IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2024, No. 2, pp. 68–84. DOI:10.46586/tosc.v2024.i2.68-84

Cryptanalysis of Full-Round BipBip

Jinliang Wang[1,4], Christina Boura[2], Patrick Derbez[3], Kai Hu[1,4], Muzhou Li[1] and Meiqin Wang[1,4,5]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, Shandong, China
jinliangwang@mail.sdu.edu.cn,kai.hu@sdu.edu.cn,muzhouli@mail.sdu.edu.cn,mqwang@sdu.edu.cn

[2] Université Paris-Saclay, Université de Versailles, Centre National de la Recherche Scientifique (CNRS), Laboratoire de mathématiques de Versailles, 78000, Versailles, France
christina.boura@uvsq.fr

[3] Univ Rennes, Inria, Centre National de la Recherche Scientifique (CNRS), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Rennes, France
patrick.derbez@irisa.fr

[4] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[5] Quan Cheng Shandong Laboratory, Jinan, China

Abstract. `BipBip` is a low-latency tweakable block cipher proposed by Belkheyar et al. in 2023. It was designed for pointer encryption inside a new memory safety mechanism called Cryptographic Capability Computing (C^3). `BipBip` encrypts blocks of 24 bits using a 40-bit tweak and a 256-bit master key and is composed of 11 rounds. In this article, we provide a Demirci-Selçuk Meet-in-the-Middle (DS-MITM) attack against the 11-round (full) variant that breaks the security claim of the designers.

Keywords: `BipBip` · Demirci-Selçuk meet-in-the-middle cryptanalysis · low latency

1 Introduction

Memory safety vulnerabilities, such as access errors or memory leaks are considered to be the most common security problems in computer systems. In 2021, a group of researchers from Intel Labs proposed a new memory safety mechanism, called Cryptographic Capability Computing (C^3) [LRD+21]. This mechanism was particularly interesting as it was the first one to be stateless, not requiring extra storage for metadata. In practice, C^3 encrypts 64-bit pointers. However, only 24 bits of each pointer are encrypted and the remaining 40 bits serve as a tweak. While each pointer is only encrypted once upon memory allocation, it needs to be decrypted within the processor core whenever it is dereferenced. Therefore, to guarantee a good system performance it is crucial for the decryption to be low-latency.

In 2023, Belkheyar, Daemen, Dobraunig, Ghosh and Rasoolzadeh designed `BipBip` [BDD+23], a low-latency tweakable block cipher, to be used inside the C^3 mechanism. `BipBip` uses 24-bit blocks and 40-bit tweaks while the master key is 256-bit long. It is an 11-round iterative design that permits for ASIC implementations with a latency of 3 cycles at a 4.5 GHz clock frequency on a modern 10 nm CMOS technology. Because of the special format and performance requirements, `BipBip` has several particular design features. First, it has a very small block size of 24 bits only. Another particularity is that it has a very long key compared to the block size that needs to be absorbed in very few rounds only due to the low latency requirements. This is done with the help of a wide, non-linear tweakey schedule.

Received: 2024-03-01 Accepted: 2024-05-01 Published: 2024-06-18

The designers of BipBip claimed a security of 96 bits against attackers and provided a preliminary security analysis inside the design document. In particular, they gave an extended and very complete analysis of the resistance of the cipher against differential and linear attacks. This analysis shows that BipBip is not vulnerable to these two classical cryptanalysis techniques. Among the other attacks, the one that showed to best apply against BipBip was the Demirci-Selçuk Meet-in-the-Middle (DS-MITM) cryptanalysis technique [DS08].

The Demirci-Selçuk Meet-in-the-Middle attack was introduced by Demirci and Selçuk in 2008 and applied to reduced-round versions of the Advanced Encryption Standard (AES). Subsequent enhancements by various researchers [DKS10, DFJ13, DF13, LJW14, LJ16] refined this method, leading to the most effective known attacks on AES variants. The versatility of the DS-MITM approach has been demonstrated through its application to many different ciphers, including PRINCE [DP15], TWINE [BDP15], Camelia [LJWD15, DLJW15], HALFLOOP [LRS23], and SKINNY [SSS+23], among others. In parallel, quantum versions of the attack have been proposed recently [HS18, BNS19] demonstrating the potential of this technique in the field of quantum cryptanalysis.

The application of the DS-MITM to BipBip permitted to its designers to break 9 out of the 11 rounds with a data complexity of 2^6 ciphertexts and a time complexity of 2^{90} encryptions and 2^{90} memory look-ups. While they did not manage to successfully extend this attack to 10 rounds, they wrote that such an extension could eventually be possible by a more careful analysis of the cipher's inner components. No third-party cryptanalysis has been proposed until now against BipBip and the 9-round DS-MITM attack of [BDD+23] remains for the moment the best attack against this cipher.

Determining the exact level of security that BipBip offers in practice is very important as this cipher has very attractive features, offers competitive performances and will very probably be implemented inside the C^3 architecture. Analyzing this cipher is also very challenging, as its structure significantly differs from the traditional tweakable block ciphers, notably by its extremely small block size. This task is even more important as we know today that many lightweight ciphers showed to be less secure than what their designers initially claimed [LAAZ11, TLS16, BDHN23, WNL+23, LAW+23, LIM21, ZCWW23, TI22, SCW23]. This sometimes wrong estimate of the security level lightweight ciphers offer can be explained by the fact that the inner components of these ciphers have most of the time a simpler algebraic expression and more aggressive parameters than mainstream designs. For example, the number of rounds is often chosen in a much less conservative way, leaving potentially a very thin security margin behind.

Our contributions In this article, we describe a Demirci-Selçuk Meet-in-the-Middle (DS-MITM) attack against 11-round BipBip, the first attack against the full version for this cipher. Our attack follows the framework proposed in [DKS10], where the differential enumeration technique (DET) is used to reduce both the time and memory complexities of the precomputation phase. The memory complexity of our attack is $2^{92.25}$ 24-bit blocks while its time complexity and its probability of success depend on the data available to the attacker. Given the full codebook for s different tweaks, corresponding to a data complexity of $s \times 2^{24}$, the time complexity is equivalent to $2^{92.13} + s \times 2^{90.92}$ BipBip encryptions, for a probability of success equal to $1 - (1 - 2^{-48})^{s \times 2^{47}}$. A summary of our attacks for $s = 1$, 2 and 3, as well as a comparison with the previous best cryptanalysis results against BipBip is shown in Table 1. As shown, we are able to generate under a secret key valid (plaintext/ciphertext/tweak) tuples with a probability twice as big as expected. For this reason, our attacks can be considered to break the security claim made by the designers of BipBip, but only under the assumption that the attacker has access to a very fast memory. In any case, our results demonstrate that the security claim is very tight and while they do not present a threat for the use of BipBip in practice, they permit to provide a better understanding of the security properties and the security margin of this cipher.

