

Round-Reduced Collisions of BLAKE-32

Jian Guo^{*1} and Krystian Matusiewicz²

¹ School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
guojian@ntu.edu.sg

² Department of Mathematics
Technical University of Denmark
K.Matusiewicz@mat.dtu.dk

Abstract. In this paper, we investigate the security of SHA-3 candidate BLAKE. We analyse the propagation of differences that are rotation-invariant in the internal function G . We show that by using them, it is possible to obtain near-collisions for the compression function reduced to 4 rounds out of 10. We also discuss the security of some variants of BLAKE.

Keywords. BLAKE, collision, variants, reduced-round

1 Introduction

BLAKE, a successor of LAKE [2], was designed by Aumasson *et al* [1] as a candidate for the SHA-3 competition. It follows HAIFA structure with internal wide-pipe design strategy. Our analysis focuses on the compression function, the internal function G , to be more precise. Hence here we only introduce the framework of BLAKE and describe G function in details, we leave other details [1] to the reader.

The compression function of BLAKE (refer to Figure 1) contains three parts: Initialization, a number of Rounds and Finalization.

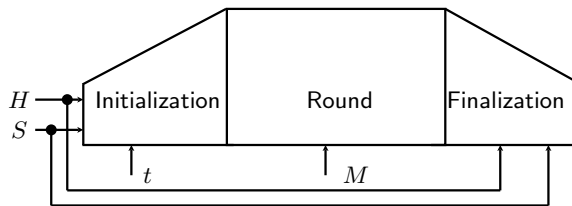


Fig. 1. Overall Structure of Compression Function of BLAKE

Initialization takes 8 words of the chaining value H , 4 words of salt S and 2 words of block index (t_0, t_1) as input and produces 16 words of internal chaining values

^{*} This work was done while visiting Technical University of Denmark and was partly supported by a DCAMM grant.

v_0, \dots, v_{15} as follows:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}.$$

Round is a bi-mapping (It is a permutation if either chaining value or message is fixed), it takes 16 word internal chaining value v_0, \dots, v_{15} and 16 word message block as input and produces the updated 16 words of the internal chaining. Each **Round** consists of 8 applications of G .

//column half-round	//diagonal half-round
$G(v_0, v_4, v_8, v_{12})$	$G(v_0, v_4, v_8, v_{12})$
$G(v_1, v_5, v_9, v_{13})$	$G(v_1, v_5, v_9, v_{13})$
$G(v_2, v_6, v_{10}, v_{14})$	$G(v_2, v_6, v_{10}, v_{14})$
$G(v_3, v_7, v_{11}, v_{15})$	$G(v_3, v_7, v_{11}, v_{15})$

Fig. 2. Compression Function: **Round**

Finalization takes 16 words chaining value from last **Round** function, 8 words chaining which is feedforward from input of SaltState, and 4 word of salt as input and produces the 8 words output of the compression function as shown below:

$$\begin{aligned} h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\ h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\ h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\ h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\ h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\ h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\ h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\ h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15} \end{aligned}$$

G function used in **Round** is based on a design of ChaCha [3]. The G function takes 4 chaining values and 2 message words as input and outputs the updated 4 chaining values. Messages are scheduled by permutation functions σ_r and xor-ed with a constant before feeding into the G function, as shown in Figure 3.

The sequence of instructions of G is shown below:

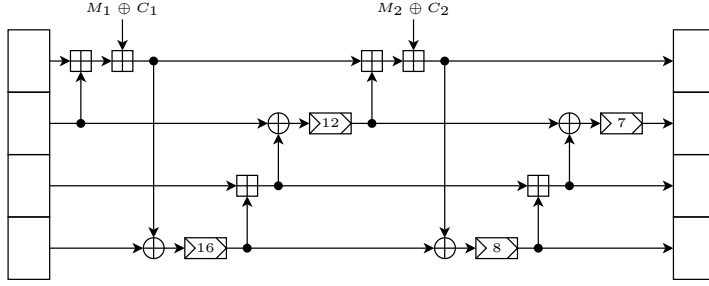


Fig. 3. G function transforms four chaining value words using two message words.

$$\begin{aligned}
a &\leftarrow a + b + (M_{\sigma_r(2i)} \oplus C_{\sigma_r(2i+1)}) \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 12 \\
a &\leftarrow a + b + (M_{\sigma_r(2i+1)} \oplus C_{\sigma_r(2i)}) \\
d &\leftarrow (d \oplus a) \ggg 8 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 7
\end{aligned}$$

Here σ is a fixed permutation used in round r , M s are message blocks and C s are constants.

2 Round Reduced Near Collisions

2.1 Linearizing G

We note that inside the function G , the number of bits rotated are 16, 12, 8 and 7. There is only one not being a multiple of 4. The idea is we use differences which are invariant with rotation by 4, such as $0x88888888$ and try to avoid differences pass through the operation “rotation by 7”. We model the compression function in \mathbb{F}_2 , where a “1” denotes a difference in the register and “0” means no difference. We linearize the G function by replacing addition with xor, and removing the rotations as the differences we choose are rotation invariant. The linearized G function is shown in Fig 4.

