IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 0, No. 0, pp. 0–0. DOI:10.13154/tosc.v0.i0.0-0

Improved MITM Cryptanalysis on Streebog

Jialiang Hua[1], Xiaoyang Dong[1] (✉), Siwei Sun[2,5], Zhiyu Zhang[3,4,2], Lei Hu[3,4,2] and Xiaoyun Wang[1,6,7]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
{huajl18,xiaoyangdong,xiaoyunwang}@tsinghua.edu.cn
[2] School of Cryptology, University of Chinese Academy of Sciences, Beijing, China
sunsiwei@ucas.ac.cn
[3] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{zhangzhiyu,hulei}@iie.ac.cn
[4] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[5] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[6] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[7] School of Cyber Science and Technology, Shandong University, Qingdao, China

Abstract. At ASIACRYPT 2012, Sasaki et al. introduced the *guess-and-determine* approach to extend the meet-in-the-middle (MITM) preimage attack. At CRYPTO 2021, Dong et al. proposed a technique to derive the solution spaces of *nonlinear constrained neutral words* in the MITM preimage attack. In this paper, we try to combine these two techniques to further improve the MITM preimage attacks. Based on the previous MILP-based automatic tools for MITM attacks, we introduce new constraints due to the combination of *guess-and-determine* and *nonlinearly constrained neutral words* to build a new automatic model.
As a proof of work, we apply it to the Russian national standard hash function `Streebog`, which is also an ISO standard. We find the first 8.5-round preimage attack on `Streebog-512` compression function and the first 7.5-round preimage attack on `Streebog-256` compression function. In addition, we give the 8.5-round preimage attack on `Streebog-512` hash function. Our attacks extend the best previous attacks by one round. We also improve the time complexity of the 7.5-round preimage attack on `Streebog-512` hash function and 6.5-round preimage attack on `Streebog-256` hash function.

Keywords: Preimage · MITM Attack · `Streebog` · MILP

1 Introduction

The cryptographic hash function is one of the fundamental building blocks in modern cryptography. It is a mathematical algorithm that takes a message of arbitrary length and outputs a bit string of fixed length. Hash functions play important roles in modern cryptography and have been used in many important applications, such as authentication, digital signatures, and message integrity. For hash functions, collision resistance, preimage resistance and second-preimage resistance form the three main security requirements.

The Meet-in-the-Middle (MITM) approach was first introduced by Diffie and Hellman [DH77] in 1977 to attack `DES`. The MITM attack has always received the attention it deserves in a key-recovery scenario, but it has only more recently been applied to preimage attacks [AS09b, AMM09, SA08]. Since then, many MITM preimage attacks on kinds of hash functions or their round-reduced variants have been proposed, including MD4 [GLRW10], MD5 [SA09], Tiger [GLRW10, WS10], SHA-0 [AS09a], SHA-1 [AS09a, EFK15, KK12], SHA-2 [AGM+09], HAVAL [SA08, GSY15], BLAKE [EFK15], RIPEMD [WSK+11], HAS-160 [HKS10], Streebog [AY14, MLHL15b, ZWW13, MLHL14], Whirlpool [SWWW12], Grøstl [WFW+12], and AES hashing modes [Sas11, WFW+12, BDG+19, BDG+21]. Meanwhile, many techniques are proposed to enhance and improve the MITM attacks on hash functions, such as *splice-and-cut* [AS09b], *initial structure* [SA09], *(indirect-)partial matching* [AS09b, SA09], *biclique* [BKR11], *sieve-in-the-middle* [CNV13], and *match-box* [FM15]. The core of a MITM preimage attack on a hash function is generally a MITM preimage attack on its compression function. In the attack, the compression function is divided into two sub-functions so that a portion of bits of the input message only affect one sub-function and another portion affects the other sub-function, which allows attackers to mount the MITM attacks. The subfunction computed forward is named forward chunk and the subfunction computed backward is named backward chunk. The bits affecting only one chunk are called neutral words. At EUROCRYPT 2021, Bao et al. [BDG+21] built an MILP-based automatic tool of MITM preimage attack and applied it to AES hashing modes and Haraka v2 [KLMR16]. Later on, Bao et al. [BGST21] improved the model by introducing the technique of guess-and-determine and applied it to Whirlpool and Grøstl. At CRYPTO 2021, Dong et al. [DHS+21] extended the automatic model into MITM key-recovery attacks and collision attacks. In 2022, Schrottenloher and Stevens [SS22] studied a simpler MILP modeling which allows to find both classical and quantum attacks on a broad class of cryptographic permutations. Besides, another automatic tool was introduced by Derbez and Fouque [DF16] for MITM and DS-MITM attacks [DS08, DKS10, DFJ13, DF16] on block ciphers. The tool is not based on MILP and wasn't used to attack hash functions.

Streebog [ISO18] is a cryptographic hash function defined in Russian national standard GOST R 34.11-2012 [GOS12]. It was created to replace the old GOST R 34.11-94 hash function [GOSan] which was theoretically broken in 2008 [MPR08a, MPR+08b]. The hash function is widely used in Russia, and it is also included as RFC 6896 [DD13] by IETF and standardized by ISO/IEC 10118-3:2018 [ISO18]. Streebog is an iterated hash function based on HAIFA framework [BD07] as a domain extension algorithm. It consists of two members: Streebog-256 and Streebog-512 which output 256-bit and 512-bit hash digest respectively. Streebog-256 uses a different initial state than Streebog-512, and truncates the output hash, but is otherwise identical. The compression function operates in Miyaguchi-Preneel (MP) mode with an AES-like block cipher, the internal state is represented as an 8×8 matrix of bytes and it is updated 12 times with the round function, followed by an XOR operation with a whitening key. In the past few years, several cryptanalysis results on Streebog have been reported, including preimage attacks, second preimage attacks, and collision attacks. Wang et al. [WYW13] focused on the compression function and they gave collision attacks on 4.5, 5.5, 7.5, and 9.5 rounds compression function of Streebog by using the rebound attack [MRST09]. In 2013, Zou et al. [ZWW13] presented collision attacks on 5-round Streebog-256 and Streebog-512 hash function with the Super-Sbox technique [GP10, LMR+09] and the multi-collision technique [Jou04]. Additionally, they constructed a preimage attack on 6-round Streebog-512 hash function by combining the guess-and-determine MITM attack [SWWW12] with multi-collision. At AFRICACRYPT 2014, AlTawy and Youssef [AY14] also proposed a preimage attack on 6-round Streebog-512. At ACNS 2014, Ma et al. [MLHL14] improved the preimage attacks on 6-round Streebog-512 hash function, and they presented collision attacks on 6.5-round Streebog-256 and 7.5-round Streebog-512. In addition, they constructed a distinguisher on 9.5-round Streebog using the limited-birthday distinguisher [IPS13]. At SAC 2014, Guo et al. [GJL+14] exploited the misuse of the counter in the HAIFA mode of Streebog and presented generic second preimage attacks on the full Streebog-512 hash function. At IWSEC 2015, Ma et al. [MLHL15b] proposed a 6.5-round preimage attack on `Streebog-256` and a 7.5-round preimage attack on `Streebog-512`. At EUROCRYPT 2016, Biryukov et al. [BPU16] reverse-engineered the S-Box of `Streebog` and recovered two completely different decompositions of the S-Box. At FSE 2019, Perrin [Per19] identified a third decomposition of the S-Box and exposed a very strong algebraic structure.

Related Works. At ASIACRYPT 2012, Sasaki et al. [SWWW12] introduced the guess-and-determine technique to improve the MITM preimage attack on `Whirlpool`. Since then, this technique has been applied to many hash functions, such as `Grøstl` [WFW+12], `Streebog` [AY14, MLHL15b, MLHL14], `Whirlwind` [MLHL15a], etc. At EUROCRYPT 2021, Bao et al. [BDG+21] built an MILP-based automatic tool of MITM preimage attack. Later, Bao et al. [BGST21] proposed an improved automatic model for MITM preimage attack, which takes the guess-and-determine technique into consideration. At CRYPTO 2021, Dong et al. [DHS+21] discovered the neutral words can be nonlinearly constrained, while the previous MITM attacks [Sas11, SWWW12] usually adopt linearly constrained neutral words, and their solution spaces are calculated by solving these linear equations. When the neutral words are nonlinearly constrained, one may have to calculate the solution spaces for the neutral words by solving a higher-order equation system, which is usually hard. To deal with the problem, Dong et al. [DHS+21] proposed a table-based technique to precompute the solution spaces before the MITM process instead of solving a nonlinear equation system directly. Finally, they succeeded in extending the *initial structure* and then the total number of rounds covered by the MITM approach. However, in Dong et al.'s [DHS+21] MITM attack framework, the guess-and-determine technique is missing.

Table 1: Summary of preimage attack results on `Streebog`

Algorithm	Target	Rounds	Time	Memory	Ref.
`Streebog-256` (12 rounds)	Compression Function	6.5	2^{232}	2^{120}	[MLHL15b]
		6.5	2^{209}	2^{160}	Sect. 7
		7.5	2^{209}	2^{192}	Sect. 5.3
	Hash Function	5	2^{192}	2^{64}	[MLHL15b]
		5	2^{208}	2^{12}	[MLHL15b]
		6.5	2^{232}	2^{120}	[MLHL15b]
		6.5	2^{209}	2^{160}	Sect. 7
`Streebog-512` (12 rounds)	Compression Function	6	2^{496}	2^{64}	[ZWW13]
		6	2^{496}	2^{112}	[AY14]
		7.5	2^{496}	2^{64}	[MLHL15b]
		7.5	2^{441}	2^{192}	Sect. A
		8.5	2^{481}	2^{288}	Sect. 5.2
	Hash Function	6	2^{505}	2^{64}	[ZWW13]
		6	2^{505}	2^{256}	[AY14]
		6	2^{496}	2^{64}	[MLHL14]
		6	2^{504}	2^{11}	[MLHL14]
		7.5	2^{496}	2^{64}	[MLHL15b]
		7.5	2^{504}	2^{11}	[MLHL15b]
		7.5	$2^{478.25}$	2^{256}	Sect. 6
		8.5	$2^{498.25}$	2^{288}	Sect. 6

Our Contributions. As shown in [DHS+21], nonlinearly constrained neutral words extend the initial structure a lot, and then extend the whole MITM attack. In fact, nonlinearly constrained neutral words describes a new way to build initial structure. When putting this technique into the MILP model, it will cover more possible MITM trails that may lead to better attacks. Therefore, it is very meaningful to study the situation where the neutral words are nonlinearly constrained in MITM attacks. In this paper, we propose a new MITM preimage attack model by combining Sasaki et al.'s guess-and-determine technique [SWWW12] and Dong et al.'s [DHS+21] nonlinearly constrained neutral words. In addition, based on previous automatic tools [Sas18, BDG+21, DHS+21, BGST21] for MITM attacks, we introduce a new automatic model to search optimal parameters for the updated MITM attack. As a proof of work, we apply the new techniques to `Streebog-256` and `Streebog-512` hash functions. Finally, we find an 8.5-round preimage attack on `Streebog-512`'s compression function and a 7.5-round preimage attack on `Streebog-256`'s compression function. Then, we give a preimage attack on 8.5-round `Streebog-512` hash function with a method proposed by AlTawy et al. [AY14] to convert the preimage attack on compression function to hash function. In addition, we also improve the 7.5-round preimage attack on `Streebog-512` and 6.5-round preimage attack on `Streebog-256`. The summary of preimage attacks on `Streebog` is shown in Table 1.

2 Definitions and Notations

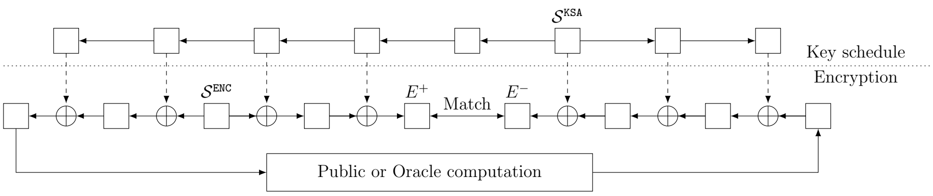


Figure 1: A high-level overview of the MITM attacks [DHS+21]

At CRYPTO 2021, Dong et al. [DHS+21] described the MITM attacks in a unified way as MITM attacks on the so-called closed computation path. The high-level overview of the MITM attacks is shown in Figure 1. We list the notations below.

