A Practical Related-Key Boomerang Attack for the Full MMB Block Cipher

Abstract. The MMB block cipher (Modular Multiplication-based Block cipher) is an iterative block cipher designed by Daemen, Govaerts, and Vandewalle in 1993 as an improvement of the PES and IPES ciphers.
In this paper we present a few new related-key differential characteristics of MMB. These characteristics can be used to form several related-key boomerangs to attack the full MMB and even extended versions with up to 8 rounds (which is two more rounds than in the full MMB). Using 2^{20} adaptive chosen plaintexts and ciphertexts we recover all key bits in $2^{35.2}$ time for the full MMB. Similar complexities also hold for the extended variants of MMB, i.e., we succeed in recovering the full key of 7-round and 8-round MMB using $2^{20.6}$ adaptive chosen plaintexts and ciphertexts in time $2^{35.2}$.
Our attack was experimentally verified. It takes less than 15 minutes on a standard Intel i5 machine to recover the full MMB key.

Key words: MMB, Differential Cryptanalysis, Related-Key Boomerang Attack.

1 Introduction

The MMB block cipher (Modular Multiplication-based Block cipher) is an iterative block cipher designed by Daemen, Govaerts, and Vandewalle [4] as an improvement of the PES and IPES ciphers [9, 10]. The cipher works with blocks of 128 bits and an equal key length. The cipher's non-linearity comes from dividing the block into 4 words of 32 bits each and multiplying them with a known constant mod $2^{32} - 1$ (hence the cipher's name). The cipher consists of 6 rounds without any initialization or finalization steps.

Previously published work on MMB includes two papers [6,12]. Both papers were able to recover the full key of the full MMB. In [12] Wang et al. use a 5-round differential in a 1R attack in $2^{95.91}$ time, 2^{118} chosen plaintexts, and 2^{65} 32-bit memory words to break the full MMB. In [6] Jia et al. present a sandwich attack using 2^{64} time, $2^{66.5}$ chosen plaintexts, and $2^{70.5}$ 32-bit memory words. We summarize these results in Table 1. Hence, all previously published works on MMB have very high time and data complexity, i.e., they cannot be experimentally verified.

In this paper we present a related-key attack that allows an adversary to recover all key bits in time of $2^{35.2}$ using 2^{20} adaptive chosen plaintexts and ciphertexts encrypted under 4 related-keys. We first present two related-key differential characteristics of two and three rounds, respectively, and use them to construct two boomerangs covering 5 rounds of MMB. We then use these 5-round boomerangs to attack the full (6 rounds) MMB. Each of the boomerangs can be used to recover 32 bits of the key. The 64 recovered bits are then further used to recover another 32 bits of the key using a 1R related-key differential attack. The remaining 32 bits are then found by a simple exhaustive search.

After presenting our results, we show that even if MMB was extended to 7 or 8 rounds, it would still be insecure. To prove this claim, we extend the first phase of our attack to extended 7-round and 8-round variants of MMB with the similar complexity. In other words, we show that using $2^{20.6}$ adaptive chosen plaintexts and ciphertexts, encrypted under 6 related keys, in time of about $2^{35.2}$ encryptions, an adversary can recover all key bits of the 128-bit key.

To verify our results experimentally, we implemented the attack on the full (6-round) MMB using a C program. The program generates the required data, encrypts and decrypts it through the presented related-key boomerangs, identifies the right quartets, and recovers the key bits in about 15 minutes on a home PC.

This paper is organized as follows: In Section 2 we give a brief description of the MMB block cipher; Section 3 describes some of the previous work done to analyze MMB; in Section 4 we describe the cryptanalytic techniques we use in the paper. In Section 5 we describe the related-key differential characteristics we use and how we use them to create the related-key boomerangs; Section 6 explains how to use the related-key boomerangs to recover the entire key of the full MMB; Section 7 discusses an extended variants of MMB with 7 and 8 rounds and how to break them, and Section 8 concludes the paper.

2 A Brief Description of MMB

As mentioned before, MMB is an iterative block cipher with a 128-bit block and a 128-bit key. The message and key are each divided into four 32-bit words x_0, x_1, x_2, x_3, and k_0, k_1, k_2, k_3, respectively. In each round, four operations, $\sigma[k^j], \gamma, \eta$, and θ are performed over the state words. Three of the four operations, namely, $\sigma[k^j], \eta$, and θ are involutions (i.e., they are their own inverse).

The key injection operation, $\sigma[k^j]$, XORs the subkey into the message such that $\sigma[k^j](x_0, x_1, x_2, x_3) = (x_0 \oplus k_0^j, x_1 \oplus k_1^j, x_2 \oplus k_2^j, x_3 \oplus k_3^j)$ where $\oplus$ denotes the exclusive-or operation and j denotes the round number. The key injection operation is done 7 times, once at the beginning of the each round and once more after the last round.

The modular multiplication operation, γ, is the only non-linear operation in the cipher. In each encryption round, each of the 32-bit words is multiplied by a fixed constant such that the result y_i is

$$y_i = \begin{cases} x_i & \text{if } x_i = 2^{32} - 1 \\ x_i \otimes G_i & \text{if } x_i \neq 2^{32} - 1 \end{cases}$$

Where the operator $\otimes$ is the modular multiplication operator (i.e., $a \otimes b = (a * b)\ mod\ (2^{32} - 1)$) and $G_0 = 025F1CDB_x, G_1 = 2 \otimes G_0 = 04BE39B6_x, G_2 = 8 \otimes G_0 = 12F8E6D8_x$, and $G_3 = 128 \otimes G_0 = 2F8E6D81_x$. The result of the γ operation is therefore $(y_0, y_1, y_2, y_3) = \gamma(x_0, x_1, x_2, x_3)$.

Inverting γ is done by multiplying the ciphertext with G_i^{-1} such that

$$x_i = \begin{cases} y_i & \text{if } y_i = 2^{32} - 1 \\ y_i \otimes G_i^{-1} & \text{if } y_i \neq 2^{32} - 1 \end{cases}$$

where $G_0^{-1} = 0DAD4694_x, G_1^{-1} = 06D6A34A_x, G_2^{-1} = 81B5A8D2_x$ and $G_3^{-1} = 281B5A8D_x$.

For every word entering γ, the trivial differential transition $0 \rightarrow 0$ holds with probability 1. Another interesting property that was mentioned in [4] is that the differential transition $FFFFFFFF_x \rightarrow FFFFFFFF_x$ through γ also holds with probability 1. The use of these transitions is described in Section 5.

The η operation is a data-dependent operation on the leftmost and rightmost words of the state. If the LSB of the word is 1 it XORs a predefined constant δ into the word, otherwise it does nothing. Namely, $\eta(x_0, x_1, x_2, x_3) = (x_0 \oplus (lsb(x_0) \cdot \delta), x_1, x_2, x_3 \oplus (lsb(x_3) \cdot \delta))$ where $\delta = 2AAAAAAA_x$.

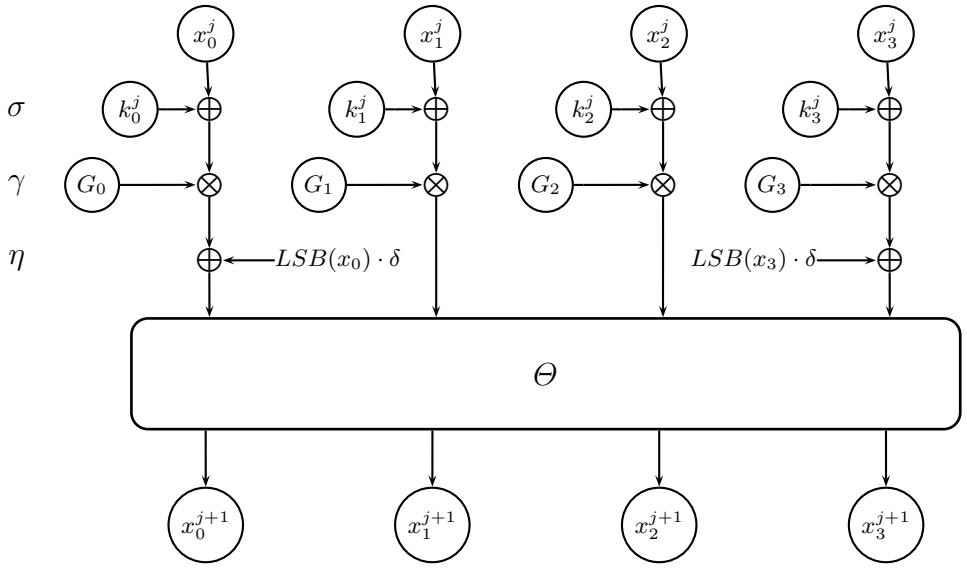
The diffusion between words comes from the θ operation that mixes the round's words such that every change in any word affects three words in the output. Namely, $\theta(x_0, x_1, x_2, x_3) = (x_0 \oplus x_1 \oplus x_3, x_0 \oplus x_1 \oplus x_2, x_1 \oplus x_2 \oplus x_3, x_0 \oplus x_2 \oplus x_3)$.