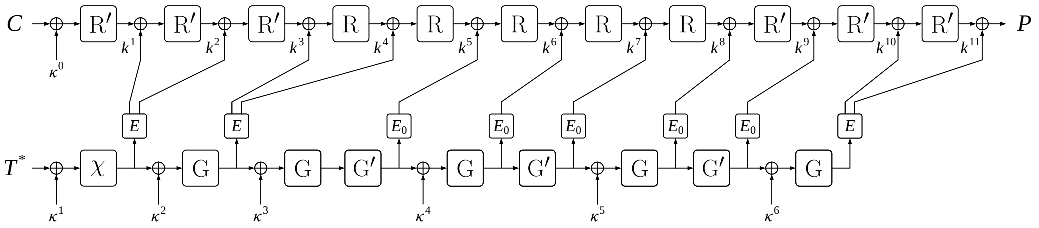
Table 1: Summary of the best cryptanalysis results against `BipBip`. CC stands for Chosen Ciphertexts. Time complexities are evaluated in decryption units, while the memory complexity is given in number of blocks. The parameter s corresponds to the number of tweaks, P_u is the upper bound on the success rate set by the designers (Eq. (2)), and “Proba.” is the success rate of our attack.

Method	Rounds	s	Data	Time	Memory	Proba.	P_u	Ref.
DS-MITM	9	1	2^6 CC	2^{91}	2^{90}	63.21 %	3.12 %	[BDD+23]
DS-MITM with DET	11	1	2^{24} CC	$2^{92.71}$	$2^{92.25}$	39.35 %	20.49 %	Sect. 3
	11	2	2^{25} CC	$2^{93.09}$	$2^{92.25}$	63.21 %	39.93 %	
	11	3	2^{26} CC	$2^{93.39}$	$2^{92.25}$	77.69 %	65.54 %	

The rest of the article is organised as follows. Section 2 describes the `BipBip` cipher and provides a brief introduction to Demirci-Selçuk MITM attacks. Section 3 describes our attack against full-round `BipBip`. Finally, Section 4 discusses the success probability of the attack and how it is related to the security claim of the designers.

2 Preliminaries

2.1 Brief Introduction to `BipBip` and Notations



C
R′
R
P
k^1 k^2 k^3 k^4 k^5 k^6 k^7 k^8 k^9 k^{10} k^{11}
κ^0
E
E_0
T^*
χ
G
G′
κ^1 κ^2 κ^3 κ^4 κ^5 κ^6

Figure 1: High-level structure of `BipBip`. The key schedule is omitted.

`BipBip` is a tweakable block cipher designed to have low-latency decryption when implemented on ASICs [BDD+23]. `BipBip` has a block size of 24 bits, a master key length of 256 bits, and a tweak length of 40 bits. Figure 1 depicts the high-level structure of the data flow in `BipBip`’s decryption. `BipBip` consists of three main parts: the datapath, the tweak schedule, and the key schedule. The key schedule forms the tweak-round keys κ^i, for $1 \leq i \leq 11$ and the whitening key κ^0 by selecting bits from the 256-bit master key K. The tweak schedule processes the tweak T and the tweak-round keys κ^i to derive the data-round keys k^i. The datapath starts with the addition of the whitening key κ^0 to the ciphertext C, followed by the alternating application of the datapath rounds R and R' and data-round key additions with k^i to finally output the plaintext P. As the master key is 256-bit long and there are 11 data-round keys k^i of 24 bits each, these ones can be considered to be (almost) independent.

The datapath uses two round functions, one called *core* round function R and the other one *shell* round function R′. The shell round function has no mixing layer and therefore has lower latency than the core round function. The core round function R consists of an

S-box layer S, a linear mixing layer θ_d, and two bit-shuffles π_1 and π_2. The shell round function R' includes the same S-box layer S and a different bit-shuffle π_3. We have:

$$\mathrm{R} = \pi_2 \circ \theta_d \circ \pi_1 \circ S \quad \text{and} \quad \mathrm{R}' = \pi_3 \circ S.$$

We describe now these inner components in detail.

S-box Layer S The S-box layer of `BipBip`'s datapath is based on a 6-bit S-box, called `BipBipBox` and given in Table 2. The state is divided into 4 words of 6 bits and applies `BipBipBox` to each word in parallel:

$$S: \quad (y_{6i+5}, \dots, y_{6i}) \leftarrow \texttt{BipBipBox}(x_{6i+5}, \dots, x_{6i}), \quad 0 \leq i < 4.$$

Table 2: The S-box (`BipBipBox`) used in `BipBip`. All elements in this table are expressed in hexadecimal. x_0 is the least significant bit.

x_5x_4	$x_3x_2x_1x_0$															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	00	01	02	03	04	06	3e	3c	08	11	0e	17	2b	33	35	2d
1	19	1c	09	0c	15	13	3d	3b	31	2c	25	38	3a	26	36	2a
2	34	1d	37	1e	30	1a	0b	21	2e	1f	29	18	0f	3f	10	20
3	28	05	39	14	24	0a	0d	23	12	27	07	32	1b	2f	16	22

Mixing Layer θ_d The mixing layer multiplies the datapath state with a binary circulant matrix:

$$\theta_d: \quad y_i \leftarrow x_i \oplus x_{i+2 \bmod 24} \oplus x_{i+12 \bmod 24}, \quad 0 \leq i < 24.$$

Bit Shuffles π_1, π_2 and π_3 The 24 bits of the datapath state are permuted by

$$\pi_1: \quad y_i \leftarrow x_{P_1(i)}, \quad \pi_2: \quad y_i \leftarrow x_{P_2(i)}, \quad \pi_3: \quad y_i \leftarrow x_{P_3(i)}, \quad 0 \leq i < 24,$$

where P_1, P_2 and P_3 are permutations of $\mathbb{Z}/24\mathbb{Z}$ specified by the following tables:

$$P_1 = [1, 7, 6, 0, 2, 8, 12, 18, 19, 13, 14, 20, 21, 15, 16, 22, 23, 17, 9, 3, 4, 10, 11, 5],$$
$$P_2 = [0, 1, 4, 5, 8, 9, 2, 3, 6, 7, 10, 11, 16, 12, 13, 17, 20, 21, 15, 14, 18, 19, 22, 23],$$
$$P_3 = [16, 22, 11, 5, 2, 8, 0, 6, 19, 13, 12, 18, 14, 15, 1, 7, 21, 20, 4, 3, 17, 23, 10, 9].$$

From now on and for the sake of convenience, $\pi_2 \circ \theta_d \circ \pi_1$, the linear layer of the core rounds, will be denoted by L_C while π_3, the linear layer of the the shell rounds, will be denoted by L_S.

Tweak Schedule The tweak schedule operates on a 53-bit tweak state. It consists of the application of two types of round functions G and G$'$, and the addition of the tweak-round keys κ_i. The tweak-round keys come from the key schedule that extracts them from the master key.