- $\mathcal{S}^{\mathtt{ENC}}$: starting state in the encryption data path (contains n w-bit cells)
- $\mathcal{S}^{\mathtt{KSA}}$: starting state in the key schedule data path (contains $\bar{n}$ w-bit cells)
- E^+/E^-: ending state of the forward/backward computation
- $\mathcal{B}^{\mathtt{ENC}}/\mathcal{B}^{\mathtt{KSA}}$: subset of $\mathcal{N}=\{0,1,\cdots,n-1\}/\overline{\mathcal{N}}=\{0,1,\cdots,\bar{n}-1\}$, index of Blue cells in $\mathcal{S}^{\mathtt{ENC}}/\mathcal{S}^{\mathtt{KSA}}$
- $\mathcal{R}^{\mathtt{ENC}}/\mathcal{R}^{\mathtt{KSA}}$: subset of $\mathcal{N}/\overline{\mathcal{N}}$, index of Red cells in $\mathcal{S}^{\mathtt{ENC}}/\mathcal{S}^{\mathtt{KSA}}$
- $\mathcal{G}^{\mathtt{ENC}}/\mathcal{G}^{\mathtt{KSA}}$: subset of $\mathcal{N}/\overline{\mathcal{N}}$, index of Gray cells in $\mathcal{S}^{\mathtt{ENC}}/\mathcal{S}^{\mathtt{KSA}}$
- $\mathcal{M}^+/\mathcal{M}^-$: subset of $\mathcal{N}$, index of cells that can be computed in E^+/E^-
- λ^+: $\lambda^+ = \mid \mathcal{B}^{\mathtt{ENC}} \mid + \mid \mathcal{B}^{\mathtt{KSA}} \mid$, the initial degrees of freedom for the forward chunk
- λ^-: $\lambda^- = \mid \mathcal{R}^{\mathtt{ENC}} \mid + \mid \mathcal{R}^{\mathtt{KSA}} \mid$, the initial degrees of freedom for the backward chunk
- DoM: the degrees of matching
- f_i^+: a function that maps $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ to a word
- f_i^-: a function that maps $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$ to a word
- $\boldsymbol{f}^+$: $\boldsymbol{f}^+ = (f_1^+, \cdots, f_{l^+}^+)$, l^+ constraints on the neutral words for the forward chunk
- $\boldsymbol{f}^-$: $\boldsymbol{f}^- = (f_1^-, \cdots, f_{l^-}^-)$, l^- constraints on the neutral words for the backward chunk
- DoF$^+$: DoF$^+ = \lambda^+ - l^+$, the degrees of freedom for the forward chunk
- DoF$^-$: DoF$^- = \lambda^- - l^-$, the degrees of freedom for the backward chunk

From $(\mathcal{S}^{\mathtt{ENC}}, \mathcal{S}^{\mathtt{KSA}})$ leading to E^+ is the forward computation and from $(\mathcal{S}^{\mathtt{ENC}}, \mathcal{S}^{\mathtt{KSA}})$ leading to E^- is the backward computation. The cells of $(\mathcal{S}^{\mathtt{ENC}}, \mathcal{S}^{\mathtt{KSA}})$ are partitioned into different subsets with different meanings which satisfy $\mathcal{B}^{\mathtt{ENC}} \cap \mathcal{R}^{\mathtt{ENC}} = \emptyset$, $\mathcal{B}^{\mathtt{KSA}} \cap \mathcal{R}^{\mathtt{KSA}} = \emptyset$,

$\mathcal{G}^{\texttt{ENC}} = \mathcal{N} - \mathcal{B}^{\texttt{ENC}} \cup \mathcal{R}^{\texttt{ENC}}$ and $\mathcal{G}^{\texttt{KSA}} = \overline{\mathcal{N}} - \mathcal{B}^{\texttt{KSA}} \cup \mathcal{R}^{\texttt{KSA}}$. A coloring system is introduced to visualize these subsets and the attack. The cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$, which are visualized by ■ cells, are the neutral words for the forward computation. The cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$, which are visualized by ■ cells, are the neutral words for the backward computation. λ^+ and λ^- are the number of ■ and ■ cells in the starting states. The cells $\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}]$ and $\mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]$ are visualized as ■ cells. The matching is between E^+ and E^-, DoM $= m$ if $E^+[\mathcal{M}^+]$ and $E^-[\mathcal{M}^-]$ form an m-cell filter.

Besides, the values of l^+ functions $\boldsymbol{f}^+ = (f_1^+, \cdots, f_{l^+}^+)$ can be computed with the knowledge of the ■ cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ and ■ cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$. The values of l^- functions $\boldsymbol{f}^- = (f_1^-, \cdots, f_{l^-}^-)$ can be computed with the knowledge of the ■ cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ and ■ cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$. If the cells $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ are fixed to an arbitrary constant, and for any fixed $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w \cdot l^+}$ and $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w \cdot l^-}$, the neutral words for the forward computation and backward computation paths fulfill the following systems of equations:

$$\begin{cases} f_1^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_1 \\ f_2^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_2 \\ \qquad\qquad \cdots \quad \cdots \\ f_{l^+}^+(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) = a_{l^+} \end{cases} \tag{1}$$

$$\begin{cases} f_1^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_1 \\ f_2^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_2 \\ \qquad\qquad \cdots \quad \cdots \\ f_{l^-}^-(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) = b_{l^-} \end{cases} \tag{2}$$

The computations for deriving $E^+[\mathcal{M}^+]$ and $E^-[\mathcal{M}^-]$ can be carried out independently. Usually, Equation (1) and (2) are linear equations (i.e., the neutral words are linearly constrained) in previous MITM preimage attacks [Sas11, SWWW12]. Therefore, the attackers can solve the linear equations to derive the solution spaces for the neutral words with ease. However, Dong et al. [DHS+21] discovered that Equation (1) and (2) can be nonlinear equations (i.e., the neutral words are nonlinearly constrained). They also invented a table-based method to efficiently solve the solution spaces for the nonlinearly constrained neutral words.

For any given $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ and $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+})$ or $\mathfrak{c}^- = (b_1, \cdots, b_{l^-})$, $\mathbb{B}(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$ and $\mathbb{R}(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^-)$ denote the solution spaces of $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ and $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ induced by Equation (1) and (2). If there are $2^{w \cdot (\lambda^+ - l^+)}$ and $2^{w \cdot (\lambda^- - l^-)}$ solutions of Equation (1) and (2) respectively, then $\text{DoF}^+ = \lambda^+ - l^+$ and $\text{DoF}^- = \lambda^- - l^-$ are the *degrees of freedom* for the forward and backward computations. In addition, if $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}])$ is fixed to a constant α, $\mathfrak{c}^+$ and $\mathfrak{c}^-$ are fixed to some contants. We can compute $E^+[\mathcal{M}^+]$ for all $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in \mathbb{B}(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^+)$ and store it in a table L. We also can compute $E^-[\mathcal{M}^-]$ for all $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) \in \mathbb{B}(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}], \mathfrak{c}^-)$, then we can test for full matching between $E^-[\mathcal{M}^-]$ and $E^+[\mathcal{M}^+]$. For different α, $\mathfrak{c}^+$ and $\mathfrak{c}^-$, the above process can be repeated many times and each time is called one MITM *episode*.

2.1 MITM Attack with Guess-and-Determine and Linearly Constrained Neutral Words

The guess-and-determine approach was introduced by Sasaki et al. [SWWW12] to extend the MITM preimage attack on `Whirlpool`. In their attack, some cells may be guessed to be Blue/Red in different states in the forward/backward computation. To explain, we introduce some new notations:

- $\mathcal{Y}_+^{\texttt{ENC}}/\mathcal{Y}_+^{\texttt{KSA}}$: the set of cells guessed to be Blue for the encryption/key schedule path

- $\mathcal{Y}_-^{\texttt{ENC}}/\mathcal{Y}_-^{\texttt{KSA}}$: the set of cells guessed to be Red for the encryption/key schedule path
- σ^+: $\sigma^+ = |\mathcal{Y}_+^{\texttt{ENC}}| + |\mathcal{Y}_+^{\texttt{KSA}}|$, the number of cells guessed to be Blue
- σ^-: $\sigma^- = |\mathcal{Y}_-^{\texttt{ENC}}| + |\mathcal{Y}_-^{\texttt{KSA}}|$, the number of cells guessed to be Red

In Sasaki et al.'s attack [SWWW12], the neutral words are linearly constrained, i.e., Equation (1) and (2) are linear, so the solution spaces of the neutral words can be easily obtained. Their MITM preimage attack is shown in Algorithm 1.

Algorithm 1: Sasaki et al.'s MITM preimage attack with guess-and-determine

Input: None
Output: Preimage X

1 **for** $(\mathcal{S}^{\texttt{ENC}}[\mathcal{G}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{G}^{\texttt{KSA}}]) \in \mathbb{G} \subseteq \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\texttt{ENC}}|+|\mathcal{G}^{\texttt{KSA}}|)}$ **do**
2   **for** $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathcal{H}_1 \subseteq \mathbb{F}_2^{w\cdot l^+}$ **do**
3   **for** $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathcal{H}_2 \subseteq \mathbb{F}_2^{w\cdot l^-}$ **do**
4   Get the solution of $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}])$ by solving the Equation (1) and store the values in a table T_1.
5   Get the solution of $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ by solving the Equation (2) and store the values in a table T_2.
6   **for** $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}]) \in T_1$ *and* $(\mathcal{Y}_+^{\texttt{ENC}}, \mathcal{Y}_+^{\texttt{KSA}}) \in \mathbb{F}_2^{w\cdot\sigma^+}$ **do**
7   Compute $E^+[\mathcal{M}^+]$ along the forward computation path.
8   Insert $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}], \mathcal{Y}_+^{\texttt{ENC}}, \mathcal{Y}_+^{\texttt{KSA}})$ into L indexed by $E^+[\mathcal{M}^+]$.
9   **for** $(\mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{KSA}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}]) \in T_2$ *and* $(\mathcal{Y}_-^{\texttt{ENC}}, \mathcal{Y}_-^{\texttt{KSA}}) \in \mathbb{F}_2^{w\cdot\sigma^-}$ **do**
10   Compute $E^-[\mathcal{M}^-]$ along the backward computation path.
11   **for** $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}], \mathcal{Y}_+^{\texttt{ENC}}, \mathcal{Y}_+^{\texttt{KSA}}) \in L[E^-[\mathcal{M}^-]]$ **do**
12   Use $(\mathcal{S}^{\texttt{ENC}}[\mathcal{B}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{B}^{\texttt{KSA}}], \mathcal{S}^{\texttt{ENC}}[\mathcal{R}^{\texttt{ENC}}], \mathcal{S}^{\texttt{KSA}}[\mathcal{R}^{\texttt{KSA}}])$ to compute and check the guessed values.
13   **if** *The values* $(\mathcal{Y}_+^{\texttt{ENC}}, \mathcal{Y}_+^{\texttt{KSA}}, \mathcal{Y}_-^{\texttt{ENC}}, \mathcal{Y}_-^{\texttt{KSA}})$ *are correct* **then**
14   Reconstruct the (candidate) message X.
15   **if** X *is a preimage* **then**
16   Output X and Stop.

Complexity. From Line 6 to Line 16 of Algorithm 1, we test $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^- + \sigma^+ + \sigma^-)}$ messages and expect $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^- + \sigma^+ + \sigma^- - m)}$ of them to pass the m-cell filter. We need to verify the correctness of these partial matchings. In Line 13, the probability that the guessed cells in the forward and backward computations are correct is $2^{-w\cdot(\sigma^+ + \sigma^-)}$. Hence, there will be $2^{w\cdot(\mathrm{DoF}^+ + \mathrm{DoF}^- - m)}$ valid partial matchings that pass the check of Line 13. Suppose we are finding a preimage of the h-cell target, the overall time complexity is

$$(2^w)^{(h-(\mathrm{DoF}^+ + \mathrm{DoF}^-))}((2^w)^{\mathrm{DoF}^+ + \sigma^+} + (2^w)^{\mathrm{DoF}^- + \sigma^-} + (2^w)^{\mathrm{DoF}^+ + \mathrm{DoF}^- + \sigma^+ + \sigma^- - m}) \approx (2^w)^{h - min(\mathrm{DoF}^- - \sigma^+,\ \mathrm{DoF}^+ - \sigma^-,\ m - \sigma^+ - \sigma^-)}. \quad (3)$$

In the attack, we need to store the tables T_1, T_2 and L, so the memory complexity is

$$2^{\mathrm{DoF}^+} + 2^{\mathrm{DoF}^-} + 2^{min(\mathrm{DoF}^+ + \sigma^+, \mathrm{DoF}^- + \sigma^-)}.$$

3 MITM Attack with Guess-and-Determine and Nonlinearly Constrained Neutral Words

If the neutral words are nonlinearly constrained, i.e., Equation (1) and (2) are nonlinear, it will be difficult to get the solution spaces of the neutral words by solving the nonlinear equations directly. At CRYPTO 2021, Dong et al. [DHS+21] introduced a table-based method to compute the solution spaces of the neutral words. However, Dong et al. did not consider the case when the guess-and-determine is included in the MITM attack. In this section, we propose a unified MITM model combining nonlinearly constrained neutral words and guess-and-determine.

Since the guess-and-determine is introduced in the MITM attacks, the guessed cells may be involved in the l^+ functions $\boldsymbol{f}^+ = (f_1^+, \cdots, f_{l^+}^+)$ of Equation (1). In order to compute their values, we need to know not only the values of ■ cells and ■ cells in the starting states, but also the values of the guessed cells in the computation path. So we define the l^+ functions by

$$f_i^+ : \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\mathtt{ENC}}|+|\mathcal{G}^{\mathtt{KSA}}|+|\mathcal{B}^{\mathtt{ENC}}|+|\mathcal{B}^{\mathtt{KSA}}|+|\mathcal{Y}_+^{\mathtt{ENC}}|+|\mathcal{Y}_+^{\mathtt{KSA}}|)} \to \mathbb{F}_2^w.$$

Similarly, we define the l^- functions $\boldsymbol{f}^- = (f_1^-, \cdots, f_{l^-}^-)$ by

$$f_i^- : \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\mathtt{ENC}}|+|\mathcal{G}^{\mathtt{KSA}}|+|\mathcal{R}^{\mathtt{ENC}}|+|\mathcal{R}^{\mathtt{KSA}}|+|\mathcal{Y}_-^{\mathtt{ENC}}|+|\mathcal{Y}_-^{\mathtt{KSA}}|)} \to \mathbb{F}_2^w.$$

Therefore, if the cells $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$ are fixed, for any fixed $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w\cdot l^+}$ and $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w\cdot l^-}$, the neutral words for the forward computation and backward computation are constrained by the following systems of equations:

$$\begin{cases} f_1^+(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}) = a_1 \\ f_2^+(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}) = a_2 \\ \qquad \cdots \quad \cdots \\ f_{l^+}^+(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}) = a_{l^+} \end{cases} \tag{4}$$

$$\begin{cases} f_1^-(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}], \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}}) = b_1 \\ f_2^-(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}], \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}}) = b_2 \\ \qquad \cdots \quad \cdots \\ f_{l^-}^-(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}], \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}}) = b_{l^-} \end{cases} \tag{5}$$

Algorithm 2: Computing the solution spaces of the neutral words with guess-and-determine

Input: $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}]) \in \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\mathtt{ENC}}|+|\mathcal{G}^{\mathtt{KSA}}|)}$
Output: V, U

$V \leftarrow [\,]$, $U \leftarrow [\,]$
for $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}) \in \mathbb{F}_2^{w\cdot(|\mathcal{B}^{\mathtt{ENC}}|+|\mathcal{B}^{\mathtt{KSA}}|+|\mathcal{Y}_+^{\mathtt{ENC}}|+|\mathcal{Y}_+^{\mathtt{KSA}}|)}$ **do**
  $\boldsymbol{v} \leftarrow \boldsymbol{f}^+(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}], \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}})$ by Equation 4.
  Insert $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{B}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{B}^{\mathtt{KSA}}])$ into V at index $(\boldsymbol{v}, \mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}})$.
for $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}], \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}}) \in \mathbb{F}_2^{w\cdot(|\mathcal{R}^{\mathtt{ENC}}|+|\mathcal{R}^{\mathtt{KSA}}|+|\mathcal{Y}_-^{\mathtt{ENC}}|+|\mathcal{Y}_-^{\mathtt{KSA}}|)}$ **do**
  $\boldsymbol{u} \leftarrow \boldsymbol{f}^-(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}], \mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}], \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}})$ by Equation 5.
  Insert $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{R}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{R}^{\mathtt{KSA}}])$ into U at index $(\boldsymbol{u}, \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}})$.

