here in advance to better explain definitions.

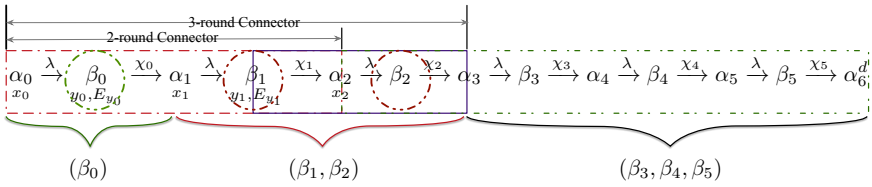

Figure 2: The 6-round collision attack model

Probabilistic property of χ. As the algebraic degree of χ is 2, its DDT shows some interesting properties. For a given input difference, all its compatible output differences share equal propagation probability. Correspondingly, for a given β_i, all its compatible α_{i+1} take the same probability or weight. On the contrary, for a given output difference, there exist one or several compatible input differences that hold a better probability than the other input differences. Likewise, for a given α_i, there exist some compatible β_{i-1} that have the best differential probability, which is also called the *minimum reverse weight* (and generally denoted by $w^{rev}(\alpha_i)$).

Trail core. As depicted in Figure 2, a general 4-round differential trail consists of input and output differences of all four rounds, *i.e.*, $(\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$. Recall that as λ is linear transformation, α_i propagates to β_i deterministically. The 4-round differential trail is also denoted by $(\beta_2, \beta_3, \beta_4, \beta_5)$. Comparatively, the 4-round ***trail core*** is composed of three differences, i.e., $(\beta_3, \beta_4, \beta_5)$, taking advantage of the property that the minimal reverse weight

of α_3 can be directly computed to evaluate the family of 4-round trails that have $(\beta_3, \beta_4, \beta_5)$ as their tail.

In the Figure 2 model, $(\beta_3, \beta_4, \beta_5)$ represents the colliding trail, and (β_1, β_2) and (β_0) represent the connecting trail.

3.4.1 SAT-based colliding trail search. To set up the colliding trail search model, description of differential trail (α_3,β_3,α_4,β_4,α_5,β_5,α_6^d) needs to be added into the SAT model. Differential propagation over the round functions is implemented in the way introduced in last section. At this stage, only constraints that are exclusively imposed by the colliding trails are introduced. Aligned with the requirements for constructing colliding trails in [GLL+20], the SAT-based search method is implemented from two aspects, *i.e.*, the digest collision and the connector construction.

From the perspective of collision search, we don't have to check α_6 for d-bit collision (denoted by α_6^d). Rather, extra constraints on β_5 that ensure α_6^d collision are considered. Take colliding trail search of `SHAKE128` as an example, to guarantee the first 4 lanes of α_6 to be 0, the input difference to the first 64 Sboxes of β_5 must belong to the set `{00000, 00001, 00101, 10101, 00011, 01011, 00111, 10111, 01111, 11111}`. The candidate input differences listed above form a space which is represented by CNFs. Through adding the corresponding CNFs on variables of β_5 to the system, constraints on digest collision is implemented.

On the other hand, to maximumly facilitate the connector, the *minimum reverse weight* of α_3 (denoted by $w^{rev}(\alpha_3)$) and propagation weight $w(\beta_3) + w(\beta_4) + w(\beta_5^d)$ of the colliding trail are taken into consideration. Altogether, the objective function of $w^{rev}(\alpha_3) + w(\beta_3) + w(\beta_4) + w(\beta_5^d)$ is described with CNFs and added to the system. To speed up the SAT solving phase, the constraints on weight is transformed to the number of active Sboxes, *i.e.*, $AS(\alpha_3) + AS(\alpha_4) + AS(\beta_4) + AS(\beta_5^d)$ which results in $(320 \times 3 \times (w+1) - w) + (64 \times (w+1) - w)$ auxiliary variables included to the SAT system.

With this implementation, 3-round colliding trails are not only generated more efficiently but also of better probability. In contrast, the best 3-round colliding trail used in previous collision attack on `SHA3-256` is of probability 2^{-43}. It's worth noticing that 4-round colliding trails which could be utilized to mount collision attacks of 6 rounds is generated for the first time. Table 2 gives comparison of search efficiency. It demonstrates that the new SAT-based trail search is superior to earlier strategies in both efficiency and effectiveness.

3.4.2 SAT-based connecting trail search. In accordance with the considerations for constructing connecting trails that promise valid connectors, the trail search of (α_0,β_0,α_1,β_1,α_2,β_2) is specified with two phases.

Phase 1. In the first phase, (β_1, β_2) are to be determined for given α_3. First, description of the differential trail (β_1,α_2,β_2,α_3) are added to the SAT system.

Table 2: Comparison of the SAT-based tools with other dedicated approaches

Type	Permutation	Rounds	Weight	Time	Reference
Colliding trail	*Keccak-f*[1600]	3	43	Several weeks*	[GLL+20]
		3	32	2s†	Secti. 3.4.1
		4	141	5mins†	Secti. 3.4.1
General trail	*Keccak-f*[1600]	4	134	-	[MDA17]
			133	47.76h	Sect. 3.4.3
	Keccak-f[800]	4	104	-	[MDA17]
			95	28.42h	Sect. 3.4.3

* There are two stages, *i.e.*, the forward extension executed with one CPU core and the backward extension deployed with three NVIDIA GeForce GTX970 GPUs.
† The SAT-based implementation is deployed with one 3.6 GHz Intel Core i9.

Afterward, constraints on propagation weight of β_1 and β_2 are established, *i.e.*, CNFs of a minimal $w(\beta_1) + w(\beta_2)$ are listed. By now, $6400 + 320$ variables are used to describe the connecting trail where 6400 variables are introduced for the 2-round propagation and 320 variables correspond to conditions of the summed weight. And we also restrict weight of each round, namely, $w(\beta_1) \leq w_1$ and $w(\beta_2) \leq w_2$ which results in an extra $(1280 \times (w_1 + 1) - w_1) + (1280 \times (w_2 + 1) - w_2)$ variables. The objective function of weight is described with the method illustrated in the last section. Overall, this model needs $6400 + 320 + (1280 \times (w_1 + 1) - w_1) + (1280 \times (w_2 + 1) - w_2)$ variables.

Phase 2. The input difference of χ_0 of the first round is determined in this phase with the SAT-based implementation. Given the output difference α_1, variables that represent a pair of messages (x_0^1, x_0^2) and the input difference β_0 are introduced to describe the half round propagation. Precisely, constraints on bit positions of capacity and padding are depicted by fixing the corresponding variables to be 0 or some settled value. Constraints on $w(\beta_0)$, the weight of β_0, are also covered to make sure that the degree of freedom will be maximally produced for connectors. Simply put, CNFs for objective function of a minimal $w(\beta_0)$ are added to the SAT model. With the SAT-based implementation, connecting trails that yield much greater DF are generated.

3.4.3 SAT-based general trail search. Except for the special trail search scenarios, SAT-based solution also performs well in general differential trail search. As can be seen from the experimental results, SAT-based implementation handles 3-round *Keccak-f* permutation quickly. It turns out that 3-round trail cores generated with the SAT-based automatic trail search method are consistent with results from previous works [DVA12, MDA17, LQT19].

We take 4-round differential trail search as an example to explain the SAT-based trail search implementation. The 4-round trail is modelled with

$$\beta_2 \xrightarrow{\chi} \alpha_3 \xrightarrow{\lambda} \beta_3 \xrightarrow{\chi} \alpha_4 \xrightarrow{\lambda} \beta_4 \xrightarrow{\chi} \alpha_5 \xrightarrow{\lambda} \beta_5 \xrightarrow{\chi} \alpha_6.$$

First, CNF description of the differential trail (α_3,β_3,α_4,β_4,α_5,β_5) is added to the SAT system. As 6 differences are involved, $10560 = 1600\times6+3\times320$ variables are required to describe the difference propagation as well as the number of active Sboxes. Similar to the colliding trail search implementation, constraint on the sum of weight $w = w^{rev}(\alpha_3) + w(\beta_3) + w(\beta_4) + w(\beta_5)$ where $w \leq 133$ is also added to the SAT system. Another $685947 = (1280\times4\times(133+1)-133)$ auxiliary variables are included in the process of transforming the objective function to CNFs. In total, there are 696507 variables in this SAT-based 4-round differential trail search implementation.