The tweak schedule receives a 40-bit tweak T and extends it to a 53-bit value T^* with $T^* = T||1||0^{12}$. The tweak schedule round function is composed of five operations, which are a non-linear layer χ, two linear mixing layers θ and θ', and two bit permutations π_4 and π_5. The two types of round functions can be described as

$$\mathrm{G} = \chi \circ \pi_5 \circ \theta \circ \pi_4 \quad \text{and} \quad \mathrm{G}' = \chi \circ \pi_5 \circ \theta' \circ \pi_4,$$

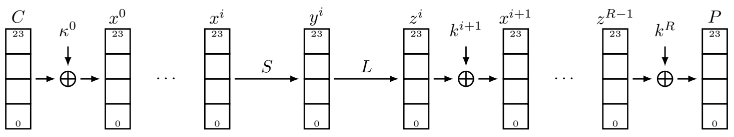


Figure 2: The notations for R-round `BipBip` states and the indices of `BipBip` state bits, where the linear layer L is L_S inside shell rounds and L_C inside core rounds.

where the five operations are

$$\begin{aligned}
\chi:&\quad a_i \leftarrow a_i \oplus (a_{i+1} \oplus 1)a_{i+2}\\
\pi_4:&\quad a_i \leftarrow a_{13i}\\
\theta:&\quad a_i \leftarrow a_i \oplus a_{i+1} \oplus a_{i+8}\\
\theta':&\quad a_i \leftarrow a_i \oplus a_{i+1} \text{ if } 0 \le i < 52 \text{ and } a_{52} \leftarrow a_{52} \text{ if } i = 52\\
\pi_5:&\quad a_i \leftarrow a_{11i}
\end{aligned}$$

where $0 \le i < 53$ and the index is calculated modulo 53.

`BipBip` derives two 24-bit data-round keys k^i and k^{i+1} from the 53-bit internal state of x^i by using two extractor functions E_0 and E_1:

$$k^i = E_0(x^i) = (x^i_0, x^i_2, \ldots, x^i_{46}) \quad \text{and} \quad k^{i+1} = E_1(x^i) = (x^i_1, x^i_3, \ldots, x^i_{47}) \tag{1}$$

Key Schedule The 24-bit whitening key κ^0 and the six 53-bit tweak-round keys $\kappa^1, \ldots, \kappa^6$ are computed from the master key K in the following way:

$$\begin{aligned}
\kappa^0 &= (K_3, K_{3^2}, \ldots, K_{3^{24}}),\\
\kappa^i &= (K_{53i}, K_{53i+1}, \ldots, K_{53i+52}), \quad i = 1, \ldots, 6,
\end{aligned}$$

where the index is computed modulo 256.

Notations For the i-th round of `BipBip` where $0 \le i < 11$, the states before the S-box, linear layer and data-round key addition will be denoted by x^i, y^i and z^i, respectively. The ciphertext and plaintext are denoted by C and P. Thus, $x^0 = C \oplus \kappa^0$ and $P = z^{10} \oplus k^{11}$. For a state s (s can be x^i, y^i, z^i, P or C), Δs represents its difference; s_j is the j-th bit of s and naturally $s_{j_0,\ldots,j_{m-1}}$ are m bits of s. If $j_0, \ldots, j_{m-1}$ are consecutive, we also write them as $s_{j_0-j_{m-1}}$. Finally, s_0 is the least significant bit (LSB) whereas s_{23} is the most significant bit (MSB). An illustration of the above notations is given in Figure 2.

Security Claim The `BipBip` security claim is provided in [BDD+23] in reference to the probability of correctly guessing that a ciphertext C_i maps to a plaintext P_i under a certain tweak T_i for an instance of `BipBip` denoted by E_K, where the master key K has been chosen randomly and the pair (P_i, C_i) has not been queried before. The bound on the probability of a correct guess is

$$p \le \frac{1}{\max(2^{24-\mu} - q_{T_i}, 1)} + \frac{q}{2^{96}} + \frac{t}{2^{96}} + \frac{qt}{2^{120}}, \tag{2}$$

where $\mu = 0.5$, q is the total number of queries to both encryption and decryption of E_K, q_{T_i} is the number of queries to both encryption and decryption of E_K with the tweak of value T_i and t is the computation time with the unit amount of computation equivalent to evaluating E_K.

2.2 Demirci-Selçuk MITM Cryptanalysis

Demirci-Selçuk Meet-in-the-Middle (DS-MITM) cryptanalysis [DS08] is a powerful cryptanalysis technique against block ciphers. In order to automatically and efficiently search for DS-MITM characteristics, Derbez and Fouque [DF13, DF16] introduced a specialized framework, implemented in `C/C++`. This development is part of a broader trend towards automation in the field, with other notable contributions including an integer programming-based approach [LWWZ13] and a constraint programming-based one [SSD+18].

In the DS-MITM attack, a cipher is divided into three consecutive parts E^0, E^1, and E^2 and we denote by s^0, s^1, s^2, s^3 the following internal states:

$$s^0 \xrightarrow{E^0} s^1 \xrightarrow{E^1} s^2 \xrightarrow{E^2} s^3.$$

In the rest of this section, we first provide the definition of the δ-set (and more precisely of the b-δ-set) a central notion in DS-MITM attacks. Then, we describe the basic MITM attack. Finally, we recall the differential enumeration technique, an enhanced method for this cryptanalysis, proposed by Dunkelman, Keller and Shamir in [DKS10] at ASIACRYPT 2010.

Definition 1 (b-δ-set and δ-set [DF16])**.** A b-δ-set is a set of 2^b states in s^1 which are all different in the b active bits and constant in the remaining ones. Whenever the value of b can be easily determined from the context, we will simply write δ-set to refer to this same collection of states.

The basic MITM attack, as described in the seminal paper of [DS08], consists of two phases, the offline phase and the online one. The main idea is to compute the sequence of differences in some b_o bits of the state s^2 from a δ-set, through E^1 during the offline phase and through $(E^2)^{-1} \circ E \circ (E^0)^{-1}$ during the online phase, assuming an access to E. In other words, we want to build the sequence $[\Delta^1 s^2_{b_1,\dots,b_{b_o}}, \dots, \Delta^{2^b-1} s^2_{b_1,\dots b_{b_o}}]$ for an ordered collection of 2^b messages forming a δ-set on the state s_1, and where $\Delta^i x$ stands for the difference on the state x between the message labeled 0 and and the i-th message of the δ-set. Note that in both phases, the sequence of differences is computed by first guessing the value of each involved S-box for one message of the δ-set and then by propagating the differences.

Offline Phase

1. Consider the encryption of a b-δ-set through E^1 by guessing the necessary internal parameters. Deduce the differences in the b_o chosen bits of s^2 and construct a sequence of $2^b - 1$ b_o-bit values.
2. Store the sequences in a table $\mathcal{H}$.