The j^{th} round of MMB over the block $X = (x_0, x_1, x_2, x_3)$ is: $\rho[k^j](X) = \theta(\eta(\gamma(\sigma[k^j](X))))$. A full description of MMB with plaintext P is:

$$\sigma[k^6](\rho[k^5](\rho[k^4](\rho[k^3](\rho[k^2](\rho[k^1](\rho[k^0](P))))))$$

A schematic view of MMB's round function can be found on Figure 1.

Fig. 1. MMB's Round Function in Round j.



2.1 Key Schedule

The original version of MMB used a simple key schedule algorithm that shifts the key words 1 position to the left (e.g. the key for round 0 is (k_0, k_1, k_2, k_3), the key for round 1 is (k_1, k_2, k_3, k_4) etc.). The key schedule is cyclic and repeats every 4 rounds [4]. To avoid exploitable symmetry properties a new version of MMB was published where in each round, in addition to the to the position change, each key word is XORed with a round-dependant constant. Therefore, the key word i for round j is $k_i^j = k_{i+j \ mod \ 4} \oplus (2^j \cdot B)$ with $B = DAE_x$ [3].

3 Previous Attacks on MMB

Wang et al. identified for MMB a 2-round differential characteristic with probability 1 [12]. This 2-round differential characteristic, described in Equation (1) was extended into a 5-round differential characteristic with probability of 2^{-110}. This 5-round differential characteristic can be used in an attack that recovers all key bits with data complexity of 2^{118} chosen plaintexts, time complexity of $2^{95.91}$, and memory requirements of 2^{65} 32-bit blocks. We note that the time complexity described in [12] does not take into account the fact that the time required to encrypt 2^{118} messages cannot be less than 2^{118}.

$$\begin{aligned}(0,\bar{0},\bar{0},0) \xrightarrow{\sigma[\mathrm{k}^0]} (0,\bar{0},\bar{0},0) \xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0}) \\ \xrightarrow{\sigma[\mathrm{k}^1]} (\bar{0},0,0,\bar{0}) \xrightarrow{\gamma} (\bar{0},0,0,\bar{0}) \xrightarrow{\eta} (\bar{\delta},0,0,\bar{\delta}) \xrightarrow{\theta} (0,\bar{\delta},\bar{\delta},0)\end{aligned} \tag{1}$$

Jia et al. [6] improved Wang's analysis to build a 5-round sandwich distinguisher with probability 1. This attack exploits the 2-round differential characteristic identified in [12] to construct

Rounds	Attack	Time*	Data	Memory†	Keys	Source
6	Differential Cryptanalysis	$2^{95.91}$	2^{118} CP**	2^{64}	1	[12]
6	Rectangle-like sandwich	2^{64}	$2^{66.5}$ CP	$2^{70.5}$	1	[6]
6	Related-key boomerang	$2^{35.2}$	2^{20} ACPC***	$2^{20.3}$	4	Section 6
7	Related-key boomerang	$2^{35.3}$	$2^{20.6}$ ACPC	$2^{20.3}$	6	Section 7
8	Related-key boomerang	$2^{35.2}$	$2^{20.6}$ ACPC	$2^{20.3}$	6	Section 7

* The time complexity reported in [6, 12] does not take into account the time needed for generating the data.

** Chosen plaintexts.

*** Adaptive chosen plaintexts and ciphertexts.

† Memory is measured in 32-bit memory words.

Table 1. Summary of the attacks on MMB

a 5-round sandwich that is then used to recover the full key of the full MMB with $2^{66.5}$ chosen plaintexts, 2^{64} time, and $2^{70.5}$ memory bytes.

Table 1 summarizes all previous results on MMB and compares them with ours.

4 Cryptanalytic Techniques for Block Ciphers Used in This Paper

4.1 Differential Cryptanalysis

One of the most notable techniques in cryptanalysis is differential cryptanalysis, developed by Biham and Shamir and published in 1990 [2], examines the evolution of differences between two inputs. An input difference is the difference between two inputs entering a cryptosystem, usually with respect to the exclusive-or operation. The output difference is the difference between the outputs of two such inputs. We say that an input difference Δ can cause an output difference Δ^* under the function f with probability p if a portion p of the possible pairs of messages having a difference Δ result in outputs having a difference Δ^* after applying f. If these conditions hold we write that $\Delta \xrightarrow{f} \Delta^*$ with probability p.

A differential characteristic that describes a single encryption round is called a 1-round differential characteristic. Biham and Shamir showed that two or more differential characteristics can be concatenated to form a longer differential characteristics if the output difference of one differential characteristic is the input difference of the other differential characteristic.

Once a good long differential characteristic is identified, the adversary tries to find a pair of messages that satisfies it. By examining many plaintext pairs, the adversary tries to distinguish the wrong pairs (i.e., those pairs which do not satisfy the differential characteristic) from the right pairs (i.e., those pairs which satisfy the differential characteristic). The amount of data needed to find a right pair is proportional to the inverse of the probability of the differential characteristic used and can be somewhat reduced by various techniques. Once a right pair is found, it can be used to recover the keys used in the cryptosystem by examining which keys cause the messages to satisfy the required differences.

4.2 Related-Key Differential Attack

Since its publication in 1990, differential cryptanalysis received great attention in the cryptographic community. Several researchers published extensions for the core technique. One of these extensions is the related-key differential attack which was published by Kelsey et al. in 1997 [7]. In a related-key differential attack the adversary is allowed, in addition to examining the evolution of differences between inputs, to introduce differences to the key. Namely, in the attack, two messages

are encrypted using two keys that have some difference chosen by the adversary. This difference is injected into the intermediate encryption values by the key injection operation and propagates. Modulo some small technical issues, the remainder of the attack is the same as in regular differential attacks.

4.3 The Boomerang Attack

Another extension to differential cryptanalysis is the boomerang attack suggested by Wagner in 1999 [11]. A boomerang attack uses two differential characteristics of relatively small number of rounds n and m with probabilities p and q, respectively, to construct a distinguisher for $m+n$ rounds.

A boomerang is composed of two differential characteristics $\triangle \rightarrow \triangle^*$ for n rounds and $\bigtriangledown^* \rightarrow \bigtriangledown$ for m rounds with probabilities p and q, respectively. The adversary chooses two plaintexts P_1 and P_2 such that $P_1 \oplus P_2 = \triangle$ and asks for their respective values C_1 and C_2 after $m+n$ encryption rounds. The adversary then XORs these ciphertexts with $\bigtriangledown$ to obtain the ciphertexts C_3 and C_4, respectively, and asks for their decrypted values P_3 and P_4. The boomerang suggests that $P_1 \oplus P_2 = P_3 \oplus P_4 = \triangle$ with probability $p^2 \cdot q^2$.

4.4 Related-Key Boomerang Attack

The related-key boomerang attack is an extension of the boomerang attack first suggested in 2004 by Kim et al. [1, 5, 8]. The idea of a related-key boomerang is to use two related-key differentials to construct the boomerang. After constructing this boomerang, the attack is then carried in the same way as with regular boomerangs (again, modulo a few small differences).

5 A Related-Key Boomerang attack for the Full MMB

Before we describe the related-key differential characteristics used to construct the boomerangs we observe that for any message, the trivial differential transition $0 \rightarrow 0$ holds with probability 1. Another interesting property which is described in [4] is that an input difference $FFFFFFFF_x$ between two input words entering $\otimes$ cause an output difference of $FFFFFFFF_x$ with probability 1 (independent of G_i). To simplify the description we denote a difference of 0 in a word as 0 and a difference of $FFFFFFFF_x$ as $\bar{0}$.