With respect to search efficiency, although it displays unexpectedly well performance in 3-round trail search, it cannot traverse the search space of 4-round trails efficiently. A tight lower bound on propagation weight for 4-round differential trails is unfortunately not settled in this paper. However, two better 4-round trails of weight 133 which is the lowest known weight so far are generated. Table 8 in supplementary material B shows the two trails.

The SAT-based differential trail search implementation is further extended to other Keccak permutations [BPVA+11] such as *Keccak-f*[800]. Analogous to *Keccak-f* (which is also denoted by *Keccak-f*[1600]), similar round functions are iterated for multiple rounds in *Keccak-f*[800] only that its state size is of 800 bits. Table 9 in supplementary material B shows a good trail that improves the lower bound of 4-round trails for *Keccak-f*[800]. Table 2 gives an overview of the advantage of the automatic search compared to previous works.

Summary. By picking up different compositions of constraints on the number of active Sboxes and weight or even considering a single state not in the whole, we obtain variant SAT models with different efficiency. The SAT-based automatic search toolkit helps us understand the differential propagation property of *Keccak-f* in a distinct viewpoint. It also demonstrates that automatic solvers could also perform efficiently on cryptographic primitives with large state size.

4 Collision Attacks against SHA-3 Instances in Classical and Quantum Settings

In this section, a classical 6-round collision attack on SHAKE128, and two 6-round quantum collision attacks on SHA3-224/SHA3-256 are mounted. Basic attack strategy will be illustrated before introducing the exact collision attacks. The methods, techniques, and results of each collision attack on the three SHA-3 instances will be explained in detail.

4.1 Basic Attack Strategy

Aided by the SAT-based automatic search toolkit, we propose advanced collision attacks on SHA-3 instances based on the analytic framework described in Section 2. The enhanced collision attack is comprised of three phases, *i.e.*,

- Phase 1, generate n_{r_2}-round colliding trails of d-bit digest with the SAT-based tool.
- Phase 2, generate n_{r_1}-round connecting trails that link the conditions of sponge construction and the input difference of the colliding trail with the SAT-based tool.
- Phase 3, construct connectors that generate a subspace of messages which follow the n_{r_1}-round connecting trails.

The brute force phase where collision messages are generated will not be included as only theoretical collision attacks are presented in this work.

4.1.1 Generating colliding trails. Based on the SAT implementation techniques elaborated in Section 3, we add the implementation of colliding trail search algorithms to the toolkit. Except that the d-bit collision must be satisfied, the propagation weight of the 4-round colliding trail core must also be small enough to promise a possible 6-round collision attack. Eventually, several 4-round colliding trail cores are generated. We select the best one to mount collision attacks. Without considering the connector, weight of the 4-round colliding trail is 141 (*i.e.*, $89+24+20+8=141$) . The propagation weight of the 4-round trail core is shown in Figure 3 while the exact differences are listed in Trail No.1 (shown in Table 5) of supplementary material B.

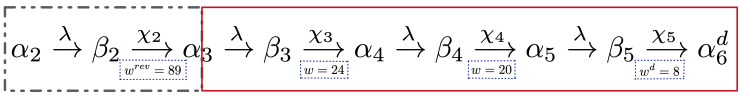

Figure 3: The 4-round colliding trail model. The 4-round trail is purposely placed at the last 4 rounds of a 6-round differential trail to be consistent with the collision attack model. In the last round, only d-bit collision is concerned and denoted by α_6^d.

4.1.2 Generating connecting trails. As shown in Figure 3, even the minimal weight (*i.e.*, ≥ 141) of 4-round colliding trails exceeds the birthday bound (*e.g.*, 128 for SHAKE128 and SHA3-256). It's impractical to randomly select a 4-round colliding trail and generate the corresponding 2-round connecting trail. We develop a two-step approach to determine the connecting trails. The input difference of the 4-round colliding trail core is generated in combination with the differences of connecting trails. Let's explain the idea with the 6-round collision attack model shown in Figure 4.

– In the first step, the input difference (*i.e.*, β_2) of the 4-round colliding trail core $(\beta_3, \beta_4, \beta_5)$ is determined together with the input difference (*i.e.*, β_1) of the second round of the connecting trails. Practically, the 2-round differential trails (β_1, β_2) that are not only compatible with α_3, but also of minimal weight are generated with the SAT-based tool.
– In the second step, the lightest β_0 (in terms of weight) that are compatible with α_1 and meet the restrictions on α_0 imposed by the sponge construction are generated with the SAT-based tool.

To demonstrate the strength of the SAT-based method, we compare experimental results on SHA3-256 with previous work. In previous results, when the first round of the connector is processed, the DF remained is estimated to be around 124 (for more illustration refer to Section 5.2 of [GLL+20]). In comparison, the new connecting trails provide a DF up to $330 \sim 430$ which is surprisingly superior. This accords with the number of active Sboxes of β_0. Almost all of the 320 Sboxes of β_0 are active (*e.g.*, the number of nonactive Sboxes is around 10) with the previous *target difference algorithm*, while with our SAT-based strategy there are around $40 \sim 50$ nonactive Sboxes in β_0. Without the extra gain of DF, it's impossible to extend the attack by one round.

Remark 1. The three undetermined differences β_0, β_1, and β_2 cannot be generated all at once. On one hand, even if $(\beta_0, \beta_1, \beta_2)$ are determined in one step, the distribution of weights (*i.e.*, $w(\beta_0)$, $w(\beta_1)$, and $w(\beta_2)$) is random. In our experiments, such $(\beta_0, \beta_1, \beta_2)$ cannot sustain a good connector in general. On the other hand, the SAT-based toolkit cannot support searching such trails efficiently.

4.1.3 Constructing connectors. The connecting trails, combined with the colliding trails, constitute the full 6-round differential trail with which the connectors that generate a subspace of messages that follow the connecting trails can be constructed. Considering that weight of the 4-round colliding trail exceeds the birthday bound, to mount a valid attack, we transfer the first round of the colliding trail to the connector. In detail, the 6-round collision attack on SHAKE128 consists of a 3-round connector and a 3-round colliding trail (refer to Figure 4). As for SHA3-224 and SHA3-256, 6-round quantum collision attacks that consist of a 2-round connector and a 4-round colliding trail are mounted (refer to Figure 4). We highlight that the connecting trails cannot provide enough DF to satisfy all the constraints in connectors even for theoretical attacks. Therefore, merely a fraction of constraints of the last round of 2/3-round connectors are treated.

2-round connectors. The algebraic-aided method adopted from previous works [DDS12, DDS14, QSLG17, SLG17, GLL+20] is improved to construct connectors that generate message pairs following partially the output difference of the connectors. Principally, the systems of linear equations on messages are listed and solved. The linear equations correspond to the conditions of sponge functions and differences of the connecting trail. The 2-round connector model

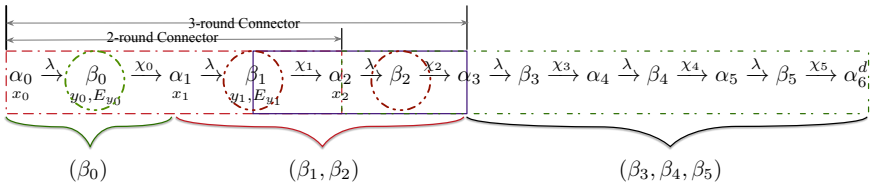

Figure 4: The 6-round collision attack model

exhibited in Figure 5 explains at length on how the system of linear equations is established.