Online Phase

3. Pick a plaintext P and guess the necessary parameters to propagate differences from the b-δ-set to the plaintext through the inverse of E^0. Identify a collection of 2^b plaintexts, containing P and forming a b-δ-set on state s^1.
4. Query the oracle for the corresponding 2^b ciphertexts (s^3).
5. On the ciphertext side, guess the internal parameters, decrypt the above 2^b ciphertexts through $(E^2)^{-1}$, compute the $((2^b-1) \times b_o)$-bit difference sequence and check whether it belongs to $\mathcal{H}$. If not, the guess can be discarded.

Note that during the offline phase it is possible to associate to each sequence the value of the internal parameters that led to it in order to recover them during the online phase. We also emphasize that the procedure only involves guessing state cells but these are related to each other by the data-round keys and thus the internal parameters contain key material as well.

Differential Enumeration Technique. First proposed by Dunkelman et al. in ASIACRYPT 2010 [DKS10], the differential enumeration technique uses a truncated differential characteristic of probability p. The idea is that for a pair following the characteristic, the number of possible values for the internal parameters might be much lower than for a random message. Thus, during the offline phase, we only construct the possible sequences for δ-sets such that at least one message belongs to a pair following the characteristic. This is what we call the differential enumeration phase (DEP). However, in return, around $1/p$ pairs are needed to ensure that at least one follows the differential characteristic. Hence, in the online phase, the adversary first detects a right pair and chooses one of the elements of the right pair to play the role of P in the construction of the δ-set.

Several differential properties of the S-box are usually used to measure the number of deduced values in the differential enumeration technique.

Property 1 (Differential Property of the S-box)**.** *Assume that Δ^{in} and Δ^{out} are respectively the random input and output differences of a bijective S-box S. Then, the equation*

$$S(x) \oplus S(x \oplus \Delta^{in}) = \Delta^{out}$$

has one solution on average.

Property 2 (Differential Property of the S-box with a concrete output difference)**.** *Let S be a bijective n-bit S-box and let Δ^{out} be a specific output difference. Further denote by $\mathcal{S}_{\Delta^{in}}$ the set of all possible input differences leading to Δ^{out} through S. Then, for a specific $\Delta^{in} \in \mathcal{S}_{\Delta^{in}}$, the equation $S(x) \oplus S(x \oplus \Delta^{in}) = \Delta^{out}$ has $\frac{2^n}{|\mathcal{S}_{\Delta^{in}}|}$ solutions on average.*

3 MITM Attack on Full-Round `BipBip`

In this section, we describe our attack against 11-round, i.e., full `BipBip`. This DS-MITM attack was found by the automatic tool of Derbez and Fouque [DF16] and its outline is depicted in Figure 3. Details on how to apply this tool to `BipBip` are given in Appendix A. For the distinguisher part (from x^3 to y^8), the δ-set should be chosen such that the active bits lie in the middle two words of x^3 and the active bits of Δy^2 are in the second and fourth words (from the bottom). More precisely, the δ-set we use for the attack is given as follows.

δ-set for the attack on 11-round `BipBip` We define the δ-set for our attack as a set containing 2^6 `BipBip` states that are all different on the bits 7, 8, 11, 15, 16 and 17 (active bits) and are constant on the remaining (inactive) bits.

The match will be performed on the difference on the bits y^8_{12} and y^8_{13}. There are theoretically $2^{2\times 63} = 2^{126}$ possible sequences of differences in those bits and we will show that, for the δ-set described above, there are much less such sequences, allowing us to distinguish 6 rounds of the cipher.

Let f be a part of the `BipBip` decryption that sends $x^3_{7,8,11,15,16,17}$ to $y^8_{12,13}$, i.e.,

$$f : \mathbb{F}_2^6 \to \mathbb{F}_2^2, \quad x^3_{7,8,11,15,16,17} \mapsto y^8_{12,13}.$$

We obtain the following observation if we apply f to one δ-set.

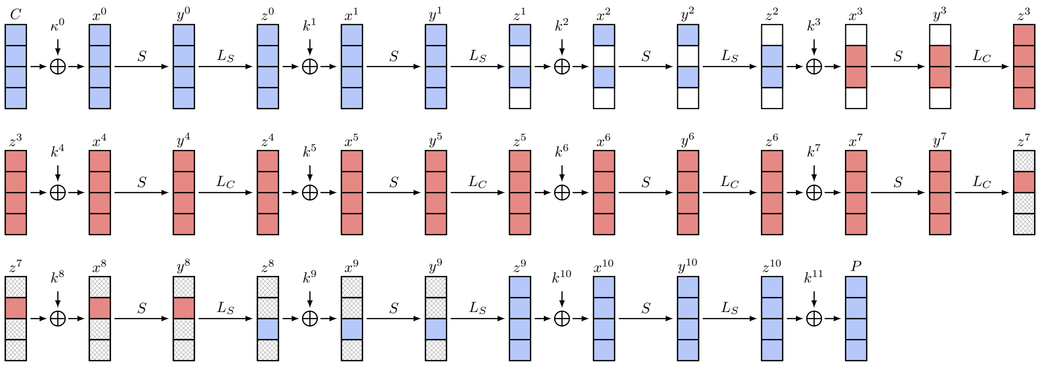


Figure 3: The DS-MITM attack on 11-round `BipBip`. The online phase is from C to z^2 and from z^8 to P. The offline phase is from x^3 to y^8. Blue words correspond to the active words in the online phase while red words correspond to the active words in the offline phase. Finally, white words are inactive while the patterned ones are those not involved in the attack procedure.

Observation 1. *Let $X_i, 0 \le i < 64$ be the 64 different elements in a δ-set. Then, the ordered sequence*

$$f(X_0) \oplus f(X_1), f(X_0) \oplus f(X_2), \dots, f(X_0) \oplus f(X_{63})$$

is fully determined by the following 114-bit parameters,

$$x^3_{6-17},\ x^4,\ x^5,\ x^6,\ x^7,\ x^8_{6-11},$$

where x^3, x^4, x^5, x^6, x^7 and x^8 are intermediate states corresponding to one message of the δ-set.

We can restrict the number of possible values for these parameters from 2^{114} to 2^{90} by applying the differential enumeration technique. To decrease as much as possible the memory complexity, we fully specify the input and output of the truncated differential characteristic used within the technique. We thus set $\Delta x^3 = (\texttt{0x00}, \texttt{0x26}, \texttt{0x38}, \texttt{0x00})$ and $\Delta y^8 = (\texttt{0x00}, \texttt{0x00}, \texttt{0x03}, \texttt{0x00})$.

Observation 2. *If $\Delta x^3 = (\texttt{0x00}, \texttt{0x26}, \texttt{0x38}, \texttt{0x00})$ and $\Delta y^8 = (\texttt{0x00}, \texttt{0x00}, \texttt{0x03}, \texttt{0x00})$, then the 114-bit parameters $x^3_{6-17}, x^4, x^5, x^6, x^7, x^8_{6-11}$ are fully determined by the differences*

$$\Delta x^4, \Delta x^5, \Delta x^6, \Delta x^7, \Delta x^8,$$

which can take 2^{12}, 2^{24}, 2^{24}, 2^{24} and 2^6 different values respectively.