Another point worth mentioning is that if the difference between the leftmost or the rightmost words entering η is $\bar{0}$, the output difference must be $\delta \oplus FFFFFFFF_x$. The η operation XORs the constant $\delta = 2AAAAAAA_x$ to the leftmost and rightmost words if their least significant bit is 1. In the event that the difference between two input words is $FFFFFFFF_x$, one of them must have 1 as its least significant bit while the other must have 0, thus, δ is XORed only to one of them, causing the transition. To simplify notations, we denote $\delta \oplus FFFFFFFF_x$ by $\bar{\delta}$.

We present two related-key differentials: The 3-round related key differential $\triangle \rightarrow \triangle^*$ with input difference $(0, 0, \bar{0}, \bar{0})$ and key difference $(0, 0, \bar{0}, \bar{0})$. This differential is an extension of (1) where we use the key difference to control the propagation of the difference. The differential

$$\begin{aligned}
\triangle = (0,0,\bar{0},\bar{0}) &\xrightarrow[(0,0,\bar{0},\bar{0})]{\sigma[\mathrm{k}^1]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0) \\
&\xrightarrow[(0,\bar{0},\bar{0},0)]{\sigma[\mathrm{k}^2]} (0,\bar{0},\bar{0},0) \xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0}) \\
&\xrightarrow[(\bar{0},\bar{0},0,0)]{\sigma[\mathrm{k}^3]} (0,\bar{0},0,\bar{0}) \xrightarrow{\gamma} (0,\bar{0},0,\bar{0}) \xrightarrow{\eta} (0,\bar{0},0,\bar{\delta}) \xrightarrow{\theta} (\delta,\bar{0},\delta,\bar{\delta}) = \triangle^*
\end{aligned}$$

holds with probability 1. We can extend this differential by prepending an additional round

$$(\bar{X},\bar{0},0,\bar{0}) \xrightarrow[(\bar{0},0,0,\bar{0})]{\sigma[\mathrm{k}^0]} (X,\bar{0},0,0) \xrightarrow{\gamma} (\bar{\delta},\bar{0},0,0) \xrightarrow{\eta} (\bar{0},\bar{0},0,0) \xrightarrow{\theta} (0,0,\bar{0},\bar{0}) = \triangle, \tag{2}$$

where $\bar{X}$ is some undetermined difference satisfying $\bar{X} \xrightarrow{\oplus \mathrm{k}_0^0} X$ and $X \xrightarrow{\otimes \mathrm{G}_0} \bar{\delta}$.

The second differential we use is a 2-round related-key differential $\bigtriangledown^* \rightarrow \bigtriangledown$ with input difference $(0,0,\bar{0},0)$ and key difference $(0,0,\bar{0},0)$

$$\begin{aligned}\bigtriangledown^* = (0,0,\bar{0},0) &\xrightarrow[(0,0,\bar{0},0)]{\sigma[\mathrm{k}^1]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0) \\ &\xrightarrow[(0,\bar{0},0,0)]{\sigma[\mathrm{k}^2]} (0,\bar{0},0,0) \xrightarrow{\gamma} (0,\bar{0},0,0) \xrightarrow{\eta} (0,\bar{0},0,0) \xrightarrow{\theta} (\bar{0},\bar{0},\bar{0},0) = \bigtriangledown\end{aligned}$$

that also holds with probability 1. This differential can also be extended by prepending an additional round:

$$(0,\bar{0},\bar{0},\bar{Y}) \xrightarrow[(0,0,0,\bar{0})]{\sigma[\mathrm{k}^0]} (0,\bar{0},\bar{0},Y) \xrightarrow{\gamma} (0,\bar{0},\bar{0},\bar{\delta}) \xrightarrow{\eta} (0,\bar{0},\bar{0},\bar{0}) \xrightarrow{\theta} (0,0,\bar{0},0) = \bigtriangledown^*, \tag{3}$$

where like in the case of $\bar{X}$, $\bar{Y}$ is an undetermined difference satisfying $\bar{Y} \xrightarrow{\oplus \mathrm{k}_3^0} Y$ and $Y \xrightarrow{\otimes \mathrm{G}_3} \bar{\delta}$. We list the most probable values of Y's and X's (with their probability) in Appendix A. In Section 6 we show how to use the birthday paradox to find pairs which satisfy these differences regardless of the exact probabilities.

We construct two boomerangs. The first 5-round related-key boomerang is the concatenation of $\bigtriangledown^* \rightarrow \bigtriangledown$ after $\triangle \rightarrow \triangle^*$ without the additional round presented in Equation (2). This boomerang has probability 1 and can only be used as a distinguisher. Extending the $\triangle \rightarrow \triangle^*$ differential characteristic by one round (as specified in Equation (2)) forms a 6-round related-key boomerang we denote by B_0. This boomerang is depicted in Figure 2.

We can use the same differentials in reverse order to construct a second boomerang which we denote as B_1. We start by concatenating the differential characteristic $\triangle \rightarrow \triangle^*$ after $\bigtriangledown^* \rightarrow \bigtriangledown$ to form a 5-round boomerang with probability 1. We then extend $\bigtriangledown \rightarrow \bigtriangledown^*$ by one round (as specified in Equation (3)) to form a 6-round boomerang that can be used in a 1R attack. The second boomerang is depicted in Figure 3.

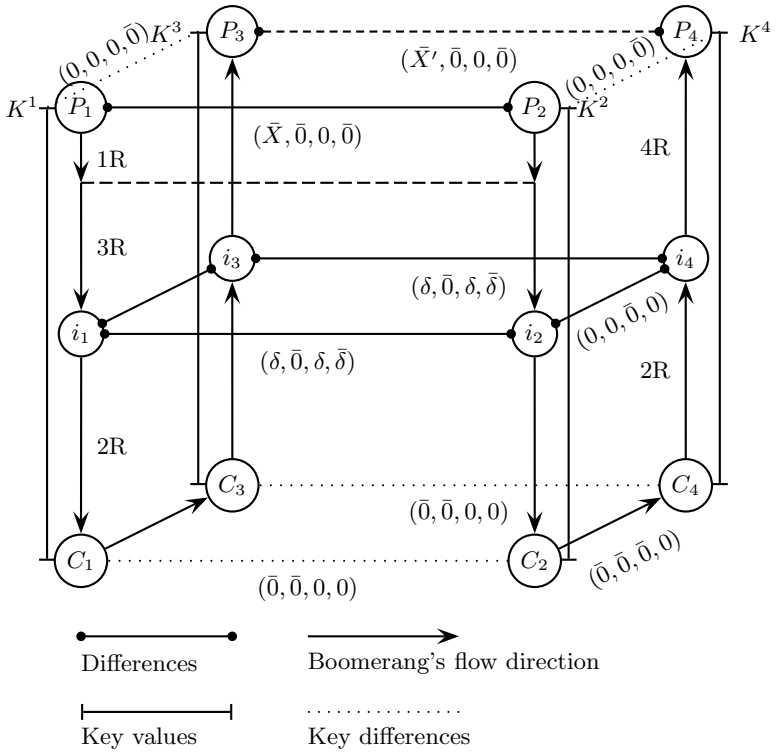
6 Description of the Key Recovery Attack

In this section we describe our related-key boomerang attack on MMB and the key recovery phase that is used to recover 64 bits out of the 128-bit key. We then show how to efficiently recover the remaining 64 key bits given the knowledge of the previous 64, for the full MMB. We conclude the section with a description of our experimental verification of this attack.

6.1 Related-Key Boomerang Attack

We recall that the 128-bit key is composed of four 32-bit key words (k_0, k_1, k_2, k_3). We recover each of these words separately. The first 32 key bits (those of k_0) are recovered using the boomerang B_0 and the last 32 key bits (those of k_3) are recovered using the boomerang B_1.

Fig. 2. The Description of B_0.