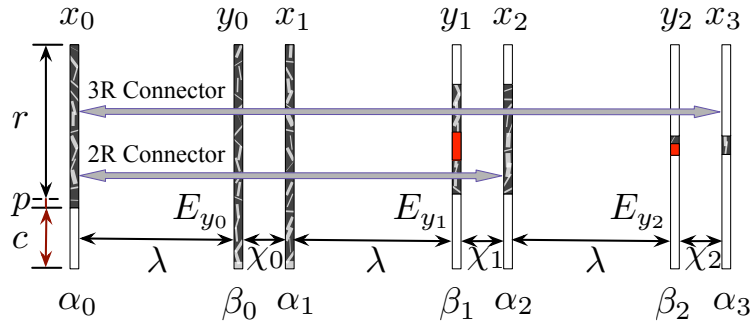

Figure 5: The 2-round and 3-round connectors

1. First, linear equations of the $(c + p)$-bit conditions imposed by the sponge construction are listed, where c and p correspond to the *capacity* and *padding* bits respectively. Take the case of `SHA3-256` as an example, the capacity is $c = 256 \times 2 = 512$ bits, and the padding rule is `10*1`. To provide as many DF as possible, we set the padding as fixed "`11`" string. Also the 2-bit string "`01`" is concatenated to the tail of the message block. In total, a 4-bit fixed string (*i.e.*, "`0111`") is considered as the p-bit condition.
   Linear equations on the $(c+p)$-bit conditions are directly listed on the input messages x_0. As y_0 and x_0 are linked with the linear transformation λ, the

linear equations on x_0 are easily transferred to equations on y_0. In the case of 2-round connectors, the systems of linear equations on y_0 are listed and denoted by E_{y_0}.

2. Next, linear equations on y_0 that meet conditions imposed by first round differential (β_0, α_1) are added to E_{y_0}. Message pairs constructed from the solutions of the current E_{y_0} system must follow the (β_0, α_1) differential. Details on how the equations can be listed are illustrated with Property 1 of the supplementary material A.
3. To list equations of conditions imposed by the second round differential (β_1, α_2), the first round must be bypassed. Linearization and partial linearization techniques on χ operation proposed in [QSLG17, SLG17] are borrowed directly to ensure that the y_1 bits can be expressed by the linear combinations of involved y_0 bits. Consequently, E_{y_1}, the system of linear equations on y_1 for (β_1, α_2), is transferred to a group of linear equations on y_0.
   To this end, extra equations on y_0 that allows the involved y_1 bits linear with respect to the χ operation must be added to E_{y_0}. Practically, as there is a whole round between y_1 and y_0, the x_1 bits that are involved to the corresponding y_1 bits according to λ operation are linearized. The principal property exploited to linearize x_1 bits is briefly summarized in Property 2 of the supplementary material A.
   The DF left after the last two steps cannot sustain solving all the β_1 active Sboxes. A greedy algorithm that sorts the active Sboxes of β_1 by the number of unlinearized x_1 bits is utilized to choose the β_1 Sboxes to be treated[7].
   To sum up, linear equations on y_0 that linearize the involved x_1 bits of partially chosen β_1 Sboxes are added to E_{y_0} in this step.
4. At last, the system of equations on y_1 (*i.e.*, E_{y_1}) of the partially treated β_1 Sboxes is transferred to linear equations on y_0 with the linearization equations generated in the last step, and added to the system E_{y_0}.

The Algorithm 1 shown in supplementary material A provides a concise description on construction of the 2-round connector. When a consistent system of linear equations on y_0 (*i.e.*, E_{y_0}) is successfully generated, the alleged 2-round connector is constructed. The solution space of E_{y_0} is composed of a subspace of messages, *i.e.*, y_0. A pair of messages (y_0^1, y_0^2) generated through XOR-ing y_0^1 with β_0, while y_0^1 is a random solution of E_{y_0}, follows (1) the input difference α_0 and (2) a fraction of the output difference α_2 of the 2-round connector.

3-round connector. In constructing 3-round connector, χ_0 of the first round is fully linearized, making the first round a linear layer. As a result, the 3-round connector can be viewed as a 2-round connector. We adopt the model shown in Figure 5 to explain how the system of linear equations of the 3-round connector is constructed.

1. First, list linear equations on y_0 for (1) the $(c + p)$-bit conditions and (2) the constraints imposed by the first round (β_0, α_1) differential. The system of linear equations is denoted by E_{y_0}.

[7] The other β_1 Sboxes that are not treated are indicated with red block in Figure 5.

2. Next, fully linearize the χ_0 layer of the first round and transfer the equations on y_0 to equations on y_1. Namely, additional equations on y_0 that corresponds to linearizing each active and non-active Sbox of (β_0, α_1) differential are added to the current E_{y_0}. Expressions of the linearized χ_0 are utilized to convert the system of linear equations on y_0 (*i.e.*, E_{y_0}) to the system of linear equations on y_1 (*i.e.*, E_{y_1}).
3. List linear equations on y_1 for constraints imposed by the second round (β_1,α_2) differential. Add those equations to the present system of equations E_{y_1}.
4. With the same greedy algorithm utilized in 2-round connector construction, select a fraction of conditions of β_2 to solve and linearize the related x_2 bits. Add the equations on y_1 that linearize the involved x_2 bits of the partially treated (β_2,α_3) differential to the current E_{y_1} system.
5. List equations on y_2 for conditions imposed by the partially solved (β_2,α_3) differential of the last round of the 3-round connector. Convert the system of linear equations on y_2 to equations on y_1 based on the linearization of involved x_2 bit in the last step. Add the y_1 equations generated at this step to the whole E_{y_1} system.

When all equations are listed and organized in the system of equations on y_1 (*i.e.*, E_{y_1}), the 3-round connector is successfully constructed. A subspace of message pairs generated from the solution space of E_{y_1} satisfy that (1) the input conditions imposed by sponge constructions are met and (2) the output difference of the 3-round connector is partially met as expected. The Algorithm 2 in supplementary material A illustrates construction of the 3-round connector.

4.2 Collision Attack against 6-Round SHAKE128

Following the basic attack strategy, a collision attack on 6-round SHAKE128 is mounted. The model in Figure 6 gives basic details of the attack.

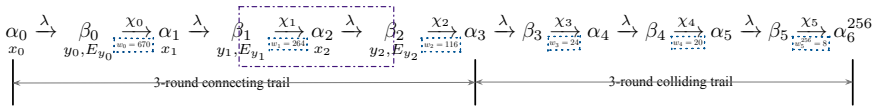

Figure 6: The 6-round collision attack model for SHAKE128

As discussed in Section 4.1.2, the minimal weight of the best 4-round colliding trail core exceeds the birthday bound. To make the collision attack feasible, the first round of the 4-round colliding trail is transferred to the connector. Hence, the 6-round collision attack consists of a 3-round connector and a 3-round

colliding trail. Propagation weight of each round is identified in Figure 6. The 4-round colliding trail core is specified in Table 5 of supplementary material B, more specifically, the $(\beta_3, \beta_4, \beta_5)$ differences of Trail No.1. The probability of the 3-round colliding trail is 2^{-52} (where $2^{-52} = 2^{-24} \cdot 2^{-20} \cdot 2^{-8}$). The two-step SAT-based connecting trail search method described in Section 4.1.2 is applied to first determine (β_1,β_2) differences and fix β_0 difference subsequently. The connecting trail is listed in Table 7, *i.e.*, Trail No.3 in supplementary material B.

Now that the whole 6-round differential trail is determined, the 3-round connector can be constructed with the method illustrated in Section 4.1.3. The third round of the 3-round connector is partially solved, *e.g.*, in our experiment, 36 out of the 116 constraints of (β_2,α_3) are solved. The DF of the 3-round connector is 27. Alternatively, the 3-round connector generates a subspace of 2^{27} messages that satisfy the 36 conditions of the input difference α_3 of the colliding trail. A pair of solution messages are given in Table 10.

The unsolved conditions of (β_2,α_3) are treated together with the colliding trail through exhaustive search. In the brute force phase, message pairs generated from connectors are verified for whether satisfying α_3 or not. If not, simply abandon the current pair and try another one. Otherwise, further check the 256-bit digests of the pair until a collision is encountered.

Remark 2. Apart from the current work that exemplifies the collision resistance of a typical 128-bit security level, inner collisions [GJMG11] could also be analyzed with the same idea. As indicated in [GLL+20] (an inner collision of a 160-bit Keccak Challenge), the inner collision attack that constructs collision on capacity bits yields collisions of any digest length.

Complexity. The overall complexity includes complexity of both the connector construction phase and the exhaustive search phase.