The proof of this observation comes directly from Property 1 which states that each input and output difference of an S-box will lead to one solution in average. Consequently, there are at most 2^{90} possible sequences when we consider all the possible choices of parameters in Observation 2 whereas theoretically there are 2^{126} possibilities. This property can then be used as a distinguisher to mount an attack against full-round `BipBip`.

3.1 Process of the 11-round Attack

Offline Phase We first compute all the 2^{90} 126-bit sequences according to Observations 1 and 2, and store them into a hash table $\mathcal{H}$.

Online Phase

1. By extending Δx^3_{6-17} backwards to ΔC, we see that ΔC is fully active. Thus, we use a structure that contains 2^{24} ciphertexts, i.e., the whole codebook. If we use s tweaks we will need $s \times 2^{24}$ ciphertexts. For each tweak, we decrypt the 2^{24} ciphertexts, get the corresponding plaintexts and by pairing them, we obtain 2^{47} pairs. In total, we will have $s \times 2^{47}$ pairs.

2. On the ciphertext side, for each ciphertext pair with difference ΔC, by enumerating each value of Δy^0 and Δz^1, we can deduce on average one κ^0, k^1 and $k^2_{6-11,18-23}$ such that Δx^3 has the desired value. On the plaintext side, by enumerating each value of Δy^9, we can deduce on average one value for $L_S^{-1}(k^{10})_{6-11}$ and k^{11} such that Δy^8 has the desired value.

3. Once κ^0, k^1, $k^2_{6-11,18-23}$, $L_S^{-1}(k^{10})_{6-11}$ and k^{11} have been deduced, we partially decrypt one ciphertext from the ciphertexts pair to z^2. Then, we partially encrypt the values of the δ-set to the ciphertexts again. Next, we partially encrypt the corresponding plaintexts of the δ-set using $(L_S^{-1}(k^{10}))_{6-11}$ and k^{11}, and construct the sequence given in Observation 1.

4. If the above sequence belongs to $\mathcal{H}$, then the corresponding values of κ^0, k^1, $k^2_{6-11,18-23}$, $L_S^{-1}(k^{10})_{6-11}$ and k^{11} are suggested as a candidate. Otherwise, we discard this key.

Recovering All Data-Round Keys The 126-bit difference sequence has a 2^{-124} sieving ability since we fix 2 bits of them in the process of checking right pairs. After performing our attack, for a fixed tweak, there will remain $2^{90+89-124} = 2^{55}$ values containing 114-bit parameters from the hash table $\mathcal{H}$, 90-bit secret keys in the online phase and the corresponding plaintext-ciphertext pair. Therefore, we can know the values of the 2^6 δ-set elements in y^7 by the 114-bit parameters while their values in z^9 can be obtained by the 90-bit keys. Then, we can guess more keys and use another match point to further sieve the keys. For example, after guessing k^8_{0-5} and $L_S^{-1}(k^{10}_{0-5})$, we can check whether the two δ-sets match in $y^8_{2,5}$. After this step, and since k^3_{6-17} and $L_S^{-1}(k^9)_{2,5,12,13}$ are deduced from state bits after each valid match, we recover the right value for $90+90+12+12+4 = 208$ bits of the data-round keys as no wrong key should survive (we expect $2^{55+12-126} = 2^{-59}$ wrong keys to remain). We finally brute-force the missing 80 bits of the data-round keys.

Time Complexity Analysis Denote by T_d the complexity of deducing the input values to the S-box from the input and output differences, by $T_{\mathcal{H}}$ a memory access to $\mathcal{H}$, and by T_s an S-box operation. As computing the difference sequences in the offline phase requires going through 18 S-boxes, for each guessed difference, we need $18T_d$ to determine the 114 parameters in Observation 1, $(2^6 \times 18)T_s$ to encrypt the δ-set and one $T_{\mathcal{H}}$ to store the parameters in the hash table. Considering that one `BipBip` encryption has 44 S-box operations, the time complexity of the offline phase is

$$\left(\frac{2^{90} \times 18 \times (T_d + 2^6 \times T_s) + 2^{90} \times T_{\mathcal{H}}}{44 \times T_s}\right) \texttt{BipBip}\text{ encryptions.}$$

With a similar analysis, the time complexity of the online phase is

$$\left(s \times \frac{2^{89} \times 15 \times (T_d + 2^6 \times T_s) + 2^{89} \times T_{\mathcal{H}}}{44 \times T_s}\right) \texttt{BipBip}\text{ encryptions.}$$

3.2 Further Reducing the Time and Memory Complexities

The time complexity can be reduced further by cleverly organizing the computations and investigating the `BipBip` S-box. First, as shown in both Algorithms 1 and 2, we can handle the 2^6 elements of the δ-set round by round, improving the time complexity related to building the sequences. Second, we notice that the differential used in the differential enumeration technique limits the possible differences of both Δx^8 and Δy^9. This can be used to amortize the cost of propagating the δ-set.

For the offline phase, $\Delta y^8_{12-17} = \texttt{0x03}$, so Δx^8 can only take 22 among the 64 possibilities, reducing the number of possible values for x^7. Furthermore, Δy^7_{6-11} can assume only 12 different values and we can thus start by guessing it before Δx^8 to reduce a bit more the complexity of the offline phase.

Algorithm 1: Efficient Algorithm for the Offline Phase against Full-Round `BipBip`.

```
Input: $\Delta x^3 = (\texttt{0x00}, \texttt{0x26}, \texttt{0x38}, \texttt{0x00})$, $\Delta y^8_{12-17} = (\texttt{0x00}, \texttt{0x00}, \texttt{0x03}, \texttt{0x00})$
Output: Hash table $\mathcal{H}$ indexed by the 126-bit difference sequence.
1  $\mathcal{H} \leftarrow \emptyset$
2  for $\Delta y^3$ $(2^{12})$ do
3      Deduce $x^3_{6-17}$ and propagate the $\delta$-set to $z^3$
4      for $\Delta y^4$ $(2^{24})$ do
5          Deduce $x^4$ and propagate the $\delta$-set to $z^4$
6          for $\Delta y^5$ $(2^{24})$ do
7              Deduce $x^5$ and propagate the $\delta$-set to $z^5$
8              for $\Delta y^6$ $(2^{24})$ do
9                  Deduce $x^6$ and propagate the $\delta$-set to $z^6$
10                 for $\Delta y^7_{6-11}$ $(12)$ do
11                     Deduce $x^7_{6-11}$ and partially propagate the $\delta$-set to $y^7_{6-11}$
12                     for $\Delta y^7_{0-5,12-23}$ $(22/12)$ do
13                         Deduce $x^7$ and propagate the $\delta$-set to $x^8$
14                         for $x^8_{12-17}$ $(64/22)$ do
15                             Propagate the $\delta$-set to $y^8_{12-17}$ and store the sequence in $\mathcal{H}$.
16 return $\mathcal{H}$
```

The complexity of Algorithm 1, computed in terms of S-box evaluations, is straightforward and dominated by the four last nested loops. After guessing the possible differences, for each active S-box, we need one T_d to deduce the internal state and $2^6 \times T_s$ to encrypt the whole 2^6 elements in the δ-set through this S-box. Finally, 2^{90} hash table write operations are required to store all the parameters into $\mathcal{H}$.