In order to use B_0 we need 4 related-keys. Two of them, namely

$$K^1 = (k_0, k_1, k_2, k_3); K^2 = K^1 \oplus (\bar{0}, 0, 0, \bar{0})$$

are used for encryption and the other two, namely

$$K^3 = K^1 \oplus (0, 0, 0, \bar{0}); K^4 = K^2 \oplus (0, 0, 0, \bar{0}) = K^1 \oplus (\bar{0}, 0, 0, 0),$$

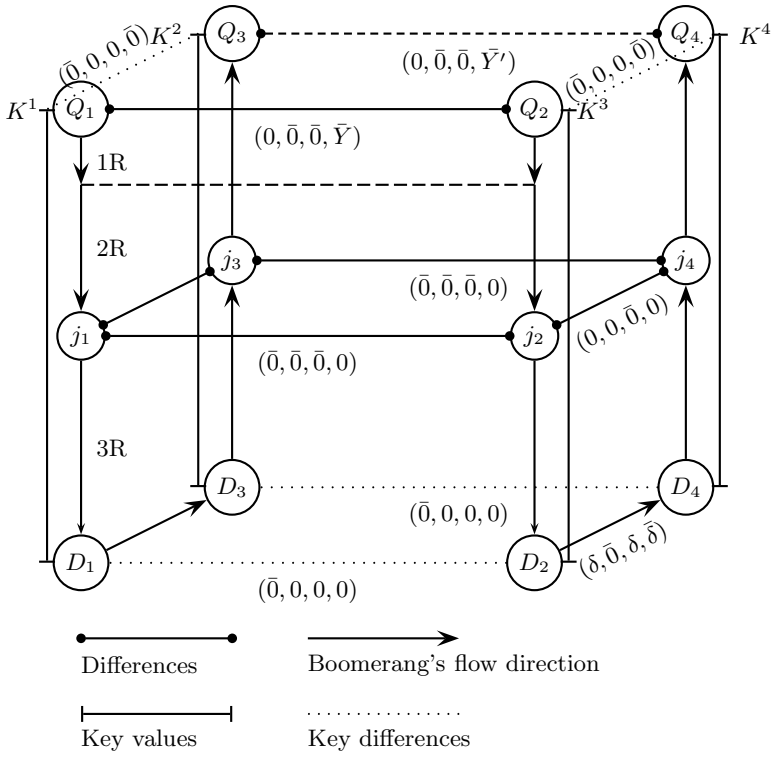
are used for decryption.

We pick a set of 2^{17} random plaintexts $P^1 = \{P^1_0, ..., P^1_{2^{17}-1}\}$ all having the same value in bits 32–127 and different values in bits 0–31. Then, we generate another set of 2^{17} plaintexts $P^2 = \{P^2_0, ..., P^2_{2^{17}-1}\}$ where $P^2_i = P^1_i \oplus (0, \bar{0}, 0, \bar{0})$. We then ask for the encryption of all the values in P^1 under K^1 to obtain the set of respective ciphertexts, $C^1 = \{C^1_0, ..., C^1_{2^{17}-1}\}$, and ask for the encryption of all values in P^2 under K^2 to obtain the respective set of ciphertexts $C^2 = \{C^2_0, ..., C^2_{2^{17}-1}\}$.

We XOR all values of C^1 and C^2 with $(\bar{0}, \bar{0}, \bar{0}, 0)$ to obtain $C^3 = C^1 \oplus (\bar{0}, \bar{0}, \bar{0}, 0) = \{C^3_0, ..., C^3_{2^{17}-1}\}$ and $C^4 = C^2 \oplus (\bar{0}, \bar{0}, \bar{0}, 0) = \{C^4_0, ..., C^4_{2^{17}-1}\}$. We ask for the decryption of the messages in C^3 under K^3 to obtain a set of plaintexts $P^3 = \{P^3_0, ..., P^3_{2^{17}-1}\}$, and the decryption of the messages in C^4 under K^4 to obtain a set of plaintexts $P^4 = \{P^4_0, ..., P^4_{2^{17}-1}\}$.

We expect, due to the birthday paradox, that two plaintexts P^1_i and P^2_j, taken from P^1 and P^2, respectively, will collide (i.e., have a zero difference) in bits 0–31 after a single round of $\sigma[k^0], \gamma$, and

Fig. 3. The Description of B_1



η with a non-negligible probability.[1] Such two colliding messages form a pair with input difference $\triangle$ as required by the differential characteristic $\triangle \rightarrow \triangle^*$ (the difference in the remaining words is set with probability 1). Since both differentials used in the boomerang hold with probability 1, the encryption, XOR by $(\bar{0},\bar{0},\bar{0},0)$ and 5-round decryption of it will inevitably result with a difference of Δ causing its respective P_i^3 and P_j^4 to also have a difference of the form $(\bar{X}',\bar{0},0,\bar{0})$ after the decryption.

Analyzing the expected number of right pairs is straightforward using the birthday paradox framework. The values of P_1 occupy 2^{17} bins out of the 2^{32} possible bins. Therefore, each of the 2^{17} possible values of P_2 has a chance of $\frac{2^{17}}{2^{32}}$ to collide with a value from P_1. Hence, the expected number of right pairs (which lead to right quartets with probability 1) is $2^{17} \cdot \frac{2^{17}}{2^{32}} = 4$. In Section 6.4 we test this prediction empirically.

We store all values of P_3 in a hash table using bits 32–127 as the hash key. Then, once obtaining the values of P_4 we search for "collisions" in those bits (taking into account the expected difference in these bits) to identify a candidate pair (and thus, a candidate quartet). The probability that among all the possible 2^{34} pairs, two messages form a wrong pair (i.e., agreeing on bits 32–127 without following the boomerang) is $2^{34} \cdot 2^{-96} = 2^{-62}$. Thus, we can safely assume that all candidate

[1] As we discuss later, we actually expect four such pairs. Given that the actual number of such pairs follows a Poisson distribution with a mean value of 4, we expect at least one such pair to exist with probability of 98.2%.

quartets are right quartets. Note that we do not store the plaintexts without their respective ciphertexts, hence, reducing the memory complexity.

Once we identify the four plaintexts forming a right quartet, $((P_i^1,P_j^2),(P_i^3,P_j^4))$, we try all 2^{32} possible values for $k_0^1 = k_0^3$ and $k_0^2 = k_0^4 = k_0^1 \oplus \bar{0}$ (the first 32 bits of K^1, K^2, K^3, and K^4) to see which of them causes both pairs to have a zero difference in the first word after one round. These 2^{32} trials suggest two possible values as the key word, either k_0^1 or $\bar{k_0^1}$. Note that usually in related-key attacks we expect **one** solution on average for these cases. However, in the specific case of $\otimes$, complementing the entire input necessarily complements the entire output. Hence, if the two inputs to $\otimes$ are x and x', and a 32-bit key word k satisfies $((x \oplus k) \otimes G_0) \oplus (x' \oplus \bar{k}) \otimes G_0 = \bar{0}$, then $\bar{k}$ also satisfies this relation, as both results are complemented when the value of k_0 is complemented. At the last part of the attack, we encrypt a message using all key combinations to determine which value is the right key and which value is its complementary.

To recover bits 96–127 of K^1 and K^2 we use the same method. We pick 2^{17} random plaintexts $Q^1 = \{Q_0^1, ..., Q_{2^{17}-1}^1\}$ all having the same value in bits 0–95 and different values in bits 96–127. Then, we generate another 2^{17} plaintexts $Q^2 = \{Q_0^2, ..., Q_{2^{17}-1}^2\}$, where $Q_i^2 = Q_i^1 \oplus (0, \bar{0}, \bar{0}, 0)$ and use the same algorithm to encrypt the messages under K^1 and K^3, XOR the ciphertexts with $\triangle^*$ and decrypt them under K^2 and K^4. The key word k_3 is then recovered by 2^{32} trails in a similar way to the one described for recovering k_0.

6.2 Recovering the Remaining Key Bits

Once we obtained key bits 0–31 and 96–127, we use an extension of the related-key differential $\bigtriangledown^* \rightarrow \bigtriangledown$ to recover key bits 32–63 with a simple 1R attack. The 4-round related-key differential characteristic

$$\begin{aligned}
\bigtriangledown^* = (0,0,\bar{0},0) &\xrightarrow[(0,0,\bar{0},0)]{\sigma[\mathrm{k}^1]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0) \\
&\xrightarrow[(0,\bar{0},0,0)]{\sigma[\mathrm{k}^2]} (0,\bar{0},0,0) \xrightarrow{\gamma} (0,\bar{0},0,0) \xrightarrow{\eta} (0,\bar{0},0,0) \xrightarrow{\theta} (\bar{0},\bar{0},\bar{0},0) \\
&\xrightarrow[(\bar{0},0,0,0)]{\sigma[\mathrm{k}^3]} (0,\bar{0},\bar{0},0) \xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0}) \\
&\xrightarrow[(0,0,0,\bar{0})]{\sigma[\mathrm{k}^4]} (\bar{0},0,0,0) \xrightarrow{\gamma} (\bar{0},0,0,0) \xrightarrow{\eta} (\bar{\delta},0,0,0) \xrightarrow{\theta} (\bar{\delta},\bar{\delta},0,\bar{\delta}) = \bigtriangledown_{ee}
\end{aligned}$$

holds with probability 1. With the extension in (3) it forms a 5-round related-key differential characteristic for MMB. i.e., once we know a quartet of values which satisfy (3) (for example, as part of a right quartet in B_1) we can use it as two right pairs with respect to this 5-round related-key differential characteristic.