Firstly, Algorithm 2 is given to combine the nonlinearly constrained neutral words and guess-and-determine. Algorithm 2 obtains the solution spaces of the neutral words for all $\mathfrak{c}^+$ and $\mathfrak{c}^-$ together with each guess of $(\mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}, \mathcal{Y}_-^{\mathtt{ENC}}, \mathcal{Y}_-^{\mathtt{KSA}})$ under a given value of $(\mathcal{S}^{\mathtt{ENC}}[\mathcal{G}^{\mathtt{ENC}}], \mathcal{S}^{\mathtt{KSA}}[\mathcal{G}^{\mathtt{KSA}}])$. Its time complexity is $(2^w)^{\lambda^+ + \sigma^+} + (2^w)^{\lambda^- + \sigma^-}$ and its memory complexity is $(2^w)^{\lambda^+ + \sigma^+} + (2^w)^{\lambda^- + \sigma^-}$. Then, we apply Algorithm 2 to the unified MITM preimage attack in Algorithm 3.

Algorithm 3: The MITM preimage attack with nonlinearly constrained neutral words and guess-and-determine

Input: None
Output: Preimage X

1 **for** $(\mathcal{S}^{\text{ENC}}[\mathcal{G}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{G}^{\text{KSA}}]) \in \mathbb{G} \subseteq \mathbb{F}_2^{w\cdot(|\mathcal{G}^{\text{ENC}}|+|\mathcal{G}^{\text{KSA}}|)}$ **do**
2   Call Algorithm 2 to build V, U.
3   **for** $\mathfrak{c}^+ = (a_1, \cdots, a_{l^+}) \in \mathbb{F}_2^{w\cdot l^+}$ **do**
4   **for** $\mathfrak{c}^- = (b_1, \cdots, b_{l^-}) \in \mathbb{F}_2^{w\cdot l^-}$ **do**
5   /* MITM episode starts */
6             $L \leftarrow [\,]$
7   **for** $(\mathcal{Y}_+^{\text{ENC}}, \mathcal{Y}_+^{\text{KSA}}) \in \mathbb{F}_2^{w\cdot(|\mathcal{Y}_+^{\text{ENC}}|+|\mathcal{Y}_+^{\text{KSA}})|}$ **do**
8   **for** $(\mathcal{S}^{\text{ENC}}[\mathcal{B}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{B}^{\text{KSA}}]) \in V[\mathfrak{c}^+, \mathcal{Y}_+^{\text{ENC}}, \mathcal{Y}_+^{\text{KSA}}]$ **do**
9   Compute $E^+[\mathcal{M}^+]$ along the forward computation path.
10   Insert $(\mathcal{S}^{\text{ENC}}[\mathcal{B}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{B}^{\text{KSA}}], \mathcal{Y}_+^{\text{ENC}}, \mathcal{Y}_+^{\text{KSA}})$ into L indexed by $E^+[\mathcal{M}^+]$.
11   **for** $(\mathcal{Y}_-^{\text{ENC}}, \mathcal{Y}_-^{\text{KSA}}) \in \mathbb{F}_2^{w\cdot(|\mathcal{Y}_-^{\text{ENC}}|+|\mathcal{Y}_-^{\text{KSA}})|}$ **do**
12   **for** $(\mathcal{S}^{\text{ENC}}[\mathcal{R}^{\text{KSA}}], \mathcal{S}^{\text{KSA}}[\mathcal{R}^{\text{KSA}}]) \in U[\mathfrak{c}^-, \mathcal{Y}_-^{\text{ENC}}, \mathcal{Y}_-^{\text{KSA}}]$ **do**
13   Compute $E^-[\mathcal{M}^-]$ along the backward computation path.
14   **for** $(\mathcal{S}^{\text{ENC}}[\mathcal{B}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{B}^{\text{KSA}}], \mathcal{Y}_+^{\text{ENC}}, \mathcal{Y}_+^{\text{KSA}}) \in L[E^-[\mathcal{M}^-]]$ **do**
15   Use $(\mathcal{S}^{\text{ENC}}[\mathcal{B}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{B}^{\text{KSA}}], \mathcal{S}^{\text{ENC}}[\mathcal{R}^{\text{ENC}}], \mathcal{S}^{\text{KSA}}[\mathcal{R}^{\text{KSA}}])$ to compute and check the guessed values.
16   **if** *The values* $(\mathcal{Y}_+^{\text{ENC}}, \mathcal{Y}_+^{\text{KSA}}, \mathcal{Y}_-^{\text{ENC}}, \mathcal{Y}_-^{\text{KSA}})$ *are correct* **then**
17   Reconstruct the (candidate) message X.
18   **if** X *is a preimage* **then**
19   Output X and Stop.
20   /* MITM episode ends */

Complexity. From Line 7 to 20 of Algorithm 3, we test $2^{w\cdot(\text{DoF}^+ + \text{DoF}^- + \sigma^+ + \sigma^-)}$ messages and expect $2^{w\cdot(\text{DoF}^+ + \text{DoF}^- + \sigma^+ + \sigma^- - m)}$ of them to pass the m-cell filter. In Line 16, we need to verify the correctness of these partial matchings. The probability that the guessed cells are correct is $2^{-w\cdot(\sigma^+ + \sigma^-)}$, so there will be $2^{w\cdot(\text{DoF}^+ + \text{DoF}^- - m)}$ valid partial matchings that pass the correctness test. Suppose we are going to find a preimage of the h-cell target. Therefore, there are about $2^{w\cdot(\text{DoF}^+ + \text{DoF}^- - h)}$ preimages passing the check at Line 19 for each episode. We need at least to repeat the process $2^{w\cdot(h-(\text{DoF}^+ + \text{DoF}^-))}$ times to produce one preimage. The time complexity to perform one MITM episode is

$$(2^w)^{\text{DoF}^+ + \sigma^+} + (2^w)^{\text{DoF}^- + \sigma^-} + (2^w)^{\text{DoF}^+ + \text{DoF}^- + \sigma^+ + \sigma^- - m}. \tag{6}$$

Depending on the number of available degrees of freedom, the loop at line 1 in Algorithm 3 does not necessarily need to try all values for all the gray cells. We assume the size of $\mathbb{G}$ in Line 1 of Algorithm 3 is $|\,\mathbb{G}\,| = (2^w)^x$, then we can know $x = h - (\lambda^+ + \lambda^-)$. Hence, we consider two situations depending on $\lambda^+ + \lambda^-$.

- $\lambda^+ + \lambda^- \geq h$: In this case, we set $x = 0$, then $|\,\mathbb{G}\,| = 1$. At Line 3 and Line 4 of Algorithm 3, we only need to traverse $(2^w)^{h-(\text{DoF}^+ + \text{DoF}^-)}$ values of $(\mathfrak{c}^+, \mathfrak{c}^-) \in \mathbb{F}_2^{w\cdot l^+ + w\cdot l^-}$, where $h - (\text{DoF}^+ + \text{DoF}^-) \leq l^+ + l^-$ due to $\lambda^+ + \lambda^- \geq h$, to find the preimage. Then, together with Equation (6), the overall time complexity is about:

$$(2^w)^{\lambda^+ + \sigma^+} + (2^w)^{\lambda^- + \sigma^-} + (2^w)^{h - \min(\text{DoF}^+ - \sigma^-,\ \text{DoF}^- - \sigma^+,\ m - (\sigma^+ + \sigma^-))}. \tag{7}$$

- $\lambda^+ + \lambda^- < h$: Set $x = h - (\lambda^+ + \lambda^-)$, and we need to build 2^x V and U in Line 2 of Algorithm 3. Hence, the overall complexity is about:

$$(2^w)^{h-\lambda^-+\sigma^+} + (2^w)^{h-\lambda^++\sigma^-} + (2^w)^{h-\min(\mathrm{DoF}^+-\sigma^-,\ \mathrm{DoF}^--\sigma^+,\ m-(\sigma^++\sigma^-))}. \quad (8)$$

Moreover, the memory complexity for both situations is about

$$(2^w)^{\lambda^++\sigma^+} + (2^w)^{\lambda^-+\sigma^-} + (2^w)^{\min(\mathrm{DoF}^++\sigma^+,\ \mathrm{DoF}^-+\sigma^-)}. \quad (9)$$

4 Automatic MITM Preimage Attacks

At EUROCRYPT 2021, Bao et al. [BDG+21] proposed an automatic method to search the MITM preimage attacks by using Mixed-Integer-Linear-Programming (MILP). At CRYPTO 2021, Dong et al. [DHS+21] extended the automatic model into MITM key-recovery and collision attacks. In [BGST21], Bao et al. enhanced the MILP model of MITM preimage attack by introducing the guess-and-determine [SWWW12], relaxed model and independent linear layer into the automatic tool. We based on their model to further introduce the constraints for both the guess-and-determine technique and nonlinearly constrained neutral words. Although Bao et al.'s [BGST21] model already contained the constraints for the guess-and-determine technique, we include the guess-and-determine into our model by a more simple and direct way.

Firstly, the ith cell of a state $\mathcal{S}$ is encoded by a pair of 0-1 variables $(x_i^{\mathcal{S}}, y_i^{\mathcal{S}})$ as the following rule:

- ■ Gray, $(x_i^{\mathcal{S}}, y_i^{\mathcal{S}}) = (1,1)$, predefined constant, it is known in both forward and backward chunks.
- ■ Blue, $(x_i^{\mathcal{S}}, y_i^{\mathcal{S}}) = (1,0)$, dependent on Gray cells and neutral words for forward chunk, it is known for forward chunk but unknown for backward chunk.
- ■ Red, $(x_i^{\mathcal{S}}, y_i^{\mathcal{S}}) = (0,1)$, dependent on Gray cells and neutral words for backward chunk, it is known for backward chunk but unknown for forward chunk.
- □ White, $(x_i^{\mathcal{S}}, y_i^{\mathcal{S}}) = (0,0)$, dependent on both neutral words for forward and backward computations, it is unknown for both forward and backward chunks.

For the starting states, we introduce variables α_i and β_i for each cell of $(\mathcal{S}^{\texttt{ENC}}, \mathcal{S}^{\texttt{KSA}})$, where $\alpha_i = 1$ if and only if the cell is ■ and $\beta_i = 1$ if and only if the cell is ■. Therefore, we can compute the initial degrees of freedom for forward and backward chunks by $\lambda^+ = \sum_i \alpha_i^{\texttt{ENC}} + \sum_i \alpha_i^{\texttt{KSA}}, \quad \lambda^- = \sum_i \beta_i^{\texttt{ENC}} + \sum_i \beta_i^{\texttt{KSA}}$. For the ending states, we assume the matching only happens at the `MixRows` in the actual attacks on `Streebog`, for each pair of rows of E^+ and E^-, we introduce a variable m_i to indicate the degree of matching in row i which can be constrained by the number of ■, ■ and ■ cells. The total degrees of matching DoM can be computed by $\mathrm{DoM} = \sum_{i=0}^{7} m_i$. For more details, we refer to [BDG+21].

Then we build attribute propagation rules for each operation of the attacked hash function and record the consumption of the degrees of freedom. The process of adding constraints on neutral words consumes the degrees of freedom of neutral words. We assume the accumulated consumed degrees of freedom of forward and backward chunks are l^+ and l^- respectively. We can compute the remaining degrees of freedom for forward and backward chunks by $\mathrm{DoF}^+ = \lambda^+ - l^+, \mathrm{DoF}^- = \lambda^- - l^-$. The rules `XOR-RULE` and `MC-RULE` introduced in [BDG+21] are used to build the rules of `AddRoundKey` and `MixColumns` of `AES`-like hashing. For more details of these rules see Section B. In the MILP model of attacking `Streebog`, we can use `XOR-RULE` to build the rules of `AddRoundKey` and use `MC-RULE` to build the rules of `MixRows`. In addition, we can easily build the rules of `Transposition` because it just permutes the color scheme of the input state. As for `SubBytes`, we can ignore it because it does not change the color of the input state.

In addition, we need to build some constraints to get the values of σ^+ and σ^- which are the number of guessed cells in the forward and backward chunks. In general, guess-and-determine is often used before the diffusion operations because one unknown cell in the input of diffusion operation may make many cells in the output unknown. Taking `MixColumns` for example, we assume the input state and output state of `MixColumns` are $\mathcal{S}_{in}$ and $\mathcal{S}_{out}$. We introduce another state $\tilde{\mathcal{S}}_{in}$ and let `MixColumns` link $\tilde{\mathcal{S}}_{in}$ and $\mathcal{S}_{out}$. Then we introduce an operation named `Guess` to link $\mathcal{S}_{in}$ and $\tilde{\mathcal{S}}_{in}$, as shown in Figure 2.