- *In the exhaustive search phase*, the time complexity is 2^{132} 6-round `SHAKE128` computations (where $2^{132} = 2^{116-36} \cdot 2^{52}$). However, taking advantage of the early-abort technique, the search process is sped up by iteratively filtering out half of the message pairs at each step. The cost of computing each additional bit constraint on β_2 equals to $\frac{11}{1600} \cdot \frac{1}{6} = 2^{-9.8}$ 6-round `SHAKE128` computation as 11 bits of α_2 states are involved. When checking all the 2^{132} message pairs with one bit constraint, only half of the pairs satisfy the restriction while the other half are discarded, *i.e.*, the so-called *early-abort*. For the remaining message pairs, another bit constraint will be checked and filter out half of those message pairs. This iterative process continues on the surviving message pairs until all the bit constraints on β_2 are checked. $1/2$ of the messages stop by first bit constraint, $1/4$ by the second bit constraint, $1/8$ by the third bit etc. Hence the time complexity would be $2^{132} \cdot 2^{-9.8} \cdot (1 \cdot 1/2 + 2 \cdot 1/4 + 3 \cdot 1/8 + \cdots) = 2^{123.2}$ 6-round `SHAKE128` computations.
- *In the connector construction phase*, the time complexity corresponds to the time used to construct 2^{105} (*i.e.*, $2^{132}/2^{27} = 2^{105}$) connectors. Let's first discuss the equivalent conversion of implementation efficiency between connector construction and 6-round `SHAKE128`. The computation cost of 6-round

SHAKE128 is $6 \cdot (\underbrace{(4 \cdot 320 + 2 \cdot 1600)}_{\theta} + \underbrace{3 \cdot 1600}_{\chi} + \underbrace{64}_{\iota}) = 56064$ bitwise operations. The time complexity of Gauss-Jordan elimination for system of boolean equations is $\mathcal{O}(m^2 n)$ bitwise operations [HJ12], where m is the number of equations and n is the number of variables. Assume there are 1600 non-redundant equations in the final system. The complexity would be $1600^3 = 4.096 \times 10^9$ bitwise operations[8]. Consequently, time cost of constructing a connector equals to $4.096 \times 10^9 / 56064 = 2^{16.2}$ 6-round SHAKE128. The time complexity in connector construction is equivalent to $2^{105} \cdot 2^{16.2} = 2^{121.2}$ 6-round SHAKE128 computations.

In total, time complexity of the classical collision attack is $2^{123.2} + 2^{121.2} = 2^{123.5}$ 6-round SHAKE128 computations. Complexity of quantum collision attack[9] is $2^{61.4}$.

Table 3 gives an overview of the time complexity tradeoff between brute force search phase and connector construction phase according to the number of constraints on β_2 solved. The more the constraints are solved, the bigger the DF of connectors is, the better the brute force complexity is and the worse the connector complexity is.

Table 3: Summary of complexity corresponding to the number of constraints solved

#constraints	DF of Connector	Data Complexity	Connector Complexity	Brute Force Complexity	Total Complexity
35	28	133	121.2	124.2	124.4
36	**27**	**132**	**121.2**	**123.2**	**123.5**
37	23	131	124.2	122.2	124.5
38	22	130	124.2	121.2	124.3
39	20	129	125.2	120.2	125.2
40	15	128	129.2	119.2	129.2
41	13	127	130.2	118.2	130.2
42	10	126	132.2	117.2	132.2
43	7	125	134.2	116.2	134.2
44	4	124	136.2	115.2	136.2
45	1	123	138.2	114.2	138.2

Remark 3. Experimental results outlined in Table 4 conforms to the theoretical complexity analysis of the connector construction phase. The average execution time of each connector construction (denoted by T_c) is 0.8 second. In our C++

[8] In our experiments, solving systems of equations dominates the time of connector construction. Although theoretically connectors won't always be successfully constructed, it succeeds with only one or two tries when the DF is large enough (*e.g.*, 28). We believe the complexity evaluation is reasonable.

[9] Complexity analysis of quantum collision attack will be illustrated in next section.

implementation, around 2^{20} 6-round SHAKE128 are computed in each second. The time of connector construction equals to $2^{105} \cdot 2^{19.67} = 2^{124.67}$ SHAKE128 computations which validates the attack.

Table 4: Experimental details of the collision attacks on 6-Round SHA-3 instances

Target	Type	Trail Core	T_c	DF	Complexity	Solution
SHAKE128	Classical Quantum	No. 3	0.8s	27	$2^{123.5}$ $2^{61.4}$	Table 10
SHA3-256	Quantum	No. 1	3s	5	$2^{102.65}$	Table 11
SHA3-224	Quantum	No. 2	3s	22	$2^{96.15}$	Table 12

4.3 Quantum Collision Attack against 6-Round SHA3-256

The colliding trail used in 6-round collision attacks on SHAKE128 is also used in attacks on SHA3-256 and SHA3-224. As shown in Figure 7, the 6-round collision attack on SHA3-256 consists of a 2-round connector and a 4-round colliding trail. Note that, the (β_1, β_2) used in the attack on SHAKE128 is also applied here. The entire 6-round differential trail is given in Table 5, *i.e.*, Trail No.1 in supplementary material B. The 2-round connector solves 226 out of the total

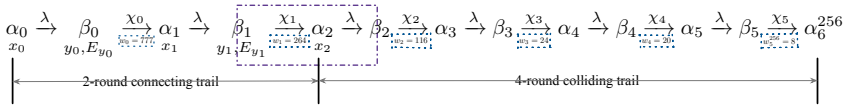

Figure 7: The 6-round collision attack model for SHA3-256

264 conditions imposed by (β_1,α_2). The solution space of the 2-round connector ensures a subspace of message pairs that follow partial α_2 difference as expected. In our experiment, the 2-round connector is constructed in 3 seconds on average. The DF of the connector is 5 or in other words the size of the solution space is 2^5. Example of a pair of messages that follow the connector is given in Table 11.

The unsolved conditions (*i.e.*, 38 left) of (β_1,α_2) are treated together with the colliding trail whose weight is $116 + 24 + 20 + 8 = 168$. In classical settings, the time complexity of the brute force phase is $2^{38+168} = 2^{206}$ 6-round SHA3-256 computations with which a valid collision attack cannot be conducted. However, such differential trails of low probability can be exploited in quantum settings.

Quantum collision attack. As stated in [HS21], no existing quantum collision attack on a random function could outperform classical attack based on parallel rho method [VOW94] in terms of time-space tradeoff. We follow their way and consider a quantum collision attack valid if its time complexity is less than $2^{n/2}/S$, where n denotes the digest length, and S is the hardware size required for the attack (or in other words, S is the maximum size of quantum computers and classical computers). We adopt the strategy of quantum collision attacks introduced in [HS21] to mount the 6-round quantum collision attack on SHA3-256.

Suppose there exists a quantum circuit $\mathcal{C}_1$ for the connector construction of depth T_c and width S_c. In other words, the circuit constructs a connector in time T_c with S_c qubits. Similarly, suppose there exists another quantum circuit $\mathcal{C}_2$ of depth T_s and width S_s for the one-block SHA-3 variants, *i.e.*, the quantum implementation of the 6-round targets (in this case SHA3-256). The idea that converts the classical attacks to the quantum collision attacks is described as follows.

1. Prepare message pairs (M, M') with the quantum circuit $\mathcal{C}_1$.
2. For each (M, M') pair, compute the digests with quantum circuit $\mathcal{C}_2$, and check whether they are identical.
3. Repeat the above two steps until a collision is found.

Complexity. Considering the solution space of the 2-round connector (which is 2^5), 2^{201} connectors are needed in theory. As only linear operations are involved in the quantum implementation of connector construction phase, comparing to T_s of the nonlinear SHA-3 variants, the depth T_c is negligible. Hence, time complexity of quantum collision attack is dominated by the time complexity of the exhaustive search phase.