Let $Q_i^1 \in Q^1$ and $Q_j^2 \in Q^2$ be two plaintexts forming a right pair, and let $D_i^1 = (w_6 \oplus k_0^6, x_6 \oplus k_1^6, y_6 \oplus k_2^6, z_6 \oplus k_3^6)$ and $D_j^2 = (w_6^* \oplus k_0^6, x_6^* \oplus k_1^6, y_6^* \oplus \bar{k_2^6}, z_6^* \oplus k_3^6)$ be their respective ciphertexts. We observe that each of the words $w, x, y, z, w^*, x^*, y^*$, and z^* is the result of the θ operation which XORs three intermediate values. We denote these intermediate values as $a, b, c, d, a^*, b^*, c^*$, and d^*, i.e.,

$$\begin{aligned}
w_6 = a_6 \oplus b_6 \oplus d_6; w_6^* = a_6^* \oplus b_6^* \oplus d_6^* \\
x_6 = a_6 \oplus b_6 \oplus c_6; x_6^* = a_6^* \oplus b_6^* \oplus c_6^* \\
y_6 = b_6 \oplus c_6 \oplus d_6; y_6^* = b_6^* \oplus c_6^* \oplus d_6^* \\
z_6 = a_6 \oplus c_6 \oplus d_6; z_6^* = a_6^* \oplus c_6^* \oplus d_6^*.
\end{aligned}$$

To recover k_2 we simply XOR the first three words of each ciphertext

$$w_6 \oplus k_0^6 \oplus x_6 \oplus k_1^6 \oplus y_6 \oplus k_2^6 = a_6 \oplus b_6 \oplus d_6 \oplus k_0^6 \oplus a_6 \oplus b_6 \oplus c_6 \oplus k_1^6 \oplus b_6 \oplus c_6 \oplus d_6 \oplus k_2^6 = b_6 \oplus k_0^6 \oplus k_1^6 \oplus k_2^6$$

and

$$w_6^* \oplus k_0^6 \oplus x_6^* \oplus k_1^6 \oplus y_6^* \oplus \bar{k_2^6} = a_6^* \oplus b_6^* \oplus d_6^* \oplus k_0^6 \oplus a_6^* \oplus b_6^* \oplus c_6^* \oplus k_1^6 \oplus b_6^* \oplus c_6^* \oplus d_6^* \oplus \bar{k_2^6} = b_6^* \oplus k_0^6 \oplus k_1^6 \oplus \bar{k_2^6}$$

where the values of k_0^6 and k_1^6 are the 64 key bits previously recovered. The adversary then searches for the values of k_2^6 and $\bar{k_2^6}$ that satisfy the equation $((b^* \oplus \bar{k_2^6} \oplus k_0^6 \oplus k_1^6) \otimes G_2^{-1}) \oplus ((b \oplus k_2^6 \oplus k_0^6 \oplus k_1^6) \otimes G_2^{-1}) = \bar{\delta}$. Taking the second pair of a right boomerang quartet allows discarding a few more of the remaining wrong options.

After recovering k_0, k_2, and k_3, the remaining k_1 (32 bits) is recovered by exhaustive search.

Analysis of the Full Attack: The first part of the attack allows recovering 64 key bits in $2^{32.4}$ time, using four related keys, $2^{20.3}$ memory,[2] and 2^{20} adaptive chosen plaintexts and ciphertexts. The second part of the attack requires running 2^{32} round operations (which are about $\frac{1}{6} \cdot 2^{32} = 2^{29.4}$ full MMB encryptions) with no additional memory and data requirements. The third part of the attack requires running $8 \cdot 2^{32} = 2^{35}$ full MMB encryptions, again, with no additional memory and data requirements. Thus, the overall complexity of this attack is $2^{35.2}$ time, $2^{20.3}$ memory, and 2^{20} adaptive chosen plaintexts and ciphertexts encrypted under 4 related-keys.

6.3 Experimental Verification

The low time, data, and memory complexities of the attack allow verifying it experimentally. The implementation of the attack uses two programming languages: C and Python. The C program was used to implement the cryptographic parts of the attack (i.e. the boomerangs and the key search). Python was used to invoke different modules of the attack and collect data for statistical analysis.

The C program was compiled and ran on a Linux Mint machine using GCC 4.6.3 with the -O3 optimization flag. The program starts by generating a random 128-bit message and a random 128-bit key. It then forks into two processes, one implementing B_0 and the other implementing B_1. The first process generates a second message and a second key with the appropriate differences and replaces the first word of both messages with a random one. It then saves the two messages and encrypts them under the related-keys to obtain their respective ciphertexts. The ciphertexts and the keys are then XORed with the appropriate values and decrypted to obtain new plaintexts. For each such new plaintext, the program stores it for later use. Once all plaintexts are decrypted, the program searches for right quartets. This is done by searching for pairs in which bits 32–127 of the decrypted plaintexts have difference of $(\bar{0}, 0, \bar{0})$.[3]

Once a right quartet is found, the key recovery is done by trying all possible values as the key for the first message word in both pairs and checking which value leads to a zero difference after a single round of $\sigma[k^0], \gamma$, and η. All such values are written into the output file as possible keys. This process is repeated for all quartets satisfying the conditions (i.e., the candidate quartets). The second process does the same with the minor change that it searches for decrypted pairs in which bits 0–95 has difference of $(0, \bar{0}, \bar{0})$ and searches for the forth key word instead of the first.

The Python program was written in Python 2.7.3 over GCC 4.6.3. Once the C program finishes its execution the Python program reads the two output files and invokes another C program that

[2] We alert the reader that in each boomerang we need to store 2^{18} 128-bit plaintexts from P^3 and P^4, and 2^{18} 32-bit representations of the plaintexts from P^1 and P^2. The ciphertexts themselves are not used in the key recovery part, and thus are not stored.

[3] Although an implementation using a hash-table is faster in theory, we found out that in practice, the required bookkeeping induces higher overhead than a simple list of values.

uses the results of the previous phase to recover key bits 32–63 by iterating over all possible key values which satisfy the conditions in Subsection 6.2. The python program then runs another C program that exhaustively searches for the last key word. The program tries in parallel all 8 possible key words combinations with all 2^{32} possible values for the remaining key word. Once the full key is identified in one of the subprograms, the program outputs it and terminates.

6.4 Results of the Experimental Verification

Our experiment included running the program 100 times. Out of these 100 trials, recovering k_0 was successful 97 times (97%), Recovering k_3 was successful 98 times (98%). In 95 of the trials (95%), both k_0 and k_3 were recovered successfully. The key word k_2 was recovered successfully 95 times (95%), i.e., whenever k_0 and k_3 were both recovered, so was k_2. We consider the experiment to be successful in recovering a key word when the Python program returns exactly 2 possible values for that word.

We also tested the actual amount of quartets. Out of the 100 trials, the program found on average 4.06 candidate quartets for B_0 and 4.01 candidate quartets for B_1. This result is perfectly aligned with the calculation we presented in Section 6.1.

The average running time of the program on an i5 personal computer with 4 GB RAM, running Linux Mint was 350.32 seconds for the first phase and 117.17 seconds for the second phase with standard deviations of 109.86 seconds and 52.25 seconds, respectively. Executing 2^{32} encryptions of the full MMB requires 341.57 seconds. When parallelized over an i5 CPU with 4 cores and terminated on key detection, the average running time of this stage is 401.66 seconds with a standard deviation of 289.9 seconds. Hence, the average total time required for the recovery of the full key is 14.5 minutes with a standard deviation of 5.5 minutes.

7 Attacking More Rounds of MMB

In this section we extend our attack to show that even if MMB was extended to 7 or 8 rounds our attack would still work. We first show that the related-key differential characteristic $\bigtriangledown^* \rightarrow \bigtriangledown$ can be extended by 2 rounds and thus, B_0 and B_1 can be extended to cover 7 and 8 rounds of MMB. We use the extended boomerangs to recover k_0 and k_3 as before.

We then present a new 4-round related-key differential characteristic. We use 3 out of the 4 rounds of this differential to construct another boomerang, B_2, which allows us to recover k_1 for the 7-round variant, and use the complete differential to construct another boomerang, B_3, that is used to recover k_1 in the 8-round variant.