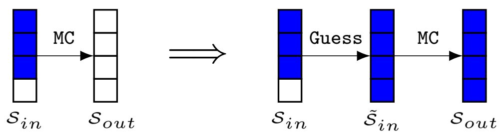
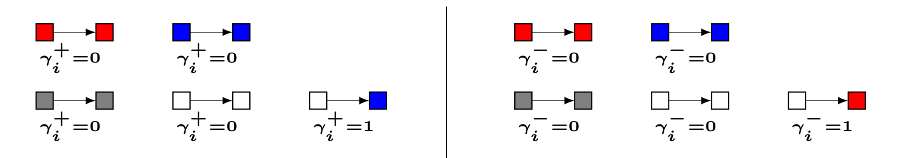


Figure 2: Introduce `Guess` operation before `MC`

In the forward chunk, we build the rule named `GUESS`$^+$`-RULE` for `Guess` operation. Concretely, the `GUESS`$^+$`-RULE` keeps the cell unchanged if the input cell is 🟥, 🟦 or ⬛, while it keeps the □ cell unchanged or changes the □ to 🟦. We introduce a variable γ_i^+ for each cell of the state, $\gamma_i^+ = 1$ if and only if the □ is changed into 🟦. The `GUESS`$^+$`-RULE` is shown in Figure 3(a). Then we need to convert the `GUESS`$^+$`-RULE` to linear inequalities to get



(a) Forward computation (GUESS$^+$-RULE) (b) Backward computation (GUESS$^-$-RULE)

Figure 3: The rule of `Guess` in forward and backward chunks

the constraints, the set of rule `GUESS`$^+$`-RULE` restricts $(x^{\mathcal{S}_{in}}, y^{\mathcal{S}_{in}}, x^{\tilde{\mathcal{S}}_{in}}, y^{\tilde{\mathcal{S}}_{in}}, \gamma_i^+)$ to subsets of $\mathbb{F}_2^5$, which can be described by a system of linear inequalities by using the convex hull computation method [SHW$^+$14]. Similarly, we can build the rule named `GUESS`$^-$`-RULE` in the backward chunk. As shown in Figure 3(b), `GUESS`$^-$`-RULE` keeps the cell unchanged if the cell of $\mathcal{S}_{in}$ is 🟥, 🟦 or ⬛, while it keeps the □ cell unchanged or changes the □ to 🟥. We also introduce a variable γ_i^- for each cell of the state. $\gamma_i^- = 1$ if and only if the □ are changed into 🟥. We use the same method to convert it to linear inequalities.

In order to distinguish the guessed cells obviously, we unifiedly use 🟨 to represent these guessed cells of $\tilde{\mathcal{S}}_{in}$ in the forward and backward chunks. Therefore, $\tilde{\mathcal{S}}_{in}[i]$ is 🟨 if $\gamma_i^+ = 1$ in the forward chunk or $\gamma_i^- = 1$ in the backward chunk. In addition, we can compute the number of guessed cells in the forward and backward chunks σ^+ and σ^- by $\sigma^+ = \sum \gamma_i^+, \sigma^- = \sum \gamma_i^-$. Finally, since the time complexity is given by Equation (7) and (8), we introduce an auxiliary variable v_{obj}, impose the constraints

$$\{v_{obj} \le \mathrm{DoF}^+ - \sigma^-, v_{obj} \le \mathrm{DoF}^- - \sigma^+, v_{obj} \le \mathrm{m} - \sigma^+ - \sigma^-\}. \tag{10}$$

Our objective function is to maximize the value of v_{obj}. Besides, additional constraints should be added to the model according to the value of $\lambda^+ + \lambda^-$.

$$\begin{cases} \lambda^+ + \sigma^+ < h, \ \ \lambda^- + \sigma^- < h; & \text{if} \ \ \lambda^+ + \lambda^- \ge h, \\ \lambda^- - \sigma^+ > 0, \ \ \lambda^+ - \sigma^- > 0; & \text{if} \ \ \lambda^+ + \lambda^- < h. \end{cases} \tag{11}$$

Let ini_r, ini_k and $match_r$ denote the round number of $\mathcal{S}^{\mathtt{ENC}}$, $\mathcal{S}^{\mathtt{KSA}}$ and E^+ respectively. For searching N-round attacks, we enumerate all possible combinations of ini_r, ini_k and

$match_r$, where $0 \le ini_r < N, 0 \le ini_k < N, 0 \le match_r < N$ and generate an MILP model for each $(ini_r, ini_k, match_r)$. Then we use the MILP solver Gurobi to search the optimal attack for each MILP model. Once a solution is found, we can draw it in a figure according to the values of pair variables of each cell.

Remark. Our model is different from the one in [BGST21]. Firstly, we employed the similarity of the encryption and key-schedule data paths. We considered two situations where `AddRoundKey` is placed before or after `MixColumns`, which can be implemented by the indicator constraints in Gurobi as mentioned in [BGST21]. However, we did not use the "relaxed model" proposed by Bao et al. [BGST21], the solution space of the "relaxed model" is larger than ours. Consequently, the optimal solution of their model should be better than ours. However, the search space of the "relaxed model" is too large and the corresponding MILP model cannot be solved in practical time. Therefore, they employ round-dependent modeling, symmetry and similarity techniques to reduce the search space. It seems that the solution space of the reduced model covers some different MITM trails than our models and at the same time misses some trails covered by our model.

Besides, they built the rule for the combination of `MC` and Guess, in detail, they introduced another variable for each cell to indicate if the cell is guessed. Hence, they have to rewrite all the rules for each cell by considering the additional variable. In our model, we make `MC` and Guess totally separate by introducing a new operation `Guess` and an auxiliary state, which will not affect other rules. Then we just need to build the rule for `Guess` and it is simple and intuitive. The total size of our model is smaller. In addition, in comparison to the MILP built in [BDG+21] without guess-and-determine, our method will not have a significant increase in the size of the MILP model and it will also not increase too much the time needed to solve it. We used Gurobi 9.0.3 to solve all the MILP models. It took about one week on a PC with Fedora Linux 30 and 128 GB memory to find the attacks on 8.5/7.5-round `Streebog-512` and 7.5-round `Streebog-256`. As for the 6.5-round `Streebog-256`, it just took about several hours to find the attack because the key is fixed in this model. The source code is provided at https://github.com/dongxiaoyang/streebog-mitm.

5 Application to `Streebog`

In this section, we give a brief description of `Streebog`, and then show our preimage attacks on round-reduced compression functions of `Streebog-512` and `Streebog-256`.

5.1 Specifications of `Streebog`

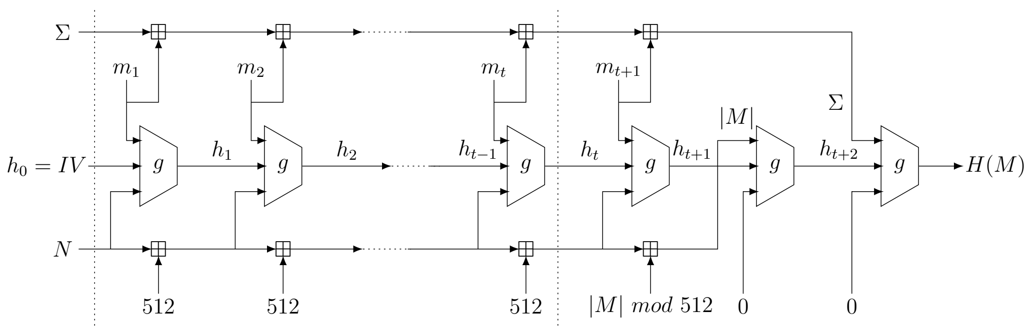


Figure 4: The `Streebog` hash function

`Streebog` is a family of two hash functions, `Streebog-256` and `Streebog-512`. They both accept message blocks size of 512 bits and output 256-bit and 512-bit hash digest respectively. As shown in Figure 4, Firstly, the input message M is padded into a multiple of 512 bits. The bit "1" is appended to the end of the message, and followed by $512-1-(|M| \; mod \; 512)$ 0-bit, where $|M|$ denotes the length of the message. Then the padded message can be divided into $t+1$ 512-bit blocks $m_1||m_2||\cdots||m_{t+1}$. The three variables Σ, N, h_0 are assigned to 0,0 and IV respectively. Secondly, each block m_i $(1 \leq i \leq t+1)$ is processed iteratively according to the following operations: $h_i = g(N, h_{i-1}, m_i), N = N + 512, \Sigma = \Sigma + m_i$. Finally, the output chaining value of the last message block h_{t+1} goes through the output transformation by: $h_{t+2} = g(0, h_{t+1}, |M|), H(M) = g(0, h_{t+2}, \Sigma)$

For `Streebog-512`, $H(M)$ is the hash digest. The $\text{MSB}_{256}(H(M))$ is the hash digest of `Streebog-256`. (MSB_{256} means the 256 most significant bits). The compression function $g(N, h, m)$ contains a 512-bit block cipher E and it is calculated as $g(N, h, m) = E(L \circ P \circ S(h \oplus N), m) \oplus h \oplus m$. The block cipher E is an `AES`-based cipher which updates an 8×8 state of 64 bytes and round key in 12 rounds. The initial state is $S_0 = m$, and in each round, the state is updated by `AddRoundKey` (X), `SubBytes`(S), `Transposition` (P) and `MixRows` (L), i.e., $S_{j+1} = L \circ P \circ S(S_j \oplus K_j), \quad j = 0, 1, \cdots, 11$, and finally, the ciphertext is computed by $S_{12} \oplus K_{12}$. K_0 is initialized by $K_0 = L \circ P \circ S(h \oplus N)$ and the round key K_i is updated as $K_i = L \circ P \circ S(K_{i-1} \oplus C_{i-1}), \; 1 \leq i \leq 12$, where C_{i-1} is a round-dependent constant. For more details, we refer to the original paper [GOS12].

5.2 Preimage Attack on Reduced `Streebog-512`'s Compression Function

We find preimage attacks on 7.5-round and 8.5-round `Streebog-512` compression function. In this section, we show the attack on 8.5-round `Streebog-512` compression function and the attack on 7.5-round is given in Appendix A. The preimage attack on the 8.5-round `Streebog-512` compression function is shown in Figure 5, K_i and K_i' represent the states in the key schedule path, X_i, Y_i, Z_i and W_i represent the states in the encryption path, The "X" operation on the key schedule path means XORing a round-dependent constant. The starting states are W_3 and K_5, the ending states are Z_6 and W_6. In W_3, there are 36 ■ cells, 4 ■ cells and 24 ■ cells. In K_5, there are 16 ■ cells and 48 ■ cells. Therefore, the initial degrees of freedom for forward and backward chunks are $\lambda^+ = 36$ and $\lambda^- = 16 + 4 = 20$, respectively. The matching happens between Z_6 and W_6, which forms a 16-cell filter. In addition, there are 12 guessed cells which are represented by ■ in Y_1.

$$\begin{pmatrix} a_1 & a_3 & a_5 & a_7 & a_9 & a_{11} & a_{13} & a_{15} \\ a_2 & a_4 & a_6 & a_8 & a_{10} & a_{12} & a_{14} & a_{16} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{pmatrix} = L^{-1} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_3[2] & Y_3[10] & Y_3[18] & Y_3[26] & Y_3[34] & Y_3[42] & Y_3[50] & Y_3[58] \\ Y_3[3] & Y_3[11] & Y_3[19] & Y_3[27] & Y_3[35] & Y_3[43] & Y_3[51] & Y_3[59] \\ Y_3[4] & Y_3[12] & Y_3[20] & Y_3[28] & Y_3[36] & Y_3[44] & Y_3[52] & Y_3[60] \\ Y_3[5] & Y_3[13] & Y_3[21] & Y_3[29] & Y_3[37] & Y_3[45] & Y_3[53] & Y_3[61] \\ Y_3[6] & Y_3[14] & Y_3[22] & Y_3[30] & Y_3[38] & Y_3[46] & Y_3[54] & Y_3[62] \\ Y_3[7] & Y_3[15] & Y_3[23] & Y_3[31] & Y_3[39] & Y_3[47] & Y_3[55] & Y_3[63] \end{pmatrix}. \tag{12}$$

Firstly, we consider the reduction of degrees of freedom for the ■ cells. From Y_3 to Z_2, the constraints in Equation (12) are applied, where $(a_1, a_2, \cdots, a_{16})$ are constants marked in Z_2. These constraints can ensure that the ■ cells of Y_3 have no impact on the first two columns of Z_2, so the first two columns of Z_2 only depend on ■ cells in K_3 and Y_3. The constraints introduce a 16-cell reduction of degrees of freedom for ■ cells, so the remaining degrees of freedom for ■ cells is $\text{DoF}^+ = \lambda^+ - l^+ = 36 - 16 = 20$. Then we call Algorithm 2 to compute and build the table V which stores the solution spaces of ■ cells, i.e., for fixed ■ in W_3, traverse the ■ cells in W_3 to compute a_i $(1 \leq i \leq 16)$. Note that, Algorithm 2 is more like a generic case in which the guessed cells are also involved. However, for the attack in Figure 5, the guessed cells are not involved in the procedure of building table V, so we do not need to traverse the guessed cells.

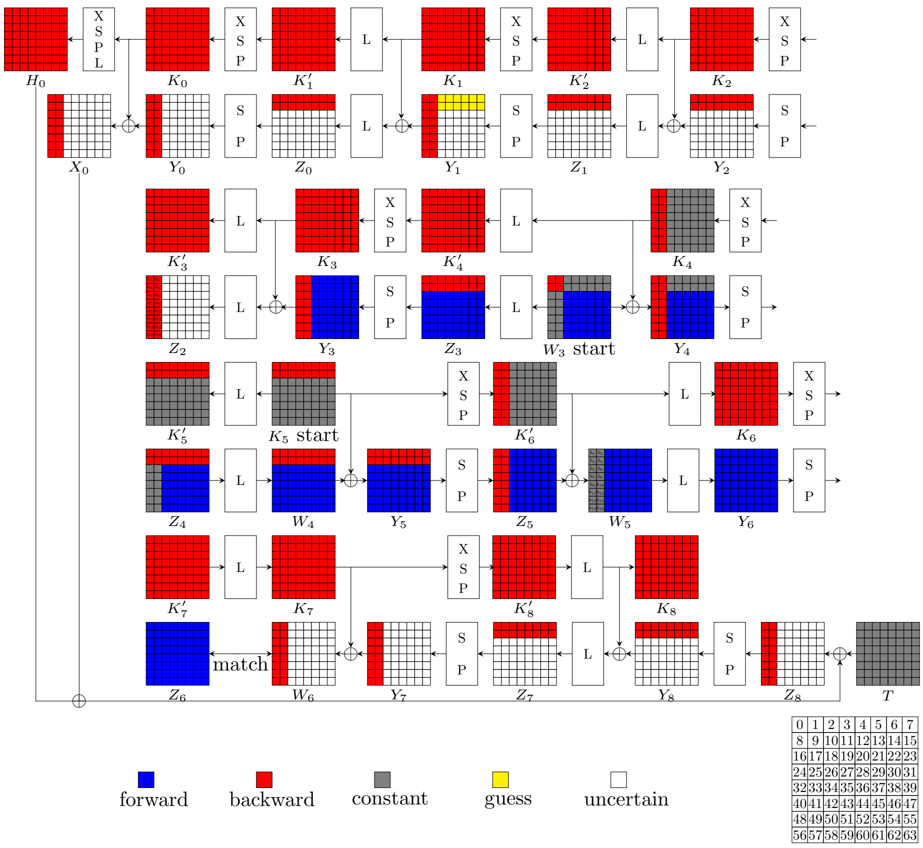


Figure 5: 8.5-round preimage attack on `Streebog-512` compression function

Then we consider the reduction of degrees of freedom for the ■ cells. From Z_5 to W_5, the first two columns of W_5 are constant. Hence, we have Equation (13) with constants $(b_1, b_2, \cdots, b_{16})$ in W_5.