Thus the total time complexity of Algorithm 1 is

$$\frac{2^{84} \times (T_d + 2^6 \times T_s) \times (4 + 12 + 22 \times 3 + 64)}{44 \times T_s} + 2^{90} \times T_{\mathcal{H}}$$
$$\approx \left(2^{91.19} \times \frac{T_d + 2^6 \times T_s}{44 \times T_s} + 2^{90} \times T_{\mathcal{H}}\right) \texttt{BipBip}\text{ encryptions.}$$

For the online phase, Δy^9 can only assume 24 different values. But more importantly, the difference at the input of each S-box of state x^{10} can take only 4, 2, 2 and 4 different values respectively.

Algorithm 2: Efficient Algorithm of Online Phase against Full-Round BipBip..

```
   Input: $\Delta z^2 = (\texttt{0x00}, \texttt{0x26}, \texttt{0x38}, \texttt{0x00}), \Delta z^8 = (\texttt{0x00}, \texttt{0x18}, \texttt{0x00}, \texttt{0x00})$
1  for $s \times 2^{47}$ pairs $(c, c')$ do
2      for $\Delta y^0$, $\Delta y^1$ $(2^{36})$ do
3          Deduce $x^0$, $x^1$ and $x^2$ and identify a $\delta$-set containing $c$.
4          Select the corresponding $2^6$ plaintexts.
5          for $\Delta x^{10}_{6-11}$ (2) do
6              Deduce $x^{10}_{6-11}$ and partially propagate the $\delta$-set
7              for $\Delta x^{10}_{12-17}$ (2) do
8                  Deduce $x^{10}_{12-17}$ and partially propagate the $\delta$-set
9                  for $\Delta x^{10}_{0-5}$ (4) do
10                     Deduce $x^{10}_{0-5}$ and partially propagate the $\delta$-set
11                     for $\Delta x^{10}$ (24/16) do
12                         Deduce $x^{10}_{18-23}$ and partially propagate the $\delta$-set
13                         for $x^9_{6-11}$ (64/24) do
14                             Propagate the $\delta$-set and check whether the sequence
                               belongs to $\mathcal{H}$
```

As for the offline phase, we count the number of S-boxes computed at each step of the algorithm and we obtain:

$$\begin{aligned} & s \times \left(\frac{2^{47} \times 2^{36} \times (T_d + 2^6 \times T_s) \times (10 + 2 + 4 + 16 + 24 + 64)}{44 \times T_s} + 2^{89} \times T_{\mathcal{H}} \right) \\ \approx\ & s \times \left(2^{88.95} \times \frac{T_d + 2^6 \times T_s}{44 \times T_s} + 2^{89} \times T_{\mathcal{H}} \right) \ \texttt{BipBip} \text{ encryptions.} \end{aligned}$$

Complexity Analysis We deduce the S-box input from the input and output differences by an *extended DDT* whose index is the input and output differences and the value is the corresponding values. We experimentally evaluated the ratio of the time complexity of a lookup of the extended DDT and a `BipBipBox` and found it to be around 5:1. Therefore, we measure T_d as $5T_s$. Besides, we assume this attack runs on a machine that has a fast memory access and we can measure each $T_{\mathcal{H}}$ as one `BipBip` encryption.

In the differential enumeration phase, we use a distinguisher of probability 2^{-48} consisting of 2^{-24} from the input difference and 2^{-24} from the output difference. Using the whole codebook for a fixed tweak, we can generate 2^{47} pairs and thus we expect that asking for the full codebook for $s = 2$ tweaks should allow to generate enough pairs to get one that follows the differential.

From the above analysis, the data complexity of the attack is $2 \times 2^{24} = 2^{25}$, the time complexity is $2^{92.19} + 2 \times 2^{90.98} \approx 2^{93.09}$ `BipBip` encryptions and the memory complexity is given by the size of the hash table $\mathcal{H}$ constructed in the offline phase, that is $114 \times 2^{90}/24 \approx 2^{92.25}$, which is measured by the block size of `BipBip`.

4 Impact of our Attack on the Security Claim of `BipBip`

As shown in the last part of Section 2, the designers provided a security claim for `BipBip` based on Eq. (2). In this section, we show that our key recovery attack against `BipBip` breaks this security claim. To this end, we first introduce a procedure for recovering the

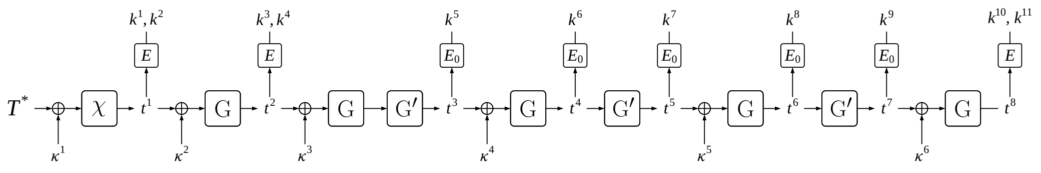


Figure 4: Tweak schedule of `BipBip`. t^i denotes the i-th internal tweak state which generates the data-round keys.

Table 3: The number of known and unknown bits of t^i.

	t^1	t^2	t^3	t^4	t^5	t^6	t^7	t^8
known	48	48	24	24	24	24	24	48
unknown	5	5	29	29	29	29	29	5

master key from the data-round keys. Then, we analyze the success rate and compare it in Section 4.2 to the security claim of `BipBip`.

4.1 Recovering the Master Key from the Data-Round Keys

In this section, we present our method for recovering the master key from the data-round keys. According to Eq. (1), the data-round keys are directly extracted from the internal state of the tweak schedule, thus we already know some bits of each state t^i from the recovered data-round keys. Table 3 summarizes the number of bits obtained from k^i directly (known) and those that still need to be guessed (unknown) for each state t^i.

Besides, since κ^0 is the first data-round key and is directly extracted from the master key K, we know 4, 4, 5, 8, 3 and 5 bits of the tweak-round keys κ^1, κ^2, κ^3, κ^4, κ^5 and κ^6 respectively.

The main idea of our attack is to guess the unknown bits of each tweak state t^i, deduce the value of κ^i, and sieve by the known bits in κ^i obtained from the knowledge of κ^0. Finally, we check if the 62-bit sequence $\kappa^1||\kappa^2_{0-8}$ is the same as $\kappa^5_{44-52}||\kappa^6$, since these two sequences are both extracted from K_{53-114}, where K is the master key.