Suppose we have a quantum computer of size S, taking parallelization into account, the time complexity of Grover search [Gro96] in the exhaustive search phase is

$$T_s \cdot (\pi/4) \cdot \sqrt{S_s/(p \cdot S)},$$

where p is the probability of finding a collision in the classical setting. The depth (resp. width) of quantum circuits (*i.e.*, $\mathcal{C}_2$) are defined as the unit depth (resp. width), meaning that $T_s = 1$ and $S_s = 1$. Therefore, the total time complexity of the quantum collision attack on 6-round SHA3-256 is

$$1 \cdot (\pi/4) \cdot \sqrt{2^{206}/S} = 2^{102.65}/\sqrt{S}.$$

Comparing to the generic attack cost under the time-space metric which is $2^{128}/S$, our quantum collision attack is valid as long as $S \leq 2^{50.7}$.

To show effectiveness of the quantum collision attack and give more sense on the complexity, we assume $S = 1$. It means the hardware resources of our quantum computer is assumed the same with the resources needed for each message pair check with the circuit $\mathcal{C}_2$. This is a conservative assumption as size of quantum computers would generally be much larger. Thus, Without further

specification, the complexity of the 6-round quantum collision attack on SHA3-256 is claimed as $2^{102.65}$.

Remark 4. In the quantum search, we should prepare 2^{206} messages which brings to the concern that whether it's possible to construct so many connectors. This concern could be answered through introducing multi-blocks. The first block (which is identical for the two messages) provides distinct capacity bits at each time which are used to construct different connectors of the same connecting trails. We can try as many as 2^{512} first blocks which are sufficient for the attack.

4.4 Quantum Collision Attack against 6-Round SHA3-224

As shown in Figure 8, the 6-round trail of SHA3-224 (which is listed in Table 7) is comprised of the same colliding trail used in attacks on SHAKE128 and SHA3-256 and a 2-round connecting trail searched with the SAT-based tool. In our

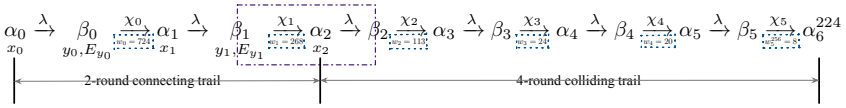

Figure 8: The 6-round collision attack model for SHA3-224

experiment, the 2-round connectors are averagely constructed in 3 seconds. The size of the solution space is 2^{22}. Example of a pair of messages that follow the connector is given in Table 12. The 2-round connector solves 240 out of the 268 conditions imposed by the (β_1,α_2) differential. Therefore, the classical complexity of the brute-force phase is $2^{28+113+24+20+8} = 2^{193}$ 6-round SHA3-224 computations. Similar to the attack on SHA3-256, we mount 6-round quantum collision attack on SHA3-224. Likewise, we adopt the strategy utilized in [HS21]. Suppose we have a quantum computer of size S, the complexity of our attack is

$$1 \cdot (\pi/4) \cdot \sqrt{2^{193}/S} = 2^{96.15}/\sqrt{S}.$$

under the time-space metric $2^{112}/S$, our quantum collision attack is faster than the generic attack when $S \leq 2^{31.7}$. Assume $S = 1$, the complexity of the 6-round quantum collision attack on SHA3-224 is claimed as $2^{96.15}$.

5 Conclusion

We investigate the previous collision attacks on SHA-3, identify the limitations of ideas, methods, and techniques employed in those attacks, and summarize directions that can be improved to mount collision attacks on SHA-3 that cover more

rounds. Briefly, if the colliding trails that cover more rounds and connecting trails that promise more degree of freedom in constructing connectors are generated, the collision attacks are most likely to be improved. The major challenge lies in the fact that differential trails of $Keccak\text{-}f$ permutation are difficult to search as the large state size results in a search space that is too enormous to be covered effectively. Luckily, we observe that the automatic search tool, *i.e.*, the SAT solver performs extraordinarily well in modeling the differential propagation of $Keccak\text{-}f$. In this work, a powerful SAT-based automatic search toolkit is proposed to overcome the clarified challenges. We demonstrate that the SAT-based trail search methods are applicable to all kind of analytic scenarios where trails are involved. With the SAT-based toolkit, advanced collision attacks on `SHA-3` instances are presented. Totally, a 6-round collision attack on `SHAKE128` of complexity $2^{123.5}$, a 6-round quantum collision attack on `SHA3-256` of complexity $2^{102.65}$, and a 6-round quantum collision attack on `SHA3-224` of complexity $2^{96.15}$ are proposed. It's not only that the 6-round classical and quantum collision attacks are introduced for the first time but also shows that quantum collision attack is able to cover more rounds or targets than classical collision attacks.

References

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

BDPVA13. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 313–314. Springer, 2013.

Ber10. Daniel J Bernstein. Second preimages for 6 (7?(8??)) rounds of keccak. *NIST mailing list*, 2010.

BGLP. Zhenzhen Bao, Jian Guo, Shun Li, and Phuong Pham. Quantum multi-collision distinguishers.

BHT98. Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *Latin American Symposium on Theoretical Informatics*, pages 163–169. Springer, 1998.

BPVA+11. Guido Bertoni, Michaël Peeters, Gilles Van Assche, et al. The keccak reference. 2011.

CKMS14. Donghoon Chang, Arnab Kumar, Pawel Morawiecki, and Somitra Kumar Sanadhya. 1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512. SHA-3 workshop, August 2014.

CNPS17. André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 211–240. Springer, 2017.

DDS12. Itai Dinur, Orr Dunkelman, and Adi Shamir. New attacks on keccak-224 and keccak-256. In *International Workshop on Fast Software Encryption*, pages 442–461. Springer, 2012.

DDS13. Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of sha-3 using generalized internal differentials. In *International Workshop on Fast Software Encryption*, pages 219–240. Springer, 2013.

DDS14. Itai Dinur, Orr Dunkelman, and Adi Shamir. Improved practical attacks on round-reduced keccak. *Journal of cryptology*, 27(2):183–209, 2014.

DSS+20. Xiaoyang Dong, Siwei Sun, Danping Shi, Fei Gao, Xiaoyun Wang, and Lei Hu. Quantum collision attacks on aes-like hashing with low quantum random access memories. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 727–757. Springer, 2020.

DVA12. Joan Daemen and Gilles Van Assche. Differential propagation analysis of keccak. In *International Workshop on Fast Software Encryption*, pages 422–441. Springer, 2012.

Dwo15. Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. 2015.

GJMG11. Bertoni Guido, Daemen Joan, P Michaël, and VA Gilles. Cryptographic sponge functions, 2011.

GLL+20. Jian Guo, Guohong Liao, Guozhen Liu, Meicheng Liu, Kexin Qiao, and Ling Song. Practical collision attacks against round-reduced sha-3. *Journal of Cryptology*, 33(1):228–270, 2020.

GLS16. Jian Guo, Meicheng Liu, and Ling Song. Linear structures: Applications to cryptanalysis of round-reduced KECCAK. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *LNCS*, pages 249–274, 2016.

Gro96. Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

HJ12. Cheng-Shen Han and Jie-Hong Roland Jiang. When boolean satisfiability meets gaussian elimination in a simplex way. In *International Conference on Computer Aided Verification*, pages 410–426. Springer, 2012.

HLY21. Le He, Xiaoen Lin, and Hongbo Yu. Improved preimage attacks on 4-round keccak-224/256. *IACR Trans. Symmetric Cryptol.*, 2021(1):217–238, 2021.

HS20. Akinori Hosoyamada and Yu Sasaki. Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. *Advances in Cryptology–EUROCRYPT 2020*, 12106:249, 2020.

HS21. Akinori Hosoyamada and Yu Sasaki. Quantum collision attacks on reduced sha-256 and sha-512. *IACR Cryptol. ePrint Arch.*, 2021:292, 2021.

LHY21. Xiaoen Lin, Le He, and Hongbo Yu. Improved preimage attacks on 3-round KECCAK-224/256. *IACR Trans. Symmetric Cryptol.*, 2021(3):84–101, 2021.

LQT19. Guozhen Liu, Weidong Qiu, and Yi Tu. New techniques for searching differential trails in keccak. *IACR Transactions on Symmetric Cryptology*, pages 407–437, 2019.

LS19. Ting Li and Yao Sun. Preimage attacks on round-reduced KECCAK-224/256 via an allocating approach. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *LNCS*, pages 556–584. Springer, 2019.