$$\begin{pmatrix} Z_5[0] & Z_5[1] \\ Z_5[8] & Z_5[9] \\ Z_5[16] & Z_5[17] \\ Z_5[24] & Z_5[25] \\ Z_5[32] & Z_5[33] \\ Z_5[40] & Z_5[41] \\ Z_5[48] & Z_5[49] \\ Z_5[56] & Z_5[57] \end{pmatrix} \oplus \begin{pmatrix} K'_6[0] & K'_6[1] \\ K'_6[8] & K'_6[9] \\ K'_6[16] & K'_6[17] \\ K'_6[24] & K'_6[25] \\ K'_6[32] & K'_6[33] \\ K'_6[40] & K'_6[41] \\ K'_6[48] & K'_6[49] \\ K'_6[56] & K'_6[57] \end{pmatrix} = \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \\ b_5 & b_6 \\ b_7 & b_8 \\ b_9 & b_{10} \\ b_{11} & b_{12} \\ b_{13} & b_{14} \\ b_{15} & b_{16} \end{pmatrix}. \tag{13}$$

Algorithm 4: The MITM preimage attack on 8.5-round `Streebog-512` compression function

```
Fix all ■ cells of K_5 to 0 and arbitrary 16 ■ cells of W_3 to 0.
for All 8 cells that are not fixed in W_3 do
    Call Algorithm 2 to build V and U.
    for c+ = (a_1, a_2, ..., a_16) ∈ F_2^{8×16} do
        for c− = (b_1, b_2, ..., b_16) ∈ F_2^{8×16} do
            for all values in V[c+] do
                Compute forward to get the full state of Z_6 and store it in a table L.
            for Y_−^ENC ∈ F_2^{8×12} (□ cells of Y_1) do
                for all values in U[c−] do
                    Compute backward to get the first two columns of W_6 and search L to
                     find matching.
                    Use the matching pairs to compute and check if the guessed values
                     Y_−^ENC are correct.
                    if The guessed values Y_−^ENC are correct then
                        Test the full preimage.
                        if The full preimage is found then
                            Output and stop.
```

By Algorithm 2, given fixed constant ■ cells in starting states W_3 and K_5, we traverse $\lambda^- = 4 + 16 = 20$ ■ cells in W_3 and K_5 to compute the solution space of ■ cells. In detail, we compute K_4 and K'_6 from K_5. Then, compute ■ cells in Y_4 by W_3 and K_4. Compute W_4 and then Y_5 and Z_5. Finally, compute b_i $(1 \le i \le 16)$ with Equation (13). We can know $l^- = 16$ and $\mathfrak{c}^- = (b_1, b_2, \cdots, b_{16}) \in \mathbb{F}_2^{16}$. The remaining degrees of freedom for ■ cells is $\mathrm{DoF}^- = \lambda^- - l^- = 20 - 16 = 4$. Therefore, we can call Algorithm 2 to build a table U which stores the solution spaces of ■ cells. Similarly, we do not need to traverse the guessed cells because they are not involved in the procedure of building U.

The whole preimage attack on `Streebog-512` compression function is shown in Algorithm 4. We are going to find a 512-bit preimage attack, the state of encryption data path and key schedule path are both 512-bit so that we have enough freedom degrees to find the preimage. Therefore, we can fix some ■ cells of K_4 and W_3 to zero in the whole attack. Note that the guessed cells are only in Y_1 in the backward computation, so $\mathcal{Y}_+^{\mathtt{ENC}}, \mathcal{Y}_+^{\mathtt{KSA}}, \mathcal{Y}_-^{\mathtt{KSA}}$ will not appear in this attack.

Complexity. As shown in Figure 5, we can know $h = 64$. $\lambda^+ = 36, \lambda^- = 20$, so $\lambda^+ + \lambda^- = 36 + 20 = 56 < h$, we can get the time complexity and memory complexity by Equations (8) and (9). In the attack, $\sigma^+ = 0,\ \sigma^- = 12,\ \mathrm{DoF}^+ = 20,\ \mathrm{DoF}^- = 4$ and $m = 16$. Therefore, we can get the time complexity

$$(2^8)^{64-20+0} + (2^8)^{64-36+12} + (2^8)^{64-(20-12)} + (2^8)^{64-(4-0)} + (2^8)^{64-(16-12)} \approx 2^{481},$$

and the memory complexity

$$(2^8)^{36+0} + (2^8)^{20+12} + (2^8)^{min(20+0,4+12)} \approx 2^{288}.$$

Remark on Complexity. In Algorithm 4, step 10-13 will be repeated 2^{480} times. We assume computing backward costs 1 encryption, in step 10, there will be 2^{480} encryptions. We assume the computation in step 11 is 1 encryption, so there will be 2^{480} encryptions. In step 13, the computation is one encryption, but it is repeated 2^{384} times. Therefore, the overall time complexity of step 10-13 is about 2^{481} encryptions.

5.3 Preimage Attack on Reduced `Streebog-256`'s Compression Function

We find a preimage attack on 7.5-round `Streebog-256` compression function, which is shown in Figure 6. The starting states are W_2 and K_4, we can know $\lambda^+ = 30$ and $\lambda^- = 24 + 6 = 30$. From Y_2 to Z_1, it consumes 16-cell degrees of freedom for ■ cells, so $\text{DoF}^+ = 30 - 16 = 14$. From Z_4 to W_4, it consumes 24-cell degrees of freedom for ■ cells, so $\text{DoF}^- = 30 - 24 = 6$. The matching point is between Z_5 and W_5 and we get a filter of $\text{DoM} = 16$ cells. In addition, we guess 8 cells represented by □ in Y_7 . Because the target is `Streebog-256`, the time complexity of exhaustive search to find a preimage is just 2^{256}. If we use Algorithm 2 to build the tables V and U, the total size of V and U are 2^{240} and 2^{304}, which will lead to a total time complexity higher than exhaustive search. However, Algorithm 2 is just a generic case and we can tweak it in kinds of attacks according to the specific situations. In the attack on `Streebog-256`, we give a procedure (Algorithm 5) to build the table V. $(a'_1, \cdots, a'_{16})$ are constants, which are marked in Z_1 shown in Figure 6. For simplicity, we use X^{col_i}/X^{row_i} $(i = \{0, 1, \cdots, 7\})$ represents the i-th column/row of X, and $X^{col_i}[j]/X^{row_i}[j]$ $(j = \{0, 1, \cdots, 7\})$ means j-th cell of i-th column/row of X.

Algorithm 5: Compute the solution space of Blue neutral cells in Figure 6

Input: $\mathfrak{c}^+ = (a'_1, \cdots, a'_{16})$
Output: $V[\mathfrak{c}^+]$

Fix the Gray cells in W_2.
$V[\mathfrak{c}^+] = \emptyset$.
for *All possible values of* $W_2^{row_i}[3, 4, 5, 6, 7](i = 2, 3, 4, 5)$ **do**
  (a). $(Z_2^{row_i})^T = L^{-1} \cdot (W_2^{row_i})^T$, $Y_2^{col_i} = S^{-1}(Z_2^{row_i})(i = 2, 3, 4, 5)$.
  (b). $L^{-1} \cdot (Y_2^{row_0})^T = (Z_1^{row_0})^T$, namely,

$$L^{-1}(0, 0, Y_2[2], Y_2[3], Y_2[4], Y_2[5], Y_2[6], Y_2[7])^T = (a'_1, a'_2, \text{-}, \text{-}, \text{-}, \text{-}, \text{-}, \text{-})^T$$

  the unknown values of $Y_2[6], Y_2[7]$ can be uniquely determined because L is a MDS matrix.
  (c). Solving $L^{-1} \cdot (Y_2^{row_i})^T = (Z_1^{row_i})^T$ $(i = 1, 2, \cdots, 7)$, $Y_2^{col_6}, Y_2^{col_7}$ can be uniquely determined.
  (d). $Z_2^{row_6} = S(Y_2^{col_6}), Z_2^{row_7} = S(Y_2^{col_7})$.
  if $W_2^{row_6}[0, 1, 2] = (L \cdot (Z_2^{row_6})^T)[0, 1, 2]$, $W_2^{row_7}[0, 1, 2] = (L \cdot (Z_2^{row_7})^T)[0, 1, 2]$ **then**
    Store the values of $W_2^{row_i}[3, 4, 5, 6, 7](i = 2, 3, 4, 5, 6, 7)$ in $V[\mathfrak{c}^+]$.

Given fixed constant ■ in the starting states W_2, together with the constant $(a'_1, \cdots, a'_{16})$, we traverse 20 ■ cells in the rows 2-5 of W_2 to compute the solution space of ■ cells. As shown in Algorithm 5, we firstly compute four rows of ■ cells of Z_2 and compute four columns of ■ cells of Y_2. Then we use Equation (14) to compute the last two unknown columns ■ cells of Y_2 and then compute the last two rows of ■ cells of Z_2. Finally we need to check

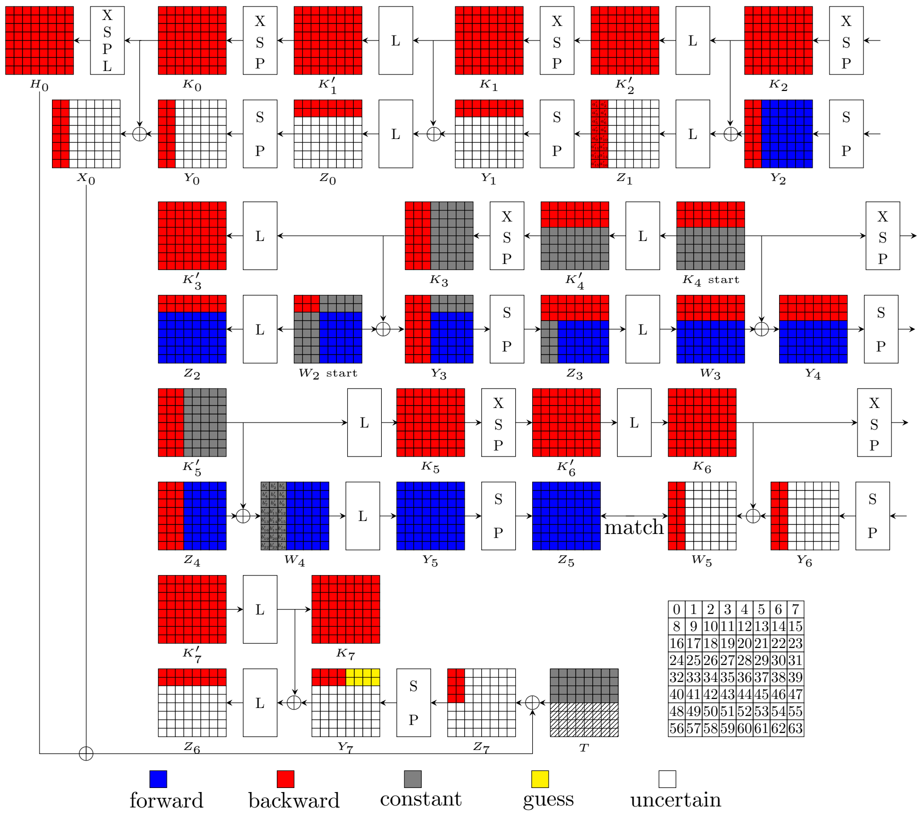


Figure 6: 7.5-round preimage attack on `Streebog-256` compression function

whether $W_2^{row_6}[0,1,2] = (L \cdot (Z_2^{row_6})^T)[0,1,2]$ and $W_2^{row_7}[0,1,2] = (L \cdot (Z_2^{row_7})^T)[0,1,2]$ hold or not. The computation of Algorithm 5 between line 3 to line 9 will be repeated $(2^8)^{20} = 2^{160}$ times. Therefore, for a given value of $\mathfrak{c}^+ = (a'_1, \cdots, a'_{16})$, we can build the table $V[\mathfrak{c}^+]$ with cost of 2^{160}. The probability of $W_2^{row_6}[0,1,2] = (L \cdot (Z_2^{row_6})^T)[0,1,2]$, $W_2^{row_7}[0,1,2] = (L \cdot (Z_2^{row_7})^T)[0,1,2]$ hold is 2^{-48}, so there are about 2^{112} elements in $V[\mathfrak{c}^+]$ in average.

$$\begin{pmatrix} a'_1 & a'_3 & a'_5 & a'_7 & a'_9 & a'_{11} & a'_{13} & a'_{15} \\ a'_2 & a'_4 & a'_6 & a'_8 & a'_{10} & a'_{12} & a'_{14} & a'_{16} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \end{pmatrix} = L^{-1} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ Y_2[2] & Y_2[10] & Y_2[18] & Y_2[26] & Y_2[34] & Y_2[42] & Y_2[50] & Y_2[58] \\ Y_2[3] & Y_2[11] & Y_2[19] & Y_2[27] & Y_2[35] & Y_2[43] & Y_2[51] & Y_2[59] \\ Y_2[4] & Y_2[12] & Y_2[20] & Y_2[28] & Y_2[36] & Y_2[44] & Y_2[52] & Y_2[60] \\ Y_2[4] & Y_2[13] & Y_2[21] & Y_2[29] & Y_2[37] & Y_2[45] & Y_2[53] & Y_2[61] \\ Y_2[6] & Y_2[14] & Y_2[22] & Y_2[30] & Y_2[38] & Y_2[46] & Y_2[54] & Y_2[62] \\ Y_2[7] & Y_2[15] & Y_2[23] & Y_2[31] & Y_2[39] & Y_2[47] & Y_2[55] & Y_2[63] \end{pmatrix}. \quad (14)$$