The process of this attack is as follows:

1. For the 40-bit tweak T used in the attack, compute the 53-bit T^* as $T^* = T||1||0^{12}$.
2. For each of the 2^5 possible values for t^1, deduce κ^1 from both T^* and t^1 and check whether it is compatible with the 4 bits of κ^1 we already know. We expect $2^{5-4} = 2$ candidates to remain for (κ^1, t^1).
3. For each of the 2^5 possible values for t^2, deduce κ^2 from t^1 and t^2 and check the extra 4-bit constraint we have on κ^2. We expect $2 \times 2^5 \times 2^{-4} = 4$ candidates to remain for $(\kappa^1, t^1, \kappa^2, t^2)$
4. Again, we guess t^3, deduce κ^3 from both t^2 and t^3 and we keep only the candidates with the right value on the 5 bits we previously obtained on κ^3. At this point, we have around $4 \times 2^{29-5} = 2^{26}$ possibilities for $(\kappa^1, t^1, \kappa^2, t^2, \kappa^3, t^3)$.
5. Since there is no key addition between t^4 and t^5, this step is a bit different. We first guess t^4, compute t^5 from it and then check t^5 against the value of the 24 bits we already know. We then use both t^3 and t^4 to obtain κ^4 and we finally check the 8-bit constraint on it. Hence, we constructed $2^{26} \times 2^{29-24} \times 2^{-8} = 2^{23}$ candidates for $(\kappa^1, t^1, \kappa^2, t^2, \kappa^3, t^3, \kappa^4, t^4, t^5)$.

6. Similarly, we construct $2^{23} \times 2^{29-24} \times 2^{-3} = 2^{25}$ candidates for $(\kappa^1, t^1, \kappa^2, t^2, \kappa^3, t^3, \kappa^4, t^4, t^5, \kappa^5, t^6, t^7)$.
7. Finally, obtain $2^{25} \times 2^5 \times 2^{-5} = 2^{25}$ candidates for the tweak-round keys. For each of them, we check whether the value of the 62-bit sequence $\kappa^1 || \kappa^2_{0-8}$ is the same as $\kappa^5_{44:52} || \kappa^6$. Since the probability for a wrong key to pass this test is 2^{-62}, only the right one should remain.
8. Recover the master key K by $\kappa^1, \ldots, \kappa^5$.

The complexity of this procedure is around 2^{31} basic operations, which is negligible compared to the complexity of our attack.

4.2 Success Rate Analysis

Assuming that we have recovered the master key, we can randomly select a tweak (except the ones which were used to recover the master key) to conduct the following challenge shown in the security claim of BipBip: for a plaintext P (resp. ciphertext C), map it to the corresponding ciphertext (resp. plaintext).

The designers of BipBip claim that the probability p to win this challenge satisfies the following constraint:

$$p \leq \frac{1}{\max(2^{24-\mu} - q_{T_i}, 1)} + \frac{q}{2^{96}} + \frac{t}{2^{96}} + \frac{qt}{2^{120}},$$

where $\mu = 0.5$, q is the total number of queries to both encryption and decryption of E_K, q_{T_i} is the number of queries to both encryption and decryption of E_K with the tweak of value T_i and t is the computation time with the unit amount of computation equivalent to evaluating E_K.

With the knowledge of the master key, this challenge can be solved with probability 1. Hence, the success rate of our attack corresponds to the probability of recovering the master key, which is itself equal to the success probability of the 11-round DS-MITM attack described in Section 3.

The success probability of our attack is exactly the probability to obtain a pair satisfying the differential used with the differential enumeration technique. Since our attack involves a differential of probability 2^{-48}, using the whole codebook for s tweaks allows to generate $s \times 2^{47}$ pairs and thus the probability that at least one follows the differential is

$$1 - (1 - 2^{-48})^{2^{47}s} \approx 1 - e^{-\frac{s}{2}}.$$

In our attack, the data complexity is $q = s \times 2^{24}$, the time complexity is $t = 2^{92.13} + s \times 2^{90.92}$ and q_{T_i} is zero since we do not query anything under this tweak. Thus, the success upper bound in the BipBip design document is

$$\begin{aligned} p_u &= \frac{1}{\max(2^{23.5} - q_{T_i}, 1)} + \frac{q}{2^{96}} + \frac{t}{2^{96}} + \frac{qt}{2^{120}} \\ &= \frac{1}{2^{23.5}} + \frac{s \times 2^{24}}{2^{96}} + \frac{2^{92.19} + s \times 2^{90.98}}{2^{96}} + \frac{(s \times 2^{24}) \times (2^{92.19} + s \times 2^{90.98})}{2^{120}} \\ &\approx \frac{1+s}{2^{3.81}} + \frac{s+s^2}{2^{5.02}}. \end{aligned}$$

To achieve a valid attack, we need to make sure our success rate is higher than the upper bound of probability in the BipBip design document, which means

$$1 - e^{-\frac{s}{2}} > \frac{1+s}{2^{3.87}} + \frac{s+s^2}{2^{5.08}},$$

It can be verified that when the number of tweaks s is 1, 2 or 3, the above inequality holds.

1. When $s = 1$, our success rate is 39.35% and the upper bound in the `BipBip` design document is 20.46%;
2. When $s = 2$, our success rate is 63.21% and the upper bound in the `BipBip` design document is 39.93%;
3. When $s = 3$, our success rate is 77.69% and the upper bound in the `BipBip` design document is 65.54%;

5 Conclusion

In this paper we presented the first attack against 11-round `BipBip`, the full version of this cipher. While the core of our attack was automatically found by an existing tool, turning it into a valid attack was a complex task, involving clever procedures to compute the sequence of differences through the cipher, looking inside the S-box to restrict the number of possible values of several parameters and inverting the robust key schedule of `BipBip`. Our results show that the security claim made by the designers of `BipBip` was too tight and that there is no security margin left behind. Our results also highlight the importance of optimally organizing computations inside algorithms, in particular to avoid redundant ones. Indeed, such procedures permitted us to decrease the complexity of our basic attack by more than a factor 10.

Acknowledgments

We would like to thank Tyge Tiessen and the other anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. We also thank the organizers of ASK 2023 (the 10th Asian Workshop on Symmetric Key Cryptography), where the collaboration was initiated. Danping Shi checked some initial results; Liu Zhang and Yiran Yao prepared the first version of the figures in this paper, we are grateful for their help. Jinliang Wang, Kai Hu, Muzhou Li and Meiqin Wang are supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704702), the National Natural Science Foundation of China (Grant No. 62032014, U2336207), the Major Basic Research Project of Natural Science Foundation of Shandong Province, China (Grant No. ZR202010220025), Department of Science Technology of Shandong Province (No.SYS202201), Quan Cheng Laboratory (Grant No. QCLZD202301, QCLZD202306). Kai Hu is also supported by the Program of Qilu Young Scholars of Shandong University. Finally, the research of Christina Boura and Patrick Derbez is partially supported through the France 2030 program under grant agreement No. ANR-22-PECY-0010, by the French Agence Nationale de la Recherche through the OREO project under Contract ANR-22-CE39-0015 and the SWAP project under Contract ANR-21-CE39-0012.