7.1 Recovering k_0 and k_3 for 7-Round and 8-Round Variants of MMB

To attack the 7-round variant of MMB we extend the related-key differential characteristic $\bigtriangledown^* \rightarrow \bigtriangledown$ by one round into a 3-round related-key differential characteristic $\bigtriangledown^* \rightarrow \bigtriangledown_e$ with probability 1:

$$\begin{aligned}\bigtriangledown^* = (0,0,\bar{0},0) &\xrightarrow[(0,0,\bar{0},0)]{\sigma[\mathrm{k}^1]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0)\\ &\xrightarrow[(0,\bar{0},0,0)]{\sigma[\mathrm{k}^2]} (0,\bar{0},0,0) \xrightarrow{\gamma} (0,\bar{0},0,0) \xrightarrow{\eta} (0,\bar{0},0,0) \xrightarrow{\theta} (\bar{0},\bar{0},\bar{0},0)\\ &\xrightarrow[(\bar{0},0,0,0)]{\sigma[\mathrm{k}^3]} (0,\bar{0},\bar{0},0) \xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0}) = \bigtriangledown_e\end{aligned}$$

In a similar way, to attack the 8-round variant of MMB we can extend $\bigtriangledown^* \rightarrow \bigtriangledown_e$ into a 4-round related-key differential characteristic:

$$\begin{aligned}\bigtriangledown^* = (0,0,\bar{0},0) &\xrightarrow[(0,0,\bar{0},0)]{\sigma[\mathrm{k}^1]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0)\\ &\xrightarrow[(0,\bar{0},0,0)]{\sigma[\mathrm{k}^2]} (0,\bar{0},0,0) \xrightarrow{\gamma} (0,\bar{0},0,0) \xrightarrow{\eta} (0,\bar{0},0,0) \xrightarrow{\theta} (\bar{0},\bar{0},\bar{0},0)\\ &\xrightarrow[(\bar{0},0,0,0)]{\sigma[\mathrm{k}^3]} (0,\bar{0},\bar{0},0) \xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0})\\ \xrightarrow[(0,0,0,\bar{0})]{\sigma[\mathrm{k}^4]} (\bar{0},0,0,0) &\xrightarrow{\gamma} (\bar{0},0,0,0) \xrightarrow{\eta} (\bar{\delta},0,0,0) \xrightarrow{\theta} (\bar{\delta},\bar{\delta},0,\bar{\delta}) = \bigtriangledown_{ee}\end{aligned}$$

With these differentials we can construct two boomerangs, B_0^e (resp., B_0^{ee}) and B_1^e (resp., B_0^{ee}) which are simply B_0 and B_1 where $\bigtriangledown^* \rightarrow \bigtriangledown_e$ (resp., $\bigtriangledown^* \rightarrow \bigtriangledown_e$) is used instead of $\bigtriangledown^* \rightarrow \bigtriangledown$. We use the same method as in Section 6 to generate two sets of plaintexts of size 2^{17} each, that differ only in bits 0–31, and another two sets of plaintexts of size 2^{17} each, that differ only in bits 96–127. Then, we encrypt the plaintexts under the appropriate related-keys, XOR them with the required differences and decrypt under the appropriate keys find right quartets with respect to B_0^e (resp., B_0^{ee}) and B_1^e (resp., B_0^{ee}). As in Section 6 we expect two plaintexts, one of each group to collide with non-negligible probability, thus, satisfying the required input differences for $\bigtriangleup \rightarrow \bigtriangleup^*$ and $\bigtriangledown^* \rightarrow \bigtriangledown_e$ (resp., $\bigtriangledown^* \rightarrow \bigtriangledown_{ee}$). The 32 bits of k_0 are then recovered by 2^{32} trials and the bits of k_3 are recovered by another 2^{32} trials.

this part of the attack uses an overall time of $2^{32.2}$ for the 7-round variant and time of 2^{32} for the 8-round variant. Both variants use $2^{20.3}$ memory, and 2^{20} adaptive chosen plaintexts and ciphertexts encrypted under four related-keys.

7.2 Recovering the Full Key of a 7-round MMB

To recover the value of key word k_1 we use the already recovered key word k_0. We pick two values w and w^*satisfying $w \oplus w^* = \bar{\delta}$ and partially decrypt them through γ and $\sigma[k^1]$ to obtain $(w \otimes G_0^{-1}) \oplus k_0$ and $(w^* \otimes G_0^{-1}) \oplus k_0$. For the sake of simplicity we denote the difference between the decrypted values as $\bar{W}$. The 3-round related-key differential characteristic $\tau \rightarrow \tau^*$ is:

$$\begin{aligned}\tau = (\bar{W},Z,0,\bar{0}) &\xrightarrow[(\bar{0},0,0,\bar{0})]{\sigma[\mathrm{k}^1]} (W,Z,0,0) \xrightarrow{\gamma} (\bar{\delta},\bar{0},0,0) \xrightarrow{\eta} (\bar{0},\bar{0},0,0) \xrightarrow{\theta} (0,0,\bar{0},\bar{0})\\ &\xrightarrow[(0,0,\bar{0},\bar{0})]{\sigma[\mathrm{k}^2]} (0,0,0,0) \xrightarrow{\gamma} (0,0,0,0) \xrightarrow{\eta} (0,0,0,0) \xrightarrow{\theta} (0,0,0,0)\\ \xrightarrow[(0,\bar{0},\bar{0},0)]{\sigma[\mathrm{k}^3]} (0,\bar{0},\bar{0},0) &\xrightarrow{\gamma} (0,\bar{0},\bar{0},0) \xrightarrow{\eta} (0,\bar{0},\bar{0},0) \xrightarrow{\theta} (\bar{0},0,0,\bar{0}) = \tau^*\end{aligned}$$

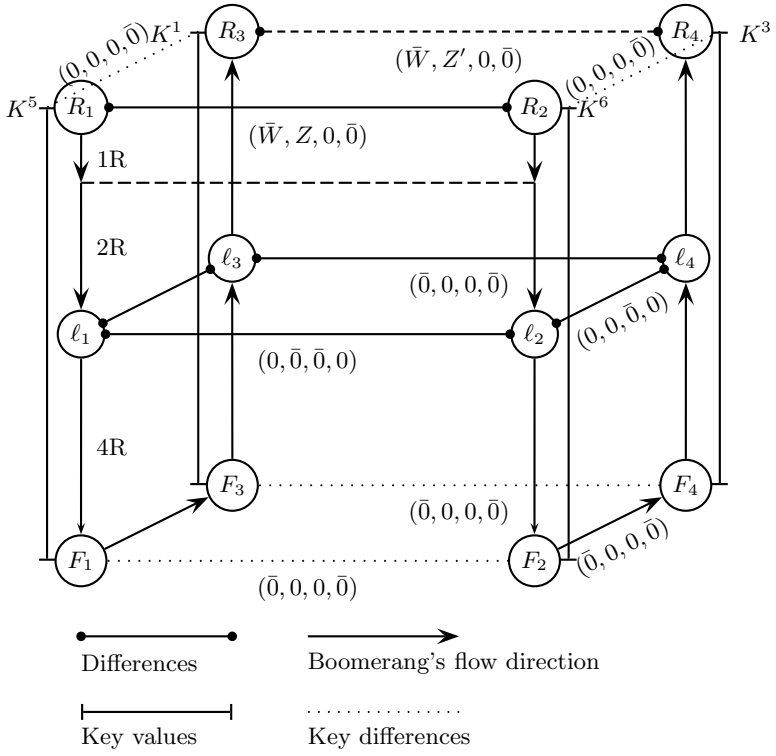
where Z is some undetermined difference satisfying $Z \xrightarrow{\otimes \mathrm{G}_0} \bar{\delta}$.

We can now construct another related-key boomerang, B_2, by concatenating $\bigtriangledown^* \rightarrow \bigtriangledown$ after $\tau \rightarrow \tau^*$. This related-key boomerang uses two of the previously used related-keys K^1 and K^3 and two new related-keys

$$K^5 = K^1 \oplus (0,0,0,\bar{0}); K^6 = K^1 \oplus (\bar{0},0,0,0)$$

and is depicted in Figure 4.