Algorithm 6: Compute the solution space of Red neutral cells in Figure 6

Input: $\mathfrak{c}^- = (b'_1, b'_2, \cdots, b'_{24})$
Output: $U[\mathfrak{c}^-]$

```
Fix Gray cells in W_2.
U[c^-] = ∅.
for All possible values of Red cells in Z_4 do
    Compute K'_5^{col_i} = W_4^{col_i} ⊕ Z_4^{col_i} (i = 0, 1, 2) (Equation (15)).
    Compute backward to get the values of ■ cells in Y_3 and K_3.
    if Y_3^{row_i}[0, 1, 2] ⊕ K_3^{row_i}[0, 1, 2] = W_2^{row_i}[0, 1, 2] (i = 2, 3, 4, 5, 6, 7) then
        Store the values of Z_4^{col_i} (i = 0, 1, 2) in U[c^-].
```

Next we give Algorithm 6 to build the table U which stores the solutions of ■ cells. Note that in Algorithm 2, the guessed cells are considered when we build the table U. However, there are no guessed cells involved in the computation of U of the attack in Figure 6, so we do not need to traverse the guessed cells in the process of building U.

$$\begin{pmatrix} K'_5[0] & K'_5[1] & K'_5[2] \\ K'_5[8] & K'_5[9] & K'_5[10] \\ K'_5[16] & K'_5[17] & K'_5[18] \\ K'_5[24] & K'_5[25] & K'_5[26] \\ K'_5[32] & K'_5[33] & K'_5[34] \\ K'_5[40] & K'_5[41] & K'_5[42] \\ K'_5[48] & K'_5[49] & K'_5[50] \\ K'_5[56] & K'_5[57] & K'_5[58] \end{pmatrix} = \begin{pmatrix} b'_1 & b'_2 & b'_3 \\ b'_4 & b'_5 & b'_6 \\ b'_7 & b'_8 & b'_9 \\ b'_{10} & b'_{11} & b'_{12} \\ b'_{13} & b'_{14} & b'_{15} \\ b'_{16} & b'_{17} & b'_{18} \\ b'_{19} & b'_{20} & b'_{21} \\ b'_{22} & b'_{23} & b'_{24} \end{pmatrix} \oplus \begin{pmatrix} Z_4[0] & Z_4[1] & Z_4[2] \\ Z_4[8] & Z_4[9] & Z_4[10] \\ Z_4[16] & Z_4[17] & Z_4[18] \\ Z_4[24] & Z_4[25] & Z_4[26] \\ Z_4[32] & Z_4[33] & Z_4[34] \\ Z_4[40] & Z_4[41] & Z_4[42] \\ Z_4[48] & Z_4[49] & Z_4[50] \\ Z_4[56] & Z_4[57] & Z_4[58] \end{pmatrix} \quad (15)$$

Given fixed constant ■ in W_2, together with the constants $(b'_1, b'_2, \cdots, b'_{24})$ which are constants marked in W_4, we traverse the 24 ■ cells in Z_4 to compute the solution space of ■. In detail, we compute the ■ of K'_5 by Equation (15). Then we compute the ■ of K_4 and K_3 from K'_5. Compute Y_4 and then W_3 and Y_3. Finally, we need to check whether $Y_3^{row_i}[0,1,2] \oplus K_3^{row_i}[0,1,2] = W_2^{row_i}[0,1,2]$ $(i = 2,3,4,5,6,7)$ hold or not. The probability that the equations hold is about 2^{-144}, so there are about 2^{48} elements in $U[\mathfrak{c}^-]$ for a given $\mathfrak{c}^-$. The memory to store U is 2^{192}. Finally we give the MITM preimage attack on 7.5-round `Streebog-256` compression function in the Algorithm 7. The time complexity is about 2^{209}, and the memory complexity is bounded by 2^{192} to store U.

Remark. In the attack on `Streebog-256`, we do not use Algorithm 2 to compute the solution spaces of neutral words because the time complexity will be greater than exhaustive search if we use Algorithm 2 directly. In the process of searching for attacks, we firstly add the additional constraints Equation (11) to the MILP model to make sure that the time complexity of Algorithm 2 is smaller than exhaustive search (e.g. the attack on Streebog-512). If there are no solution found, then we will delete the additional constraints

Algorithm 7: The MITM preimage attack on 7.5-round `Streebog-256` compression function

Fix all the ■ cells of W_2 to 0 and arbitrary 28 ■ cells of K_4 to 0.
$\mathfrak{c}^+ = (a'_1, a'_2, \cdots, a'_{16}) \leftarrow 0$
$\mathfrak{c}^- = (b'_1, b'_2, \cdots, b'_{24}) \leftarrow 0$
Call Algorithm 5 and 6 to build table $V[\mathfrak{c}^+]$ and $U[\mathfrak{c}^-]$.
for *All 12 not fixed* ■ *cells in* K_4 **do**
  for *all values in* $V[\mathfrak{c}^+]$ **do**
    Compute forward to get the full state of Z_5 and store it in a table $L[\,]$.
  for *all values in* $U[\mathfrak{c}^-]$ *and* $\mathcal{Y}_-^{\mathtt{ENC}} \in \mathbb{F}_2^{8\times 8}$ *(*□ *cells of* Y_7*)* **do**
    Compute backward to get the first two columns of W_5 and search L to find matching.
    Use the matching pairs to compute and check if the guessed values $\mathcal{Y}_-^{\mathtt{ENC}}$ are correct.
    if *The guessed values* $\mathcal{Y}_-^{\mathtt{ENC}}$ *are correct* **then**
      Test the full preimage.
      if *The full preimage is found* **then**
        Output and stop.

of Equation (11) in the MILP model and run it again. In this case, we may get many MITM trails, but it is not sure that they lead to a successful attack. Therefore, we usually need some tweaked algorithms (Algorithm 5 and Algorithm 6) to replace Algorithm 2 to build the tables for the solution spaces of neutral words. Luckily, we find a successful MITM attack on Streebog-256 whose time complexity of building table V and U can be lower than exhaustive search by Algorithm 5 and Algorithm 6.

6 Preimage attack on Round-Reduced `Streebog-512`

In this section, we generate the preimage attacks on 7.5-round and 8.5-round `Streebog-512` hash function by using the attacks on their compression function in Section 5 and other techniques by AlTawy et al. [AY14]. The attacks are similar and they are both composed of four steps, as shown in Figure 7. The detailed procedure is shown below.

1. Given the hash function output $H(M)$, we produce 2^k preimages (Σ, h_{515}) for the last compression function and store them in a table T.
2. Using Joux's multicollisions [Jou04] we construct 2^{512} messages with a length of 512 message blocks, which all lead to the same value of h_{512}. Specifically, $M_i = m_1^j || m_2^j || \cdots || m_{512}^j$ $(i \in \{1, 2, \cdots, 2^{512}\}, j \in \{1, 2\})$, so we have 2^{512} candidates Σ_{M_i}.
3. Assume the message is 513 complete blocks, then m_{pad} and $|M|$ are known constants. By randomly choosing 2^{512-k} m_{513}, together with h_{512} produced in step 2 and the known values N_{513}, N_{514}, to compute h_{515}, it is expected to find a right m_{513} such that $h_{515} \in T$. Once we find a matching, Σ is known, so we can compute the sum Σ_{M_i} by $\Sigma_{M_i} = \Sigma - m_{pad} - m_{513}$.
4. We compute all the 2^{256} sums of all the 2^{256} 256-block message $\Sigma_{M_1} = m_1^j + m_2^j + \cdots + m_{256}^j$ and store them in a table T_1. Then, compute the sum of other 256-block messages $\Sigma_{M_2} = m_{266}^j + m_{267}^j + \cdots + m_{512}^j$ and check if $\Sigma_{M_i} - \Sigma_{M_2}$ is in T_1. Once we find a matching, we know that the full 513-block message $M = m_1^j || m_2^j || \cdots || m_{512}^j || m_{513}$ is the preimage of the given $H(M)$.

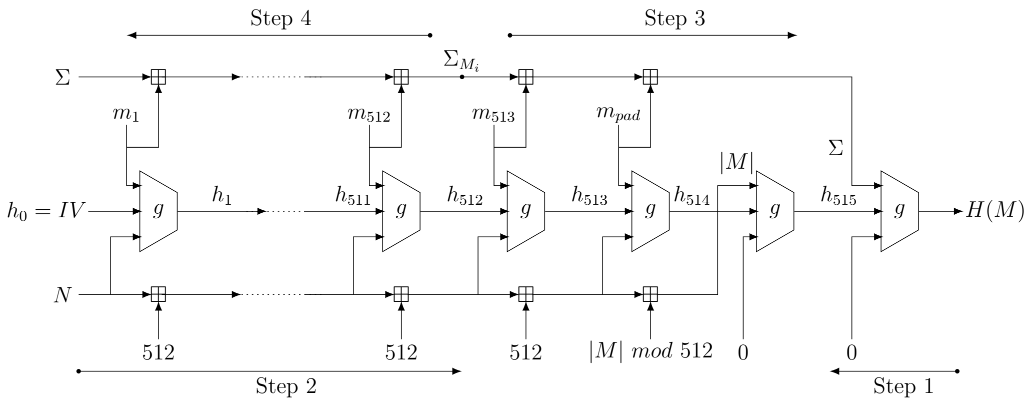


Figure 7: Framework of preimage attack on `Streebog-512`

Complexity. For 8.5-round `Streebog-512`, $k = 16.25$ is an almost optimal solution, so the time complexity is about $2^{16.25} \cdot 2^{481} + 512 \times 2^{256} + 3 \times 2^{495.75} + 2^{256} \approx 2^{498.25}$ and the memory complexity is about 2^{288}, which is bounded by the preimage attack on the compression function. For 7.5-round `Streebog-512`, $k = 36.25$ is an almost optimal solution, the time complexity is about $2^{36.25} \cdot 2^{441} + 512 \times 2^{256} + 3 \times 2^{475.75} + 2^{256} \approx 2^{478.25}$ and the memory complexity is about 2^{256}.

7 Preimage attack on Round-Reduced `Streebog-256`

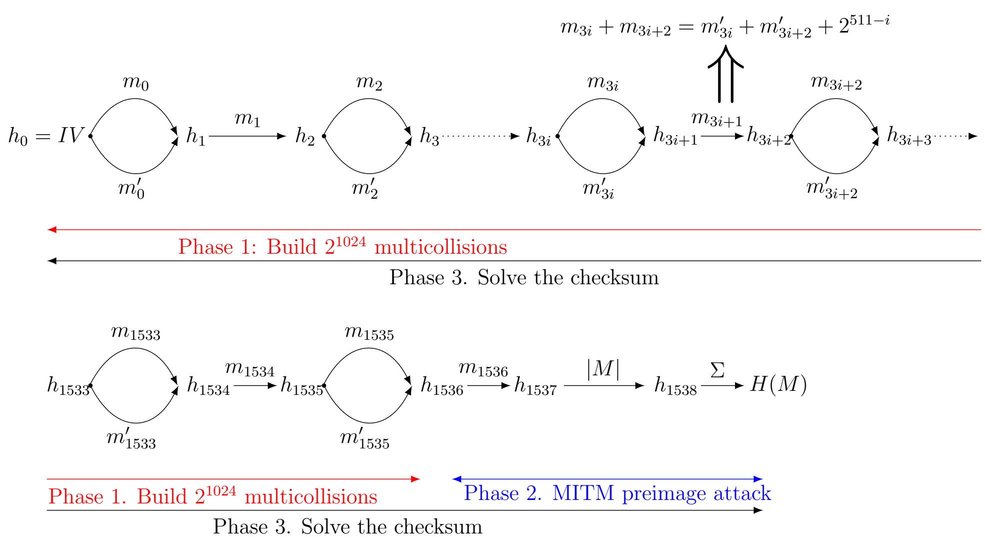


Figure 8: Framework of preimage attack on `Streebog-256`

For `Streebog-256`, we give an improved preimage attack on 6.5-round `Streebog-256` hash function. We use a better preimage attack on 6.5-round `Streebog-256` compression function and then apply Ma's [MLHL15b] method to find the preimage attack on the 6.5-round hash function with lower time complexity. As shown in Figure 8, the attack consists of three phases.

Phase 1: Construct the Multicollisions. We need to construct 2^{1024}-multicollisions which are composed of 512 pairs of 4-multicollisions, namely, $(m_{3i}, m'_{3i})||m_{3i+1}||(m_{3i+2}, m'_{3i+2})$ for $i = 0, 1, \cdots, 511$ and they satisfy $m_{3i} + m_{3i+2} = m'_{3i} + m'_{3i+2} + 2^{511-i}$. We utilize the collision attack on 6.5-round `Streebog-256` compression function in [MLHL14] to construct the multicollisions. Their attack uses the rebound attack [MRST09] and the Super-SBox technique [GP10, LMR+09], the differential trail is shown in Figure 9. We

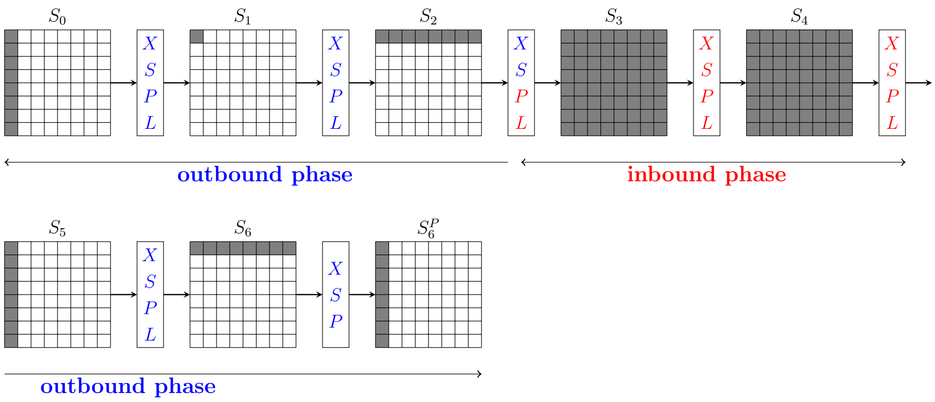


Figure 9: Collision attack on 6.5-round `Streebog-256` compression function

do not describe the attack concretely and just show the time and memory complexity of the attack are 2^{120} and 2^{64}. The 4-multicollisions can be generated by the following steps [MLHL15b]:

1. From the chaining value h_{3i}, we use the collision attack to find a collision pair (m_{3i}, m'_{3i}) on 6.5-round `Streebog-256` with the cost of 2^{120} time and 2^{64} memory.
2. From h_{3i+1}, we randomly choose m_{3i+1} and compute the value of h_{3i+2}.
3. From h_{3i+2}, we find a collision pair (m_{3i+2}, m'_{3i+2}) on 6.5-round `Streebog-256`. Then we check whether $m_{3i} + m_{3i+2} = m'_{3i} + m'_{3i+2} + 2^{511-i}$ holds. Note that the difference between m_{3i} and m'_{3i} and the difference between m_{3i+2} and m'_{3i+2} lie in the same active column, thus this equation holds with probability 2^{-64}.
4. If all pairs of step 3 can not make the equation holds, we go back to step 2 and choose another m_{3i+2} then redo step 3.