LSLW17. Ting Li, Yao Sun, Maodong Liao, and Dingkang Wang. Preimage attacks on the round-reduced KECCAK with cross-linear structures. *IACR Trans. Symmetric Cryptol.*, 2017(4):39–57, 2017.

MDA17. Silvia Mella, JJC Daemen, and G Van Assche. New techniques for trail bounds and application to differential trails in keccak. 2017.

MP13. Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. Cryptology ePrint Archive, Report 2013/328, 2013. https://ia.cr/2013/328.

MPS13. Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced KECCAK. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *LNCS*, pages 241–262. Springer, 2013.

MS13. Paweł Morawiecki and Marian Srebrny. A sat-based preimage analysis of reduced keccak hash functions. *Information Processing Letters*, 113(10-11):392–397, 2013.

NRM11. María Naya-Plasencia, Andrea Röck, and Willi Meier. Practical analysis of reduced-round KECCAK. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *Progress in Cryptology - INDOCRYPT 2011 - 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011. Proceedings*, volume 7107 of *LNCS*, pages 236–254. Springer, 2011.

QSLG17. Kexin Qiao, Ling Song, Meicheng Liu, and Jian Guo. New collision attacks on round-reduced keccak. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 216–243. Springer, 2017.

QUE19. SEPARATE DECISION QUEUE. Cadical at the sat race 2019. *SAT RACE 2019*, page 8, 2019.

Raj19. Mahesh Sreekumar Rajasree. Cryptanalysis of round-reduced KECCAK using non-linear structures. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, volume 11898 of *LNCS*, pages 175–192. Springer, 2019.

SBH+19. Mate Soos, Armin Biere, M Heule, M Jarvisalo, and M Suda. Cryptominisat 5.6 with yalsat at the sat race 2019. *Proc. of SAT Race*, pages 14–15, 2019.

SDG+20. Mate Soos, Jo Devriendt, Stephan Gocht, Arijit Shaw, and Kuldeep S Meel. Cryptominisat with ccanr at the sat competition 2020. *SAT COMPETITION 2020*, page 27, 2020.

Sin05. Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

SLG17. Ling Song, Guohong Liao, and Jian Guo. Non-full sbox linearization: Applications to collision attacks on round-reduced keccak. In *Annual International Cryptology Conference*, pages 428–451. Springer, Cham, 2017.

SNC09. Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT Solvers to Cryptographic Problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *LNCS*, pages 244–257. Springer, 2009. available via https://github.com/msoos/cryptominisat.

SNC10. Mate Soos, K Nohl, and C Castelluccia. Cryptominisat. *SAT Race solver descriptions*, 2010.

Soo14. Mate Soos. Cryptominisat v4. *SAT Competition*, 23, 2014.

Soo16. Mate Soos. The cryptominisat 5 set of solvers at sat competition 2016. *Proceedings of SAT Competition*, page 28, 2016.

SSK+20. Mate Soos, Bart Selman, Henry Kautz, Jo Devriendt, and Stephan Gocht. Cryptominisat with walksat at the sat competition 2020. *SAT COMPETITION 2020*, page 29, 2020.

SWW18. Ling Sun, Wei Wang, and Meiqin Wang. More accurate differential properties of led64 and midori64. *IACR Transactions on Symmetric Cryptology*, pages 93–123, 2018.

SWW21. Ling Sun, Wei Wang, and Meiqin Wang. Accelerating the search of differential and linear characteristics with the sat method. *IACR Transactions on Symmetric Cryptology*, pages 269–315, 2021.

VOW94. Paul C Van Oorschot and Michael J Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 210–218, 1994.

Supplementary Material

A χ Properties Exploited in Connector Construction

In constructing connectors for collision attacks, systems of linear equations are listed and solved. Several properties of χ operation that are principal for listing equations are introduced in this section.

Property 1. **The inputs of χ operation that follow a compatible differential (α, β) form a subspace.** Assume that $(x_4, x_3, x_2, x_1, x_0)$ are the 5 input bits of the χ operation which is also called Sbox in general, and $(\beta_4, \beta_3, \beta_2, \beta_1, \beta_0)$ (with respect to $(\alpha_4, \alpha_3, \alpha_2, \alpha_1, \alpha_0)$) are the 5 bits of the input (output) difference, based on the definition of χ, the relationship between the differential bits and the input bits can be deduced. As demonstrated in the following expressions, when the input difference α and the compatible output difference β of χ are given, equations on the five input bits x_i where $0 \leq i < 5$ are of at most degree 1, *i.e.*, linear equations.

$$
\begin{aligned}
\beta_0 &= (\boldsymbol{x_1} \cdot \alpha_2 + \alpha_1 \cdot \boldsymbol{x_2} + \alpha_0 + \alpha_2 + \alpha_1 \cdot \alpha_2) \\
\beta_1 &= (\boldsymbol{x_2} \cdot \alpha_3 + \alpha_2 \cdot \boldsymbol{x_3} + \alpha_1 + \alpha_3 + \alpha_2 \cdot \alpha_3) \\
\beta_2 &= (\boldsymbol{x_3} \cdot \alpha_4 + \alpha_3 \cdot \boldsymbol{x_4} + \alpha_2 + \alpha_4 + \alpha_3 \cdot \alpha_4) \\
\beta_3 &= (\boldsymbol{x_4} \cdot \alpha_0 + \alpha_4 \cdot \boldsymbol{x_0} + \alpha_3 + \alpha_0 + \alpha_4 \cdot \alpha_0) \\
\beta_4 &= (\boldsymbol{x_0} \cdot \alpha_1 + \alpha_0 \cdot \boldsymbol{x_1} + \alpha_4 + \alpha_1 + \alpha_0 \cdot \alpha_1)
\end{aligned}
$$

Following the above relations, when the χ-compatible differential (α, β) is determined, the linear equations on input bits x_i are directly listed.

Property 2. **Linearizable affine subspaces.** As proved in [QSLG17, SLG17], when the inputs $x = (x_4, x_3, x_2, x_1, x_0)$ of Sbox are restricted to a small subspace (*e.g.*, a subset of 4 inputs), the outputs $y = (x_4, x_3, x_2, x_1, x_0)$ can be expressed with linear combinations of input bits, *i.e.*, the so-called **Sbox linearization**, while by definition $y_i := x_i \oplus (x_{i+1} \oplus 1) \cdot x_{i+2}$, y_i bits are computed from nonlinear equations on x_i bits. In detail, the linearization of Sbox is discussed in two classifications.

- For active Sboxes, as the inputs are already grouped to subset with the constraints of differentials, some active Sboxes are already fully linearized while the other active Sboxes still need to linearize one bit. More specifically,
  1. for active Sboxes of DDT=2 and DDT=4 entry (*i.e.*, for a certain (α, β), there exist 2 or 4 input pairs that follow the differential), the subset of inputs of the differential form an affine subspace. In other words, any of the five output bits y_i is expressed by a linear expression of input bits x_i. Within the subset restricted by the DDT entry, the nonlinear χ operation is essentially linear.

2. for active Sboxes of DDT=8 entry, 4 out of the 5 output bits of y are naturally linearized, to linearize the only nonlinear y_i bit, extra equations on x_i bits are added.

– For non-active Sboxes, according to distinct attack scenarios, either the whole Sbox or some output bits y_i are linearized. Either case, the linearizations can be achieved by adding additional equations on input bits.

Algorithm 1: Constructing 2-round connector.

Input: 2-round connecting trail, $(\alpha_0, \beta_0, \alpha_1, \beta_1, \alpha_2)$
Output: A system of linear equations on y_0, *i.e.*, E_{y_0}

List and add equations on y_0 for $(c+p)$-bit conditions of α_0 to E_{y_0};
List and add equations on y_0 for conditions of (β_0, α_1) differential to E_{y_0};
while E_{y_0} *is not consistent* **do**
    Select the partially treated β_1 Sboxes with greedy algorithms;
    List equations on y_0 to linearize χ_0 for involved x_1 bits;
    Add the y_0 equations to E_{y_0};
    Store the partially linearized expression of χ_0;
    List and add equations on y_1 for conditions of (β_1, α_2) differential to E_{y_1} ;
    `// According to the partially linearized expression of` χ_0
    Convert E_{y_1} to equations on y_0 and add to E_{y_0};
end

B Trails and Messages

In this section, we give details of differential trails of $Keccak$-f permutation and the message pairs that satisfy the connecotrs of the collision attacks.