Fig. 4. The Description of B_2



We pick a set of 2^{17} random plaintexts $R^1 = \{R^1_0, ..., R^1_{2^{17}-1}\}$ all having the fixed value $(w \otimes G_0^{-1}) \oplus k_0$ in bits 0–31, a random value in bits 32–63 and the same value in bits 64–127. Then, we generate another set of 2^{17} plaintexts $R^2 = \{R^2_0, ..., R^2_{2^{17}-1}\}$ where $R^2_i = R^1_i \oplus (\bar{W}, 0, 0, \bar{0})$. We then ask for the encryption of all the values in R^1 under K^5 to obtain the set of respective ciphertexts, $F^1 = \{F^1_0, ..., F^1_{2^{17}-1}\}$, and ask for the encryption of all values in R^2 under K^6 to obtain the respective set of ciphertexts $F^2 = \{F^2_0, ..., F^2_{2^{17}-1}\}$.

We XOR all values of F^1 and F^2 with $(\bar{\delta}, \bar{\delta}, 0, \bar{\delta})$ to obtain $F^3 = F^1 \oplus (\bar{\delta}, \bar{\delta}, 0, \bar{\delta}) = \{F^3_0, ..., F^3_{2^{17}-1}\}$ and $F^4 = F^2 \oplus (\bar{\delta}, \bar{\delta}, 0, \bar{\delta}) = \{F^4_0, ..., F^4_{2^{17}-1}\}$. We ask for the decryption of the messages in F^3 under K^1 to obtain a set of plaintexts $R^3 = \{R^3_0, ..., R^3_{2^{17}-1}\}$, and the decryption of the messages in F^4 under K^3 to obtain a set of plaintexts, $R^4 = \{R^4_0, ..., R^4_{2^{17}-1}\}$.

We store all values of R^3 in a hash table indexed by bits 0–31 and 64–127. We identify a candidate quartet when we find a value in R^4 that has a W difference in bits 0–31, a zero difference in bits 64–95, and a $\bar{0}$ difference in bits 96–127 with some value from R^3. Once we identify a candidate quartet, we use the same technique described in Section 6.1 to recover the key word k_1. The rest of the key is recovered by an exhaustive search.

The second part of the attack recovers k_1 in time of $2^{31.2}$ encryptions, using 2^{19} adaptive chosen plaintexts and ciphertexts encrypted under the same two related keys as the first part and two new related-keys, with no additional memory requirements. The third part of the attack requires

2^{35} 7-round MMB invocations with no additional data, memory or related keys requirements. Therefore, the overall complexity of the attack is $2^{35.3}$ time, $2^{20.3}$ memory, and $2^{20.6}$ adaptive chosen plaintexts and ciphertexts encrypted under 6 related-keys.

7.3 Recovering the Full Key of an 8-round MMB

To recover k_1 in the 8-round variant of MMB we extend the related-key differential characteristic $\tau \rightarrow \tau^*$ to be $\tau \rightarrow \tau_e^*$:

$$\begin{aligned}
\tau = (\bar{W}, Z, 0, \bar{0}) &\xrightarrow[(\bar{0},0,0,\bar{0})]{\sigma[\mathrm{k}^1]} (W, Z, 0, 0) \xrightarrow{\gamma} (\bar{\delta}, \bar{0}, 0, 0) \xrightarrow{\eta} (\bar{0}, \bar{0}, 0, 0) \xrightarrow{\theta} (0, 0, \bar{0}, \bar{0}) \\
&\xrightarrow[(0,0,\bar{0},\bar{0})]{\sigma[\mathrm{k}^2]} (0, 0, 0, 0) \xrightarrow{\gamma} (0, 0, 0, 0) \xrightarrow{\eta} (0, 0, 0, 0) \xrightarrow{\theta} (0, 0, 0, 0) \\
&\xrightarrow[(0,\bar{0},\bar{0},0)]{\sigma[\mathrm{k}^3]} (0, \bar{0}, \bar{0}, 0) \xrightarrow{\gamma} (0, \bar{0}, \bar{0}, 0) \xrightarrow{\eta} (0, \bar{0}, \bar{0}, 0) \xrightarrow{\theta} (\bar{0}, 0, 0, \bar{0}) \\
&\xrightarrow[(\bar{0},\bar{0},0,0)]{\sigma[\mathrm{k}^4]} (0, \bar{0}, 0, \bar{0}) \xrightarrow{\gamma} (0, \bar{0}, 0, \bar{0}) \xrightarrow{\eta} (0, 0, 0, \bar{\delta}) \xrightarrow{\theta} (\delta, \bar{0}, \delta, \bar{\delta}) = \tau_e^*
\end{aligned}$$

where Z, w, w^*, W, and k_0 are as defined before.

We use this related-key differential characteristic to construct a forth boomerang B_3, which is simply B_2 where $\tau \rightarrow \tau_e^*$ is used instead of $\tau \rightarrow \tau^*$. Then, we use B_3 in exactly the same way as B_2 to recover k_1. B_3 is depicted in Figure 5.

The second part of the attack for the 8-round variant recovers k_1 in time of 2^{31} encryptions, using 2^{19} adaptive chosen plaintexts and ciphertexts encrypted under the same two related keys as the first part and two new related-keys, with no additional memory requirements. The third part of the attack requires 2^{35} 7-round MMB invocations with no additional data, memory or related keys requirements. Therefore, the overall complexity of the attack is $2^{35.2}$ time, $2^{20.3}$ memory, and $2^{20.6}$ adaptive chosen plaintexts and ciphertexts encrypted under 6 related-keys.

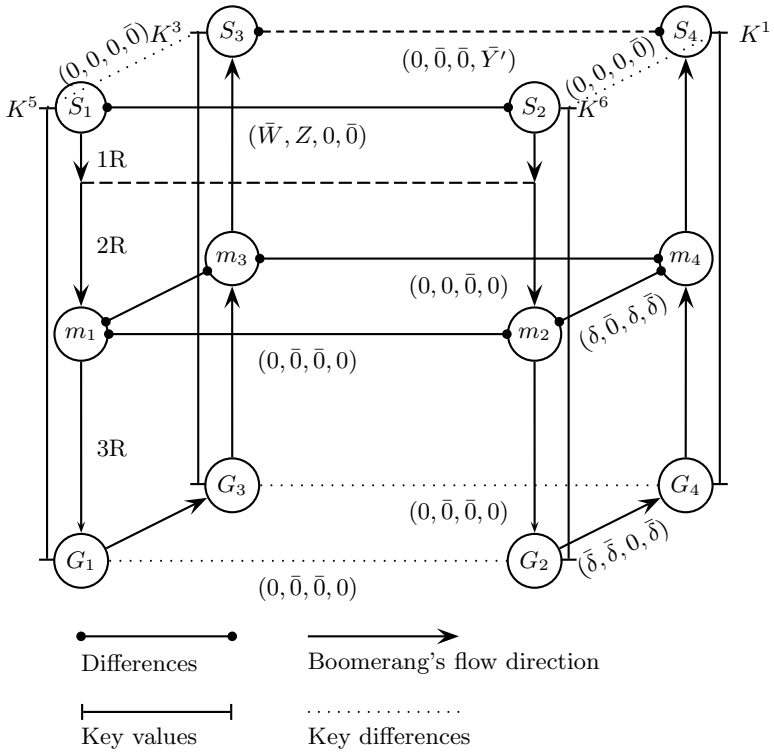
8 Conclusions

In this paper we have used various techniques from the differential cryptanalysis family to break the MMB block cipher. By extending previous results along with a new related-key differential we discovered, we were able to identify two related-key differentials that allowed us to construct two 5-round related-key distinguishers with probability 1. We then used each of these distinguishers as the basis for a 6-round boomerang that is able to recover 32 key bits using 2^{19} data in $2^{19.22}$ time using four related keys. We then used the already recovered key bits to recover another 32 key bits using a simple 1R attack. Finally, the last 32 bits are recovered by exhaustive search. The suggested attack can recover all the key bits in $2^{35.2}$ time using 2^{20} adaptive chosen plaintexts and ciphertexts and $2^{20.3}$ memory.

We verified our results experimentally by writing a program that recovers the required key bits in about 15 minutes on a home PC. To the best of our knowledge, though it has been many years since MMB was presented, this is the first practical time attack that recovers its full key.

Finally, we showed that even if MMB had been extended to include 7 or 8 rounds an adversary can still all its key bits using the same techniques, with similar time, data and memory complexities.