The position of the active cell can be placed in any column of the state, so we can generate the 4-multicollisions for any item 2^{511-i} where $i = 0, 1, \cdots, 511$. Therefore, we repeat the above procedure 512 times, the 2^{1024}-multicollisions can be constructed with the cost of $512 \times (2^{120} + 2^{120+64}) \approx 2^{193}$ time and 2^{64} memory.

Phase 2: Invert the Output Transformation. After we know the value of h_{1536}, we randomly choose another message block m_{1536} which satisfies padding. Hence, the message bit length $|M|$ is known and we can compute the value of h_{1538}. Then we need to find Σ such that $H(M) = g(0, h_{1538}, \Sigma)$ which can be achieved by the preimage attack on `Streebog-256` compression function.

Different from the 7.5-round attack in Section 5.3, we find a preimage attack on 6.5-round `Streebog-256` compression function in Figure 10, whose neutral words are all from the internal state. There are 30 ■ cells and 6 ■ cells in the starting state Y_3, so $\lambda^+ = 30$ and $\lambda^- = 6$. From Y_2 to Z_1, it consumes 16 degrees of freedom of ■ cells and the ■ cells are not consumed in the attack. Therefore, $\text{DoF}^+ = 14$ and $\text{DoF}^- = 6$. The matching point is between Z_4 and Y_5, we can get a filter of DoM = 16 cells. We guess 8 cells of Y_6 in the backward computation, so $\sigma^- = 8$. If we use Algorithm 2 to build the table V and U to get the solution spaces of neutral words, the time complexity will be $(2^8)^{30} = 2^{240}$

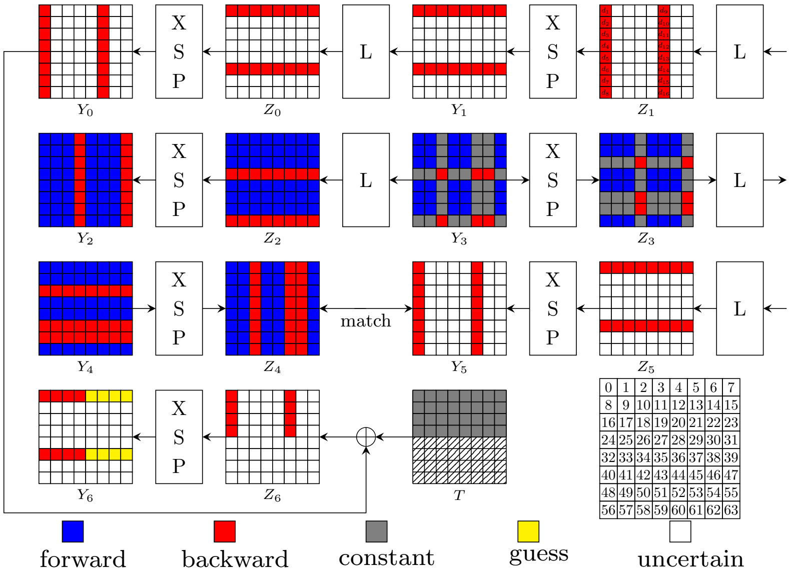


Figure 10: 6.5-round preimage attack on `Streebog`-256 compression function

because $\lambda^+ = 30$ and it will be the lower limit of the whole attack. Therefore, we use the method which is similar to Algorithm 5 to compute the solution space of ■ cells and the procedure is shown in Algorithm 8. The computation of Algorithm 8 between line 3

Algorithm 8: Compute the solution space of Blue cells in Figure 10

Input: $\mathfrak{c}^+ = (d_1, \cdots, d_{16})$ (marked in Z_1)

Output: $V[\mathfrak{c}^+]$

1 Fix the Gray cells in Y_3.

2 $V[\mathfrak{c}^+] = \emptyset$.

3 **for** *All possible values of* $Y_3^{row_i}[0,1,3,4,7] (i = 0,1,2,4)$ **do**

4 (a). $(Z_2^{row_i})^T = L^{-1} \cdot (Y_3^{row_i})^T$, $Y_2^{col_i} = (S \circ X)^{-1}(Z_2^{row_i}) (i = 0,1,2,4)$.

5 (b). $L^{-1} \cdot (Y_2^{row_0})^T = (Z_1^{row_0})^T$, namely,

$$L^{-1}(Y_2[0], Y_2[1], Y_2[2], 0, Y_2[4], Y_2[5], Y_2[6], 0)^T = (d_1, \text{-}, \text{-}, \text{-}, \text{-}, d_9, \text{-}, \text{-})^T$$

the values of $Y_2[5], Y_2[6]$ can be uniquely determined because L is a MDS matrix.

6 (c). Solving $L^{-1} \cdot (Y_2^{row_i})^T = (Z_1^{row_i})^T$ $(i = 1, 2, \cdots, 7)$, $Y_2^{col_5}, Y_2^{col_6}$ can be uniquely determined.

7 (d). $Z_2^{row_5} = S \circ X(Y_2^{col_5})$, $Z_2^{row_6} = S \circ X(Y_2^{col_6})$.

8 **if** $Y_3^{row_5}[2,5,6] = (L \cdot (Z_2^{row_5})^T)[2,5,6]$, $Y_3^{row_6}[2,5,6] = (L \cdot (Z_2^{row_6})^T)[2,5,6]$ **then**

9 Store the values of $Y_3^{row_i}[0,1,3,4,7] (i = 0,1,2,4,5,6)$ in $V[\mathfrak{c}^+]$.

---

and line 9 will be repeated $(2^8)^{20} = 2^{160}$ times, so the time complexity of the procedure is 2^{160} to build the table $V[\mathfrak{c}^+]$. The probability of $Y_3^{row_5}[2,5,6] = (L \cdot (Z_2^{row_5})^T)[2,5,6]$, $Y_3^{row_6}[2,5,6] = (L \cdot (Z_2^{row_6})^T)[2,5,6]$ hold is 2^{-48}, so there are about 2^{112} elements in $V[\mathfrak{c}^+]$ in average. We give Algorithm 9 to find preimage attack on the 6.5-round compression function of `Streebog-256`. The procedure of building $V[\mathfrak{c}^+]$ repeats 2^{16} times, so it costs 2^{176} time in total. For the MITM procedure, we can get the time complexity is about 2^{209}. The whole time complexity is about 2^{209}, and the memory complexity is about 2^{160}. Therefore, we can get a Σ such that $H(M) = g(0, h_{1538}, \Sigma)$ with time of 2^{209} and memory of 2^{160}.

Phase 3: Generate the Preimage. After we get the checksum value Σ, we need to find message blocks which satisfy Σ. We use the same method as in [MLHL14] to find the message blocks.

1. Let $Q = \Sigma - m_{1536}$, M be an empty message.
2. Compute $C = Q - (\sum_{i=0}^{511}(m_{3i} + m_{3i+2})) = \sum_{i=0}^{511} k_i 2^i \ (k_i \in \{0, 1\})$.
3. For $i = 0$ to 511:
   (a) If $k_i = 0$, then $M \leftarrow M||m_{3i}||m_{3i+1}||m_{3i+2}$.
   (b) If $k_i = 1$, then $M \leftarrow M||m'_{3i}||m_{3i+1}||m'_{3i+2}$
4. $M \leftarrow M||m_{1536}$

After the three phases, we can know M, which contains 1537 blocks, corresponds to the desired checksum and M is a preimage of $H(M)$. The time complexity of the preimage attack on `streebog-256` is 2^{209} and the memory complexity is 2^{160}.

Algorithm 9: The MITM preimage attack on 6.5-round `Streebog-256` compression function

```
Y3[2,5,6,10,13,14,18,21,22,34,37,38,42,45,46,50] ← 0.
c+ = (d1, ··· , d16) ← 0.
for each value of Y3[53,54] do
    Call Algorithm 8 build table V[c+] which stores the solution space of ■ cells.
    for each value of Y3[24,25,27,28,31,56,57,59,60,63] do
        for each value in V[c+] do
            Compute forward to the matching point Z4, store the values in L.
        for each value of ■ cells in Y3 and each guess of □ cells in Y6 do
            Compute forward to get the values of ■ cells in Z4 and compute backward to
             the matching point Y5 to match.
            Use the matching pairs to compute and check if the guessed values Y_-^ENC are
             correct.
            if The guessed values Y_-^ENC are correct then
                Test the full preimage.
                if The full preimage is found then
                    Output and stop.
```

8 Conclusion

In [DHS+21], Dong et al. introduced the table-based technique to solve the problem of nonlinearly constrained neutral words in the MITM preimage attacks. Based on their work, we further consider the complex situation which Sasaki et al.'s [SWWW12] guess-and-determine approach is used in the MITM preimage attacks. Moreover, based on previous automatic tools for MITM preimage attack, we propose a new one taking the two techniques into consideration. Finally, we improve the preimage attacks against `Streebog-512` by one more round and also reduce the time complexity of the 7.5-round preimage attack on `Streebog-512` and 6.5-round preimage attack on `Streebog-256`.

Acknowledgements

We would like to thank André Schrottenloher and the anonymous reviewers from ToSC for their detailed comments. This paper was supported by the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008), the National Key Research and Development Program of China (Grant Nos. 2018YFA0704701 and 2017YFA0303904), the National Natural Science Foundation of China (Grant Nos. 61902207, 62032014, 62072270, and 62072207), Shandong Province Key Research and Development Project(Grant Nos. 2020ZLYS09 and 2019JZZY010133), the Fundamental Research Funds for the Central Universities, CAS Project for Young Scientists in Basic Research, the Natural Science Foundation of Shanghai (Grant No. 19ZR1420000), and Open Foundation of Network and Data Security Key Laboratory of Sichuan Province (University of Electronic Science and Technology of China).

References

[AGM+09] Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 578–597. Springer, Heidelberg, December 2009.

[AMM09] Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage attacks on 3-pass HAVAL and step-reduced MD5. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 120–135. Springer, Heidelberg, August 2009.

[AS09a] Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 70–89. Springer, Heidelberg, August 2009.

[AS09b] Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008*, volume 5381 of *LNCS*, pages 103–119. Springer, Heidelberg, August 2009.

[AY14] Riham AlTawy and Amr M. Youssef. Preimage attacks on reduced-round Stribog. In David Pointcheval and Damien Vergnaud, editors, *AFRICACRYPT 14*, volume 8469 of *LNCS*, pages 109–125. Springer, Heidelberg, May 2014.

[BD07] Eli Biham and Orr Dunkelman. A framework for iterative hash functions - HAIFA. Cryptology ePrint Archive, Report 2007/278, 2007. https://eprint.iacr.org/2007/278.

[BDG+19] Zhenzhen Bao, Lin Ding, Jian Guo, Haoyang Wang, and Wenying Zhang. Improved meet-in-the-middle preimage attacks against AES hashing modes. *IACR Trans. Symm. Cryptol.*, 2019(4):318–347, 2019.

[BDG+21] Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on AES-like hashing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 771–804. Springer, Heidelberg, October 2021.

[BGST21] Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. MITM meets guess-and-determine: Further improved preimage attacks against AES-like hashing. *IACR Cryptol. ePrint Arch.*, page 575, 2021.

[BKR11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 344–371. Springer, Heidelberg, December 2011.

[BPU16] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 372–402. Springer, Heidelberg, May 2016.

[CNV13] Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. Sieve-in-the-middle: Improved MITM attacks. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 222–240. Springer, Heidelberg, August 2013.

[DD13] Vasily Dolmatov and Alexey Degtyarev. GOST R 34.11-2012: Hash function. *RFC*, 6986:1–40, 2013.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 157–184. Springer, Heidelberg, August 2016.

[DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 371–387. Springer, Heidelberg, May 2013.

[DH77] Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.

[DHS+21] Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 278–308, Virtual Event, August 2021. Springer, Heidelberg.

[DKS10] Orr Dunkelman, Nathan Keller, and Adi Shamir. Improved single-key attacks on 8-round AES-192 and AES-256. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 158–176. Springer, Heidelberg, December 2010.

[DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 116–126. Springer, Heidelberg, February 2008.

[EFK15] Thomas Espitau, Pierre-Alain Fouque, and Pierre Karpman. Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 683–701. Springer, Heidelberg, August 2015.

[FM15] Thomas Fuhr and Brice Minaud. Match box meet-in-the-middle attack against KATAN. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 61–81. Springer, Heidelberg, March 2015.

[GJL+14] Jian Guo, Jérémy Jean, Gaëtan Leurent, Thomas Peyrin, and Lei Wang. The usage of counter revisited: Second-preimage attack on new russian standardized hash function. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 195–211. Springer, Heidelberg, August 2014.

[GLRW10] Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 56–75. Springer, Heidelberg, December 2010.

[GOS12] Information protection and special communications of the federal security service of the russian federation: GOST R 34.11-2012, information technology cryptographic data security hashing function, 2012.

[GOSan] Information protection and special communications of the federal security service of the russian federation: GOST R 34.11-94, information technology cryptographic data security hashing function, 1994, (In Russian).

[GP10] Henri Gilbert and Thomas Peyrin. Super-sbox cryptanalysis: Improved attacks for AES-like permutations. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 365–383. Springer, Heidelberg, February 2010.

[GSY15] Jian Guo, Chunhua Su, and Wun-She Yap. An improved preimage attack against HAVAL-3. *Inf. Process. Lett.*, 115(2):386–393, 2015.

[HKS10] Deukjo Hong, Bonwook Koo, and Yu Sasaki. Improved preimage attack for 68-step HAS-160. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09*, volume 5984 of *LNCS*, pages 332–348. Springer, Heidelberg, December 2010.