2.2 Round-Reduced Near Collisions

Under the linearized model, we have 16 bits of message and 16 bits of chaining values, hence the search space is 2^{32} , which is feasible. However we can further reduce the search space by taking into account “no differences pass through rotation by 7” conditions, and we aim to find 4 round-reduced collisions. As the model is linear over

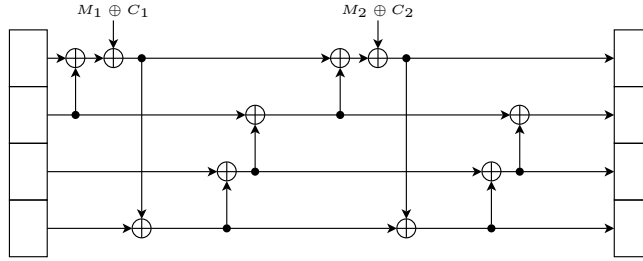


Fig. 4. linearized G function in \mathbb{F}_2

\mathbb{F}_2 , the whole compression function can be expressed by a bit vector consisting message and chaining value multiple by a matrix. By the constraint of “0” in b , we use Magma to efficiently reduce the search space to 2^4 for 4-round reduced compression function. On the other hand, to linearize the addition, each bit comes with a cost 2^1 , so linearizing a difference pattern $0x88888888$ cost 2^7 as we can get the most significant bit for free. As such, we aim to find those configurations which linearize the addition operation as less as possible. Note that by choosing proper chaining values and messages, we can get the first 1.5 rounds for free. We did the search, and the configuration with differences in $M[11]$ and $V[0,1,5,6,7,8,10,11,13,15]$ and starting point at round 6 gives count 6 only. This gives us complexity 2^{42} with no memory requirements as we can try and test the trail in sequence. This configuration gives final result $H[0, \dots, 7] = (0, 1, 1, 0, 0, 1, 1, 0)$ after feedforward (assuming there is no difference in salts). There are 4 registers containing differences, which gives 40-bits (Note $H[5]$ contains 16 bits difference) near collision. Figure 2.2 demonstrates how differences propagate in inter chaining values from round 6 to 9.

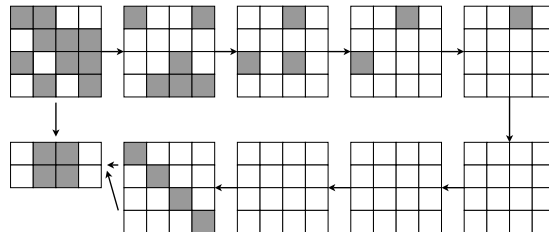


Fig. 5. Tracing the differences for 4 rounds-reduced collisions

2.3 Generalization of Rotation-Invariant Patterns

Further more, we note that with the number of rotations 16, 8, 12, only 12 is not multiple of 8. If we have differences patterns like $\alpha = 0x80808080$, $\beta = 0x08080808$ and $\alpha \oplus \beta$, this may reduce the complexity as linearizing α , β costs 2^3 and 2^4 only. Hence, within the birthday bound, we hope to extend to more rounds.

We performed a search for such configurations, but it seems that this idea does not help with reducing the complexity. Configurations which give the minimum number of linearizations are same as those using the difference pattern `0x88888888` only.

2.4 Extending to More Rounds

We note the method is limited to 4 rounds (near) collisions because forcing each output b of G reduces the search space by a factor of 2. We have 2^{32} choices for chaining and messages, each round will reduce the search space by 2^8 (8 G s in each round), hence 4 rounds is the maximum. However, instead of forcing differences in output b of G to be “0”, we allow it to be one, and hoping that it will vanish in next half round. This is possible as operation $\ggg 7$ can be treated as $(\ggg 8) \times 2$ if the MSB is 0. We verified that it is possible that carries from addition operations from other inputs may cancel this difference. Note that the differences pattern becomes `0x11111111` when rotated towards LSB by 7. Even if this differences can not be canceled, we still have near collision with additional 0.5 round as half round can only propagate the difference to at most 4 registers (actually 3).

Current experiments show that this comes with unaffordable cost, i.e. the number of linearized additions increases a lot.

3 Variants of BLAKE

If identical constants are used as a variant of BLAKE, then it can be broken as follows.

1. set all message words to be equal.
2. set $v[i] = v[i + 1] = v[i + 2] = v[i + 3]$ for $i = 0, 4, 8, 12$.

The final result of the compression function will retain the symmetry property, i.e. $h[0] = h[1] = h[2] = h[3]$ and $h[4] = h[5] = h[6] = h[7]$. To find a collision of this variant, it costs 2^{32} for BLAKE-32 and 2^{64} BLAKE-64.

This does not apply to unmodified BLAKE because $v[12] \neq v[13]$ as different constants are xor-ed with $t[0]$, and similarly $v[14] \neq v[15]$ for $t[1]$.

4 Conclusion

In this paper, we presented a near collision attack for 4 round-reduced BLAKE and discussed security level for a variant using identical constants.

References

1. J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan. SHA-3 proposal BLAKE. Submission to NIST, 2008.
2. J.-P. Aumasson, W. Meier, and R. C.-W. Phan. The hash function family LAKE. In K. Nyberg, editor, *FSE*, volume 5086 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 2008.
3. D. J. Bernstein. Chacha, a variant of salsa20, 2008. Available at <http://cr.yp.to/chacha.html>.