Fig. 5. The Description of B_3



References

1. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 507–525
2. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology **4**(1) (1991) 3–72
3. Daemen, J.: Cipher and Hash Function Design. Strategies based on linear and differential cryptanalysis. PhD thesis, Katholieke Universiteit Leuven (1995) René Govaerts and Joos Vandewalle (promotors).
4. Daemen, J., Govaerts, R., Vandewalle, J.: Block ciphers based on modular arithmetic. In Wolfowicz, W., ed.: Proceedings of the 3rd Symposium on State and Progress of Research in Cryptography, Rome,IT, Fondazione Ugo Bordoni (1993) 80–89
5. Hong, S., Kim, J., Lee, S., Preneel, B.: Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192. In Gilbert, H., Handschuh, H., eds.: FSE. Volume 3557 of Lecture Notes in Computer Science., Springer (2005) 368–383
6. Jia, K., Chen, J., Wang, M., Wang, X.: Practical Attack on the Full MMB Block Cipher. In Miri, A., Vaudenay, S., eds.: Selected Areas in Cryptography. Volume 7118 of Lecture Notes in Computer Science., Springer (2011) 185–199
7. Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Han, Y., Okamoto, T., Qing, S., eds.: ICICS. Volume 1334 of Lecture Notes in Computer Science., Springer (1997) 233–246
8. Kim, J., Kim, G., Hong, S., Lee, S., Hong, D.: The Related-Key Rectangle Attack - Application to SHACAL-1. In Wang, H., Pieprzyk, J., Varadharajan, V., eds.: ACISP. Volume 3108 of Lecture Notes in Computer Science., Springer (2004) 123–136

9. Lai, X., Massey, J.L.: A Proposal for a New Block Encryption Standard. In Damgård, I., ed.: EUROCRYPT. Volume 473 of Lecture Notes in Computer Science., Springer (1990) 389–404
10. Lai, X., Massey, J.L.: Markov Ciphers and Differentail Cryptanalysis. In Davies, D.W., ed.: EUROCRYPT. Volume 547 of Lecture Notes in Computer Science., Springer (1991) 17–38
11. Wagner, D.: The Boomerang Attack. In Knudsen, L.R., ed.: FSE. Volume 1636 of Lecture Notes in Computer Science., Springer (1999) 156–170
12. Wang, M., Nakahara, J., Sun, Y.: Cryptanalysis of the Full MMB Block Cipher. In Jr., M.J.J., Rijmen, V., Safavi-Naini, R., eds.: Selected Areas in Cryptography. Volume 5867 of Lecture Notes in Computer Science., Springer (2009) 231–248

A Probabilities for the Transitions $X \xrightarrow{\otimes \mathbf{G_0}} \bar{\delta}$ and $Y \xrightarrow{\otimes \mathbf{G_3}} \bar{\delta}$

In this Appendix we present a list of transitions from some input differences to δ with respect to modular multiplication by G_0 and G_3, and their probabilities:

Rank	Input Difference	Probability	$-log_2(p)$
1	$\texttt{7FBFFB64}_x$	$32768 \cdot 2^{-32}$	17
2	$\texttt{45440164}_x$	$31872 \cdot 2^{-32}$	17.03
3	$\texttt{7F3FFB64}_x$	$31744 \cdot 2^{-32}$	17.04
4	$\texttt{C5440164}_x$	$28032 \cdot 2^{-32}$	17.22
5	$\texttt{4000C164}_x$	$26912 \cdot 2^{-32}$	17.28
6	$\texttt{C000C164}_x$	$26336 \cdot 2^{-32}$	17.31
7	$\texttt{90440164}_x$	$26112 \cdot 2^{-32}$	17.32
8	$\texttt{88240164}_x$	$26112 \cdot 2^{-32}$	17.32
9	$\texttt{08240164}_x$	$26112 \cdot 2^{-32}$	17.32
10	$\texttt{80240164}_x$	$26112 \cdot 2^{-32}$	17.32
11	$\texttt{00240164}_x$	$26112 \cdot 2^{-32}$	17.32
12	$\texttt{10440164}_x$	$26112 \cdot 2^{-32}$	17.32
13	$\texttt{90C40164}_x$	$25344 \cdot 2^{-32}$	17.37
14	$\texttt{10C40164}_x$	$25344 \cdot 2^{-32}$	17.37
15	$\texttt{C0014404}_x$	$25024 \cdot 2^{-32}$	17.38
16	$\texttt{C0014164}_x$	$24992 \cdot 2^{-32}$	17.39
17	$\texttt{C00A0164}_x$	$24960 \cdot 2^{-32}$	17.39
18	$\texttt{80012404}_x$	$24576 \cdot 2^{-32}$	17.41
19	$\texttt{80011C04}_x$	$24576 \cdot 2^{-32}$	17.41
20	$\texttt{00012404}_x$	$24576 \cdot 2^{-32}$	17.41
21	$\texttt{00011C04}_x$	$24576 \cdot 2^{-32}$	17.41
22	$\texttt{400A0164}_x$	$24192 \cdot 2^{-32}$	17.43
23	$\texttt{40014164}_x$	$24160 \cdot 2^{-32}$	17.43
24	$\texttt{40014404}_x$	$24128 \cdot 2^{-32}$	17.44
25	$\texttt{D77FFB64}_x$	$23328 \cdot 2^{-32}$	17.49

Table 2. 25 Most Probable Transitions for $X \xrightarrow{\otimes G_0} \bar{\delta}$

Rank	Input Difference	Probability	$-log_2(p)$
1	$\texttt{7FFD7FF1}_{\texttt{x}}$	$17920 \cdot 2^{-32}$	17.87
2	$\texttt{FFF7FED1}_{\texttt{x}}$	$16640 \cdot 2^{-32}$	17.97
3	$\texttt{7FFD7FF9}_{\texttt{x}}$	$16384 \cdot 2^{-32}$	18
4	$\texttt{409004D1}_{\texttt{x}}$	$14848 \cdot 2^{-32}$	18.14
5	$\texttt{C09004D1}_{\texttt{x}}$	$14336 \cdot 2^{-32}$	18.19
6	$\texttt{7FFBF9D1}_{\texttt{x}}$	$14336 \cdot 2^{-32}$	18.19
7	$\texttt{5FDED9D1}_{\texttt{x}}$	$12480 \cdot 2^{-32}$	18.39
8	$\texttt{400801C9}_{\texttt{x}}$	$12304 \cdot 2^{-32}$	18.41
9	$\texttt{7FFBFDD9}_{\texttt{x}}$	$12288 \cdot 2^{-32}$	18.41
10	$\texttt{C00801C9}_{\texttt{x}}$	$12272 \cdot 2^{-32}$	18.41
11	$\texttt{5FFD79D1}_{\texttt{x}}$	$12032 \cdot 2^{-32}$	18.44
12	$\texttt{D41004D1}_{\texttt{x}}$	$11400 \cdot 2^{-32}$	18.52
13	$\texttt{775F7FF9}_{\texttt{x}}$	$11280 \cdot 2^{-32}$	18.53
14	$\texttt{775F7FF1}_{\texttt{x}}$	$11280 \cdot 2^{-32}$	18.53
15	$\texttt{541004D1}_{\texttt{x}}$	$10632 \cdot 2^{-32}$	18.62
16	$\texttt{77FDFE51}_{\texttt{x}}$	$10016 \cdot 2^{-32}$	18.70
17	$\texttt{FFF7D9D1}_{\texttt{x}}$	$9984 \cdot 2^{-32}$	18.71
18	$\texttt{7FDFD851}_{\texttt{x}}$	$9984 \cdot 2^{-32}$	18.71
19	$\texttt{C30011D1}_{\texttt{x}}$	$9760 \cdot 2^{-32}$	18.74
20	$\texttt{508011D1}_{\texttt{x}}$	$9632 \cdot 2^{-32}$	18.76
21	$\texttt{430011D1}_{\texttt{x}}$	$9504 \cdot 2^{-32}$	18.78
22	$\texttt{805041D1}_{\texttt{x}}$	$9472 \cdot 2^{-32}$	18.79
23	$\texttt{005041D1}_{\texttt{x}}$	$9472 \cdot 2^{-32}$	18.79
24	$\texttt{C41111D1}_{\texttt{x}}$	$9456 \cdot 2^{-32}$	18.79
25	$\texttt{908011D1}_{\texttt{x}}$	$9440 \cdot 2^{-32}$	18.79

Table 3. 25 Most Probable Transitions for $Y \xrightarrow{\otimes \mathrm{G}_3} \bar{\delta}$