The 1600-bit state is displayed as a 5×5 array, ordered from left to right, where '|' acts as the separator; each lane is denoted in hexadecimal using little-endian format; '0' is replaced with '-' for differential trails.

B.1 Differential Trails

In this section, differential trail of 6-round collision attacks are listed. The full 6-round differential trail of the quantum collision attack on `SHA3-256`, *i.e.*, Trail No.1 is given in Table 5. Note that the differential probability of the last round is 2^{-8} as only Sboxes of the digest bits are considered. As collision attacks on the three `SHA-3` instances make use of the same 4-round colliding trail core, the last three round differences are omitted for `SHA3-224` and `SHAKE128`.

Next, two newly searched 4-round trails that are of the best probability are given.

Algorithm 2: Constructing 3-round connector.

Input: 3-round connecting trail, $(\alpha_0, \beta_0, \alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3)$
Output: A system of linear equations on y_1, *i.e.*, E_{y_1}

List and add equations on y_0 for $(c+p)$-bit conditions of α_0 to E_{y_0};
List and add equations on y_0 for conditions of (β_0, α_1) differential to E_{y_0};
while E_{y_0} *is not consistent* **do**
    Fully linearize χ_0 and store the linearized expression of χ_0;
    Add equations on y_0 for linearizing χ_0 to E_{y_0};
    `// According to the linearized expression of` χ_0
    Convert E_{y_0} to E_{y_1};
    while E_{y_1} *is not consistent* **do**
        Select the partially treated β_2 Sboxes with greedy algorithms;
        List equations on y_1 to linearize χ_1 for involved x_2 bits;
        Add the y_1 equations to E_{y_1};
        Store the partially linearized expression of χ_1;
        List and add equations on y_2 for partial conditions of (β_2, α_3) differential to E_{y_2} ;
        `// According to the partially linearized expression of` χ_1
        Convert E_{y_2} to equations on y_1 and add to E_{y_1};
    end
end

B.2 Instances of Solutions of Connectors

In this section, examples of message pairs that follow the connectors of 6-round collision and quantum collision attacks on `SHA3-256`, `SHA3-224` and `SHAKE128` are given respectively.

Table 5: Trail No.1 used in the 6-round quantum collision attack on SHA3-256

β_0	-259C1951A-F934-|595-912-259--9C5|132E481-427751C-|87C8B28--5-32-28|A-8-441881-8922- 26-19181-4-8C---|5C984-32AB-623E5|9-3458--142E914-|-419A51-4359A268|-9D---9C15-6424- F6D3F4913626149-|DA1-3D2227DD818-|22EAF4964678D52-|8282-11-62-42248|3598-896-3-3142- 8731E18-45-B95A8|CBF862287E4183A1|3BCCEC316-3AD728|-27-5419-9--274-|2819-1443E4594-- 3CDB1FBD-E-2B12-|43C3AA3981-46-81|62FE14AC847D612-|3FDB2B85478-7-4-|B--1EA-D958681--	2^{-777}
β_1	-48-8--4-----22-|----4------82D88|1--2---46---1---|---14------4--1-|-82-4-228------- -48-C-12------2-|-------6---8---8|1--24---6----48-|-------4--7--21-|-8--4-248------- -48-8-------12-8|----4--44--8--2-|1--2---9-----4--|--5-4--2---4--1-|-82-4---8------- -48-C--2------2-|-------2-------8|----4---6----4--|--5--------4-21-|-82-4-248-----43 ----8--4------2-|----4------82988|1--2---46---14--|-8--4--4--74-21-|--2---228------3	2^{-264}
β_2	----------17--|----------24--|----------7B--|----------A--|----------C2-- ----------14--|----------42--|----------B1--|----------CC-1|----------82-- ----------73--|----------A4--|----------E1--|----------4B--|----------12-- ----------6--|----------1--|----------E--1|----------8-1|----------7--- ----------5--|----------3A--|----------11--|----------C--|----------F6--	2^{-116}
β_3	----------|----------|----------|----------2-----|---------- ----------|----------|----------|--E----------|---------- ----------|----------2-----|----------|----------A-----|---------- ----------|----------|----------8-----|----------8-----|---------- ----------|--8----------|--2----------|--E----------|----------	2^{-24}
β_4	----------|----------|----------|----1----------|---------- ---2----------|----------|----------|----------|---------- ----------|----------|---14----------|----------|---------- ----------|----------|----------|----------|----E---------- ----------|----7----------|----------|----------|----------	2^{-20}
β_5	----------|----------|----------|----------|2C----------1 ----------9--|----------49|----------|----------|----28--------- ----2----------|--5----------|----------1--|-47----------|---------- ----------248-|----------2-----|----------|8----------|------8--------- ----5----------|------4---------|---------248----|----------|---1C----------	2^{-8}

Table 6: Trail No.2 used in the 6-round quantum collision attack on SHA3-224

β_0	82-3-89-59-2-12-|71C-D431-3--BD84|84418-42A9E11AC-|5-164719-B8524-8|893844-2--8822-- ----4A5-59-2----|F28334B---8-86-4|-5438-C-B-41--4-|5-14-118--281--8|-879-4C3929-3C84 4-333-2-DD83--1-|E19AB522883B96--|857AA1D8A-E-1-1-|-224E11--8-22-88|1A3BB5819321881- C-B-44F1DA47B22-|9657B26--82-A648|81778-68A3E-3-4-|F-B485FB92-1A488|5C292-428-B238C8 9-134ABBC-A3344-|A6AA8C51C8E9AD--|F918C4AEA1E-14C-|E-36791-4-286B4-|3C597-EB42219F8-	2^{-724}
β_1	-48-8--------22-|----4--4---82988|1--2----6---1---|---14--4---4--1-|-82-4-268------- -48-C-12------2-|-------2---82--8|1--24--46-----8-|----------7--21-|-8--4-248----8-- -48-C-------16-8|-------44--82-2-|1--24--9--------|--5----2---4--1-|-82-4---8----8-- -48-C---------2-|-------2----2--8|----4--26-------|--5--------4-21-|-82-4-268----843 ----8---------2-|-------4---81D88|1--24---6-------|-8--4--4--74-21-|--2---268------3	2^{-268}
β_2	----------13--|----------24--|----------7B--|----------A--|----------C2-- ----------14--|----------42--|----------B1--|----------CC-1|----------8--- ----------33--|----------A4--|----------E1--|----------4B--|----------12-- ----------6--|----------1--|----------E--1|----------8-1|----------7--- ----------4--|----------3A--|----------11--|----------C--|----------F6--	2^{-113}

Table 7: Trail No.3 used in the 6-round collision attack on `SHAKE128`

β_0	--49--151E-E822-|DA8-912-218--845|-2EE-89-44775-A-|1418738--3-E3--8|334-C418851-81E- 26-1--81-3-4C--1|58-811A2BD-E-185|9-A4881--82AD12-|-41-A59-56558249|-1C851-C18-22-2- 14-1C11-1226--9-|C88811A-2-8C8-88|1-2--8-3-171D-2-|--5AA481462C3348|27--5-84---8-52- -425218-1--A-1--|C888823821--9-A1|1--4C82-167BC1--|--9-14195D-83248|23C82-44-1-4152- 53C99221-C--A-8-|-96223988-CC--81|55C6-CA--33F5-2-|-49-B31845--51C-|A12-61341244812-	2^{-670}
β_1	-48-8--4-----22-|----4------82D88|1--2---46---1---|---14------4--1-|-82-4-228------- -48-C-12------2-|-------6---8---8|1--24---6----48-|-------4--7--21-|-8--4-248------- -48-8-------12-8|----4--44--8--2-|1--2---9-----4--|--5-4--2---4--1-|-82-4---8------- -48-C--2------2-|-------2-------8|----4---6----4--|--5--------4-21-|-82-4-248-----43 ----8--4------2-|----4------82988|1--2---46---14--|-8--4--4--74-21-|--2---228------3	2^{-264}
β_2	------------17--|------------24--|------------7B--|-------------A--|------------C2-- ------------14--|------------42--|------------B1--|------------CC-1|------------82-- ------------73--|------------A4--|------------E1--|------------4B--|------------12-- -------------6--|-------------1--|------------E--1|-------------8-1|------------7--- -------------5--|------------3A--|------------11--|-------------C--|------------F6--	2^{-116}