[IPS13] Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-birthday distinguishers for hash functions - collisions beyond the birthday bound can be meaningful. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 504–523. Springer, Heidelberg, December 2013.

[ISO18] ISO/IEC 10118-3:2018 it security techniques — hash-functions — part 3: Dedicated hash-functions, 2018.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 306–316. Springer, Heidelberg, August 2004.

[KK12] Simon Knellwolf and Dmitry Khovratovich. New preimage attacks against reduced SHA-1. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 367–383. Springer, Heidelberg, August 2012.

[KLMR16] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - Efficient short-input hashing for post-quantum applications. *IACR Trans. Symm. Cryptol.*, 2016(2):1–29, 2016. https://tosc.iacr.org/index.php/ToSC/article/view/563.

[LMR+09] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 126–143. Springer, Heidelberg, December 2009.

[MLHL14] Bingke Ma, Bao Li, Ronglin Hao, and Xiaoqian Li. Improved cryptanalysis on reduced-round GOST and Whirlpool hash function. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14*, volume 8479 of *LNCS*, pages 289–307. Springer, Heidelberg, June 2014.

[MLHL15a] Bingke Ma, Bao Li, Ronglin Hao, and Xiaoqian Li. Cryptanalysis of reduced-round Whirlwind. In Ernest Foo and Douglas Stebila, editors, *ACISP 15*, volume 9144 of *LNCS*, pages 20–38. Springer, Heidelberg, June / July 2015.

[MLHL15b] Bingke Ma, Bao Li, Ronglin Hao, and Xiaoqian Li. Improved (pseudo) preimage attacks on reduced-round GOST and Grøstl-256 and studies on several truncation patterns for AES-like compression functions. In Keisuke Tanaka and Yuji Suga, editors, *IWSEC 15*, volume 9241 of *LNCS*, pages 79–96. Springer, Heidelberg, August 2015.

[MPR08a] Florian Mendel, Norbert Pramstaller, and Christian Rechberger. A (second) preimage attack on the GOST hash function. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 224–234. Springer, Heidelberg, February 2008.

[MPR+08b] Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak, and Janusz Szmidt. Cryptanalysis of the GOST hash function. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 162–178. Springer, Heidelberg, August 2008.

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

[Per19] Léo Perrin. Partitions in the S-box of Streebog and Kuznyechik. *IACR Trans. Symm. Cryptol.*, 2019(1):302–329, 2019.

[SA08] Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 253–271. Springer, Heidelberg, December 2008.

[SA09] Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 134–152. Springer, Heidelberg, April 2009.

[Sas11] Yu Sasaki. Meet-in-the-middle preimage attacks on AES hashing modes and an application to Whirlpool. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 378–396. Springer, Heidelberg, February 2011.

[Sas18] Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to GIFT. In Atsuo Inomata and Kan Yasuda, editors, *IWSEC 18*, volume 11049 of *LNCS*, pages 227–243. Springer, Heidelberg, September 2018.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 158–178. Springer, Heidelberg, December 2014.

[SS22] André Schrottenloher and Marc Stevens. Simplified mitm modeling for permutations: New (quantum) attacks. Cryptology ePrint Archive, Report 2022/189, 2022. https://ia.cr/2022/189.

[SWWW12] Yu Sasaki, Lei Wang, Shuang Wu, and Wenling Wu. Investigating fundamental security requirements on Whirlpool: Improved preimage and collision attacks. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 562–579. Springer, Heidelberg, December 2012.

[WFW⁺12] Shuang Wu, Dengguo Feng, Wenling Wu, Jian Guo, Le Dong, and Jian Zou. (Pseudo) preimage attack on round-reduced Grøstl hash function and others. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 127–145. Springer, Heidelberg, March 2012.

[WS10] Lei Wang and Yu Sasaki. Finding preimages of Tiger up to 23 steps. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 116–133. Springer, Heidelberg, February 2010.

[WSK⁺11] Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. (Second) preimage attacks on step-reduced RIPEMD/RIPEMD-128 with a new local-collision approach. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 197–212. Springer, Heidelberg, February 2011.

[WYW13] Zongyue Wang, Hongbo Yu, and Xiaoyun Wang. Cryptanalysis of GOST r hash function. Cryptology ePrint Archive, Report 2013/584, 2013. https://eprint.iacr.org/2013/584.

[ZWW13] Jian Zou, Wenling Wu, and Shuang Wu. Cryptanalysis of the round-reduced GOST hash function. In Dongdai Lin, Shouhuai Xu, and Moti Yung, editors, *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers*, volume 8567 of *Lecture Notes in Computer Science*, pages 309–322. Springer, 2013.

Appendix

A Preimage Attack on 7.5-round `Streebog-512`'s Compression Function

We find a preimage attack on 7.5-round `Streebog-512` compression function as shown in Figure 11. The starting state are Y_3 and K_4. The matching point is between Z_5 and W_5, and there are 24 cells matching, so DoM = 24. In Y_3, there are 64 $\textcolor{blue}{\blacksquare}$ cells. In K_4, there are 16 $\textcolor{red}{\blacksquare}$ cells and 48 $\textcolor{gray}{\blacksquare}$ cells, so $\lambda^+ = 64$ and $\lambda^- = 16$. In the backward chunk, there are 15 guessed cells which are represented by $\textcolor{yellow}{\blacksquare}$ in Y_1.

From Y_3 to Z_2, the contraints in Equation (16) are applied, where $(c_1, c_2, \cdots, c_{40})$ are constants which are marked in Z_2. It consumes 40-cell degrees of freedom for $\textcolor{blue}{\blacksquare}$ cells, so $\mathrm{DoF}^+ = 64 - 40 = 24$. While the $\textcolor{red}{\blacksquare}$ cells are not consumed in this attack. We can easily know the contraints on $\textcolor{blue}{\blacksquare}$ cells are linearly, so we can use Algorithm 1 to mount the MITM attack. The procedure is shown in Algorithm 10.

$$\begin{pmatrix} c_1 & c_6 & c_{11} & c_{16} & c_{21} & c_{26} & c_{31} & c_{36} \\ c_2 & c_7 & c_{12} & c_{17} & c_{22} & c_{27} & c_{32} & c_{37} \\ - & - & - & - & - & - & - & - \\ - & - & - & - & - & - & - & - \\ c_3 & c_8 & c_{13} & c_{18} & c_{23} & c_{28} & c_{33} & c_{38} \\ - & - & - & - & - & - & - & - \\ c_4 & c_9 & c_{14} & c_{19} & c_{24} & c_{29} & c_{34} & c_{39} \\ c_5 & c_{10} & c_{15} & c_{20} & c_{25} & c_{30} & c_{35} & c_{40} \end{pmatrix} = L^{-1} \begin{pmatrix} Y_3[0] & Y_3[8] & Y_3[16] & Y_3[24] & Y_3[32] & Y_3[40] & Y_3[48] & Y_3[56] \\ Y_3[1] & Y_3[9] & Y_3[17] & Y_3[25] & Y_3[33] & Y_3[41] & Y_3[49] & Y_3[57] \\ Y_3[2] & Y_3[10] & Y_3[18] & Y_3[26] & Y_3[34] & Y_3[42] & Y_3[50] & Y_3[58] \\ Y_3[3] & Y_3[11] & Y_3[19] & Y_3[27] & Y_3[35] & Y_3[43] & Y_3[51] & Y_3[59] \\ Y_3[4] & Y_3[12] & Y_3[20] & Y_3[28] & Y_3[36] & Y_3[44] & Y_3[52] & Y_3[60] \\ Y_3[4] & Y_3[13] & Y_3[21] & Y_3[29] & Y_3[37] & Y_3[45] & Y_3[53] & Y_3[61] \\ Y_3[6] & Y_3[14] & Y_3[22] & Y_3[30] & Y_3[38] & Y_3[46] & Y_3[54] & Y_3[62] \\ Y_3[7] & Y_3[15] & Y_3[23] & Y_3[31] & Y_3[39] & Y_3[47] & Y_3[55] & Y_3[63] \end{pmatrix}. \tag{16}$$

Algorithm 10: The MITM preimage attack on 7.5-round `Streebog-512` compression function

- Fix all $\textcolor{gray}{\blacksquare}$ cells of K_4 to 0.
- $(c_{25}, c_{26}, \cdots, c_{40}) \leftarrow 0$.
- **for** $\mathfrak{c}^+ = (c_1, c_2, \cdots, c_{24}, 0, 0, \cdots, 0) \in \mathbb{F}_2^{8\times 24}$ **do**
  - **for** $(Y_3[0], Y_3[1], \cdots, Y_3[24]) \in \mathbb{F}_2^{8\times 24}$ **do**
    - Solve Equation (16) to get the solution of $\textcolor{blue}{\blacksquare}$ and compute forward to get the values of $\textcolor{blue}{\blacksquare}$ in Z_5 and store them in a table $L[\ ]$.
  - **for** *all* $\textcolor{red}{\blacksquare}$ *cells of* $K_4 \in \mathbb{F}_2^{8\times 16}$ *and* $\mathcal{Y}_{-}^{\texttt{ENC}} \in \mathbb{F}_2^{8\times 15}$ *(*$\textcolor{yellow}{\blacksquare}$ *cells of* Y_1*)* **do**
    - Compute backward to get the values of $\textcolor{red}{\blacksquare}$ in W_5 and search L to find matching.
    - Use the matching pairs to compute and check if the guessed values $\mathcal{Y}_{-}^{\texttt{ENC}}$ are correct.
    - **if** *The guessed values* $\mathcal{Y}_{-}^{\texttt{ENC}}$ *are correct* **then**
      - Test the full preimage.
      - **if** *The full preimage is found* **then**
        - Output and stop.

Complexity. The time complexity is about 2^{441} and the memory complexity is 2^{192}.

B Rules of operations

The rules `XOR-RULE` introduced in [BDG+21] are used to build the rules of `AddRoundKey` of `AES`-like hashing. `XOR-RULE` is different in different directions, the coloring patterns are shown in Figure 12.

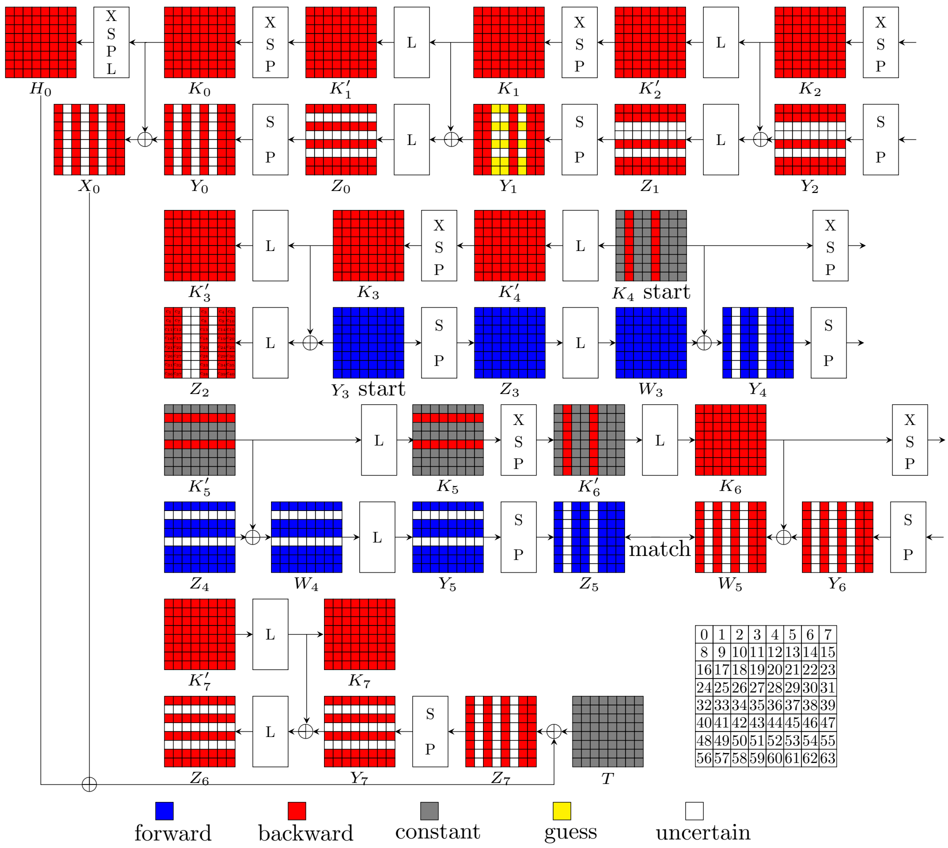


Figure 11: 7.5-round preimage attack on `Streebog-512` compression function

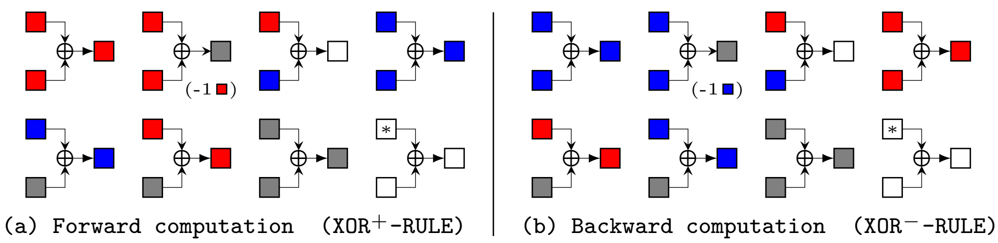


Figure 12: Rules for XOR ("*" represents the cell can be any color, "-1" means reducing the degrees of freedom by one) [BDG+21]

MC-RULE which are the rules of MixColumns can be built similarly. Some valid coloring schemes of MC-RULE in the forward computation (denoted by MC$^+$-RULE) are shown in Figure 13. For more details of these rules, we refer to the paper [BDG$^+$21].

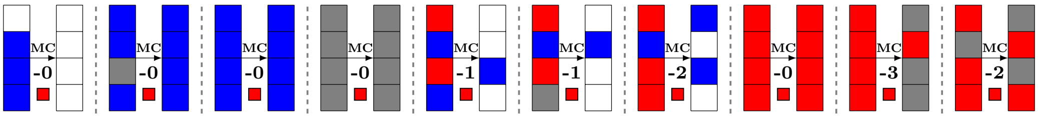


Figure 13: Some valid coloring schemes of MC$^+$-RULE [BDG$^+$21]