References

[BDD$^+$23] Yanis Belkheyar, Joan Daemen, Christoph Dobraunig, Santosh Ghosh, and Shahram Rasoolzadeh. BipBip: A low-latency tweakable block cipher with small dimensions. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):326–368, 2023.

[BDHN23] Christina Boura, Nicolas David, Rachelle Heim Boissier, and María Naya-Plasencia. Better steady than SPEEDY: Full break of SPEEDY-7-192. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 36–66. Springer, 2023.

[BDP15] Alex Biryukov, Patrick Derbez, and Léo Perrin. Differential analysis and meet-in-the-middle attack against round-reduced TWINE. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2015.

[BNS19] Xavier Bonnetain, María Naya-Plasencia, and André Schrottenloher. Quantum security analysis of AES. *IACR Trans. Symmetric Cryptol.*, 2019(2):55–93, 2019.

[DF13] Patrick Derbez and Pierre-Alain Fouque. Exhausting Demirci-Selçuk meet-in-the-middle attacks against reduced-round AES. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 541–560. Springer, 2013.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. 9815:157–184, 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 *Lecture Notes in Computer Science*, pages 371–387. Springer, 2013.

[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 *Lecture Notes in Computer Science*, pages 158–176. Springer, 2010.

[DLJW15] Xiaoyang Dong, Leibo Li, Keting Jia, and Xiaoyun Wang. Improved attacks on reduced-round Camellia-128/192/256. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 59–83. Springer, 2015.

[DP15] Patrick Derbez and Léo Perrin. Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 190–216. Springer, 2015.

[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 *Lecture Notes in Computer Science*, pages 116–126. Springer, 2008.

[HS18] Akinori Hosoyamada and Yu Sasaki. Quantum Demiric-Selçuk meet-in-the-middle attacks: Applications to 6-round generic Feistel constructions. In Dario Catalano and Roberto De Prisco, editors, *SCN 2018*, volume 11035 of *Lecture Notes in Computer Science*, pages 386–403. Springer, 2018.

[LAAZ11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of PRINTcipher: The invariant subspace attack. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2011.

[LAW+23] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Coefficient grouping: Breaking Chaghri and more. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 287–317. Springer, 2023.

[LIM21] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of full LowMC and LowMC-M with algebraic techniques. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 368–401. Springer, 2021.

[LJ16] Rongjia Li and Chenhui Jin. Meet-in-the-middle attacks on 10-round AES-256. *Des. Codes Cryptogr.*, 80(3):459–471, 2016.

[LJW14] Leibo Li, Keting Jia, and Xiaoyun Wang. Improved single-key attacks on 9-round AES-192/256. In Carlos Cid and Christian Rechberger, editors, *FSE 2014*, volume 8540 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2014.

[LJWD15] Leibo Li, Keting Jia, Xiaoyun Wang, and Xiaoyang Dong. Meet-in-the-middle technique for truncated differential and its applications to CLEFIA and Camellia. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 48–70. Springer, 2015.

[LRD+21] Michael LeMay, Joydeep Rakshit, Sergej Deutsch, David M. Durham, Santosh Ghosh, Anant Nori, Jayesh Gaur, Andrew Weiler, Salmin Sultana, Karanvir Grewal, and Sreenivas Subramoney. Cryptographic capability computing. In *MICRO '21*, pages 253–267. ACM, 2021.

[LRS23] Gregor Leander, Shahram Rasoolzadeh, and Lukas Stennes. Cryptanalysis of HALFLOOP block ciphers: Destroying HALFLOOP-24. *IACR Trans. Symmetric Cryptol.*, 2023(4):58–82, 2023.

[LWWZ13] Li Lin, Wenling Wu, Yanfeng Wang, and Lei Zhang. General model of the single-key meet-in-the-middle distinguisher on the word-oriented block cipher. In Hyang-Sook Lee and Dong-Guk Han, editors, *ICISC 2013*, volume 8565 of *Lecture Notes in Computer Science*, pages 203–223. Springer, 2013.

[SCW23] Yimeng Sun, Jiamin Cui, and Meiqin Wang. Improved attacks on LowMC with algebraic techniques. *Cryptology ePrint Archive*, 2023.

[SSD+18] Danping Shi, Siwei Sun, Patrick Derbez, Yosuke Todo, Bing Sun, and Lei Hu. Programming the Demirci-Selçuk meet-in-the-middle attack with constraints. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 3–34. Springer, 2018.

[SSS+23] Danping Shi, Siwei Sun, Ling Song, Lei Hu, and Qianqian Yang. Exploiting non-full key additions: Full-fledged automatic Demirci-Selçuk meet-in-the-middle cryptanalysis of SKINNY. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 67–97. Springer, 2023.

[TI22] Yosuke Todo and Takanori Isobe. Hybrid code lifting on space-hard block ciphers application to Yoroi and SPNbox. *IACR Trans. Symmetric Cryptol.*, 2022(3):368–402, 2022.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack - practical attack on full SCREAM, iSCREAM, and Midori64. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33, 2016.

[WNL+23] Jinliang Wang, Chao Niu, Qun Liu, Muzhou Li, Bart Preneel, and Meiqin Wang. Cryptanalysis of SPEEDY. In Leonie Simpson and Mir Ali Rezazadeh Baee, editors, *ACISP 2023*, volume 13915 of *Lecture Notes in Computer Science*, pages 124–156. Springer, 2023.

[ZCWW23] Zhuolong Zhang, Shiyao Chen, Wei Wang, and Meiqin Wang. Full round distinguishing and key-recovery attacks on SAND-2 (full version). *IACR Cryptol. ePrint Arch.*, page 1697, 2023.

A On the usage of the tool from [DF16]

We provide here the exact parameters we used when applying the tool of [DF16] to `BipBip`. Let n and k be respectively the block and key sizes of a cipher. In the case where the cipher is described at the bit-level, the tool with the default parameters searches for *patterns* with a data complexity strictly below n, a time complexity strictly below k and a memory complexity strictly below $n + k$. It also searches for patterns with a match on only 1 bit and a δ-set of dimension 1. We first used the parameter `-d 24` to allow for attacks requiring the full codebook. We also allowed the time and the memory to be equal to the limit by using the parameters `-t 96 -m 120`. And finally we forced the patterns to filter enough wrong keys by trying the sets of parameters `-i 5 -o 3`, `-i 6 -o 2` and `-i 7 -o 1`. The parameter `-i` sets the minimal dimension of the δ-set while `-o` affects the number of bits on which the match is performed.