Table 8: 4-round differential trails of $Keccak\text{-}f[1600]$

4-Round Trail No. 1, with weight of 133		
β_0	----------4|----------|----------|----------2|----------8 ----------4|----------|----------|----------2|----------8 ----------4|----------|----------|----------2|----------8 ----------4|----------|1---------|----------2|---------- ----------4|----------|1---------2|----------|----------8	2^{-32}
β_1	----------|----------|----------|----------|---------- ----------|----------|----------|----------|-2-------- ----------|----------|----------|----------8--|---------- ----------|----------|----------8--|----------8--|-2-------- ----------|----------|----------|----------|----------	2^{-9}
β_2	----------|----------|----------|------1---------|---------- ----------|----------2---|----------|----------|---------- ----------|----------|------1---------|----------2|---------- ----------|----------|----------|----------|---------- ----------|----------|----------|----------|----------	2^{-8}
β_3	----------4---|-2---------2----|-1--2-------9---|-2----------|----4-------8--- 1----------1|--1-------2-----|----------2----|----------4----|8-------4----4-- ------4---------|------8----8-1--|2--------4------|-----1-------2--|-------1-------- -8------1-------|---4----------|----8----------|---1----1--2----|--------1------- --------8----8-1|---------8------|-----1--------8-|--8----------|------8---------	2^{-84}

4-Round Trail No. 2, with weight of 133		
β_0	4----------|1----------|----------|1----------|2---------- ----------|3----------|2----------|4----------|9---------- 1----------|----------|1----------|6----------|A---------- 1----------|----------|----------|7----------|A---------- 1----------|2----------|3----------|4----------|8----------	2^{-47}
β_1	4----------|----------|----------|----------|---------- --------4------|----------|----------|----------|---------- 4----------|----------|----------|----------|---------- ---------4------|----------|----------|----------|---------- ----------|----------|----------|----------|----------	2^{-8}
β_2	4----------|----------|----------|----------|---------- ----------|----------|----------2|----------|---------- ----------|----------|----------|----------|---------- ----------|4----------|----------|----------|---------- ----------|----------|----------|----------8|----------	2^{-8}
β_3	C----------|----4----------|----82---------|----------4-----|----------2---- --------2-------|----------8-----|----------4|----88---------|-8----------2 ----------8|----------49-|---------4------|----------8--|----------2---- --------4-------|-------8--------|----------1---|----------82---|-A---------- 1----------4|-1----------|-----4---------|-----18--------|----------1-	2^{-70}

Table 9: 4-round differential trail of $Keccak\text{-}f[800]$

β_0	4-------|24------|--------|--------|-------- 4-------|--------|2-------|4-------|-------- 4-------|--------|2-------|--------|-------- --------|-4------|2-------|--------|-------- 4-------|--------|2-------|--------|--------	2^{-22}
β_1	--------|--------|--------|--------|-------- --------|--------|--------|------8-|-------- -8------|--------|--------|------4-|-------- --------|--------|--------|--------|-------- -8------|--------|--------|------C-|--------	2^{-12}
β_2	--------|--------|--------|--------|-------- --------|--------|4-------|--------|-------- --------|--------|8-------|--------|----2--- --------|--------|--------|--------|C------- --------|4-------|--------|--------|--------	2^{-12}
β_3	----2---|--------|-----6--|--2----8|----1--- 1----4--|--------|---1----|--------|-8------ --------|--------|-2----8-|------C-|8------- --------|---2----|--------|----4---|-1----4- 1-------|--8---2-|--1-----|--4-----|-------1	2^{-49}

Table 10: Message pairs that follow the 3-round connector of SHAKE128

M_1	7177E1DAC7F3DCAF|7FE004EEE547BE60|86632CAD7A156312|C77DE0993E8503D1|9C1999581C84789A F5D9FCCA0B11B895|CF5F13219A51703F|4D59EF658AE05CB0|7E6360300015E0CD|CA68151E871DE340 51F55064B2A1DDB3|BA8F0C027A5DFA4A|10C31EC601E06620|2B4745C447B6A99A|7450C64A9C16C0EE 7EF18823DDEC3257|31FDCE0100143597|4F28C1C60406BF14|3FABF55262DC1B14|3BE6E710AC00CFE2 FE199297DE836625|0000000000000000|0000000000000000|0000000000000000|0000000000000000
M_2	7168266633A8C91F|5DA8FC2BE8C59E60|C707EC498A06E013|3F11A48A5A802BE2|9AD9745B88425CDA 779F3269FDCCA726|F515136C1A9DF87E|60394743CA736FB2|720415309C01C2FC|2DF08D9D1665DCD2 43B706CF59F11007|D2C3157D76C944F2|0C89184CD5F13AE1|5B8F11CD53A78839|33D675E9CCDF58B5 F285CFA2D7BBED9E|353764E016CD355A|25FA4886EC250404|860BF1F3A5553B2F|746670B73D24F793 FF075CFE20F7F1B7|0000000000000000|0000000000000000|0000000000000000|0000000000000000

Table 11: Message pairs that follow the 2-round connector of SHA3-256

M_1	9BE7DEE6B6466BFA|655978BF618EB06B|0751B61615B0C99D|977D97FDF64DE550|754D6F03F1616012 42E520398CA4FCCF|F42227A8DB14CD98|30392DB5AC74056F|08E94682644AA93E|B8227616B1E07EF0 4C7BB462C374409A|2E7C77A9473A09BE|662FC2C39C847BDF|A1CF8BCBA082588B|6D00BE23B7022FA9 3231FB3409F11485|E7B6B3F16F21ECA6|0000000000000000|0000000000000000|0000000000000000 0000000000000000|0000000000000000|0000000000000000|0000000000000000|0000000000000000
M_2	5CB61215AE6C787A|3230F8749FFC1A63|BABDCC0285881F1D|CEF0D3C1D3C0FD68|9CBD5812073F5438 00093565923EFA2D|CA20042B2220F8D1|7DD169A18CD96169|96BC71BE291FAB91|929EA1DE25815062 9B753204C3D41272|C872FE112E03A74B|2AA788C90CB40C16|84A45FE4906C1380|7C8AC6E0E7A5D7A2 359A366A2BCB79D0|E334D3BF19006C2B|0000000000000000|0000000000000000|0000000000000000 0000000000000000|0000000000000000|0000000000000000|0000000000000000|0000000000000000

Table 12: Message pairs that follow the 2-round connector of SHA3-224

M_1	DC7B6C706F133812|B50C7222F3A9A9B0|4E81751A4654512D|5C9ED7FA0FC92A8C|58E2DC5C755F30A0 6A16CB699BAB9455|0546AF387CEF0131|2CC41EAFF4F69865|7A8DAE1C8448704A|09E8F2DF73EBBD0D B12136F52C3FD4B6|F9BB0902E72FE733|94792317346855F1|033A58E698652530|E2008610306D6862 21D61434E90B327B|4E1EA3E780B0C1D2|E32D8E608E222478|0000000000000000|0000000000000000 0000000000000000|0000000000000000|0000000000000000|0000000000000000|0000000000000000
M_2	3C7C626EDB715DFA|9A03B6084DE04E58|4D0479E9D059640F|E59E957ABDF547D5|9816BCA5735FACBB 0890C783FFAE8BBB|4940C30974207ADC|6C8A5267EA57921D|06B54F0DE52AB95B|898A991650A8BF25 D38D4063D7579076|E68D08D82D8F78DF|DFE43928F8F939E1|5A121224970D013C|374162D8A1975BCB CBF2068FA41B4D8F|49C8FE9DD07A2692|E9F449151154E45B|0000000000000000|0000000000000000 0000000000000000|0000000000000000|0000000000000000|0000000000000000|0000000000000000