Volume : 2 | Issue : 1 | Jan 2013 • ISSN No 2277 - 8160

Research Paper | Computer Science

Rationale and analysis of BLAKE

Kalpesh R. Rakholiya Research Scholor of C.M.J. University , Meghalaya To Junagadh, Uday nagar block no 7/b, b/h godhwani high school, timbawadi

Dr. Dhaval Kathiriya Research Guide in computer science, C.M.J. University ,Meghalaya

ABSTRACT

This paper reports elements of analysis of BLAKE, with a focus on BLAKE-32. We study properties of the function's components, resistance to generic attacks, and dedicated attack strategies. A more extensive analysis can be found in the submission document sent to NIST.

KEYWORDS: Permutation, Compare, Cipher

Choosing Permutations :
The permutations σ0 , . . . , σ9 were chosen to match several security criteria:

1. No message word should be input twice at the same point.
2. No message word should be XOR'd twice with the same constant.
3. Each message word should appear exactly five times in a column step and exactly five times in a diagonal step.
4. Each message word should appear exactly five times in first position in G and exactly five times in second position.

This is equivalent to say that, respectively:
1. For all i = 0, . . . , 15, there should exist no distinct permutations σ, σ′ such that σ(i) = σ′(i).
2. No pair (i, j) should appear twice at a position of the form (2k, 2k + 1), for all k = 0, . . . , 7.
3. For all i = 0, . . . , 15, there should be exactly five distinct permutations σ such that σ(i) < 8, and exactly five such that σ(i) > 8.
4. For all i = 0, . . . , 15, there should be exactly five distinct permutations σ such that σ(i) is even, and exactly five such that σ(i) is odd.

These criteria imply that a difference in some given word will produce distinct differential trails at each round, and ensure a balanced distribution of the message bits within the implicit algebraic normal form.

In BLAKE-64, four of the permutations are repeated because it makes fourteen rounds instead of ten. The above criteria thus just apply to the first ten rounds. The slight loss of balance in the four last rounds seems unlikely to affect security.

Compression Function :
This Paper gives a bottom-up analysis of BLAKE's compression function, starting with the low-level algorithm, and finishing with the structure of the compression function.

1.The G Function
The G function is inspired from the "quarter-round" function of the stream cipher ChaCha, which transforms (a, b, c, d) as follows:

a d	← ←	a + b (d ⊕ a) ≫ 16
c b	← ←	c + d (b ⊕ c) ≫ 12
a d	← ←	a + b (d ⊕ a) ≫ 8
c b	← ←	c + d (b ⊕ c) ≫ 7

To build BLAKE's compression function on ChaCha, we add input of two message words and constants, and let the function be otherwise unchanged. We keep the rotation distances of ChaCha, which provide a good tradeoff security/efficiency: 16- and 8-bit rotations preserve byte alignment, so are fast on 8-bit processors (no rotate instruction is needed), while 12- and

7-bit rotations break up the byte structure, and are reasonably fast.

ChaCha's function is itself a modified version of the "quarter round" of the stream cipher Salsa20 . The idea of a 4×4 state with four parallel mappings for rows and columns goes back to the cipher Square [1], and was then successfully used in Rijndael [2], Salsa20 and ChaCha. Detailed design rationale and preliminary analysis of ChaCha and Salsa20 can be found in [3, 4], and its cryptanalysis can be found in [5].

G can be easily inverted: given a message m, and a round index r, the inverse function of G_i is defined as follows:

b ← c ⊕ (b ≪ 7)
c ← c − d
d ← a ⊕ (d ≪ 8)
a ← a − b − (mσr (2i+1) ⊕ cσr (2i))
b ← c ⊕ (b ≪ 12)

c ← c − d
d ← a ⊕ (d ≪ 16)
a ← a − b − (mσr (2i) ⊕ cσr (2i+1))

Hence for any (a′, b′, c′, d′), one can efficiently compute the unique (a, b, c, d) such that Gi(a, b, c, d) = (a′, b′, c′, d′), given i and m. In other words, Gi is a permutation of {0, 1}128 .

We found several linear approximations of differentials; the notation m $(\Delta_0, \Delta_1, \Delta_2, \Delta_3) \mapsto (\Delta'_0, \Delta'_1, \Delta'_2, \Delta'_3)$

inputs with the leftmost difference lead to outputs with the rightmost difference, when $(m_{\sigma r\,(2i+1)} \oplus c_{\sigma r\,(2i)}) = (m_{\sigma r\,(2i)} \oplus c_{\sigma r\,(2i+1)}) = 0$. For random inputs we have for example

$$(80000000, 00000000, 80000000, 80008000) \mapsto (80000000, 0, 0, 0)$$
$$(00000800, 80000800, 80000000, 80000000) \mapsto (0, 0, 80000000, 0)$$
$$(80000000, 80000000, 80000080, 00800000) \mapsto (0, 0, 0, 80000000)$$

with respective probabilities 1, 1/2, and 1/2. Many such probability differentials can be iden- tified for G, and one can use standard message modification techniques (linearization, neutral bits) to identify a subset of inputs over which the probability is much higher than over the whole domain. Similar linear differentials exist in the Salsa20 function, and were exploited [5] to attack the compression function Rumba [6], breaking 3 rounds out of 20.

Other noteworthy properties of G are that the only fixed point in G is the zero input, and that no preservation of differences can be obtained by linearization. The first observation is straightforward when writing the corresponding equations. The second one means that there exist no pair of inputs whose difference (with respect to XOR) is preserved in the corresponding pair of outputs, in the linearized model. This follows from the fact that, if an input difference gives the same difference in the output, then this difference must be a fixed point for G, since the only fixed point is the null value, there exists no such difference.

2 Round Function
Recall that the round function of BLAKE computes

$G_0(v_0, v_4, v_8, v_{12})$ $G_1(v_1, v_5, v_9, v_{13})$ $G_2(v_2, v_6, v_{10}, v_{14})$ $G_3(v_3, v_7, v_{11}, v_{15})$
$G_4(v_0, v_5, v_{10}, v_{15})$ $G_5(v_1, v_6, v_{11}, v_{12})$ $G_6(v_2, v_7, v_8, v_{13})$ $G_7(v_3, v_4, v_9, v_{14})$

Because G is a permutation, a round is a permutation of the inner state v for any fixed message. In other words, given a message and the value of v after r rounds, one can determine the value of v at rounds r − 1, r − 2, etc., and thus the initial value of v. Therefore, for a same initial state a sequence of rounds is a permutation of the message. That is, one cannot find two messages that produce the same internal state, after any number of rounds.

After one round, all 16 words are affected by a modification of one bit in the input (be it the message, the salt, or the chaining value). Here we illustrate diffusion through rounds with a concrete example, for the zero message and the zero initial state. The matrices displayed below represent the differences in the state after each step of the first two rounds (column step, diagonal step, column step, diagonal step), for a difference in the least significant bit of v0 :

In comparison, in the linearized model (i.e., where all additions are replaced by XOR's), we have:

$$\text{column step} \begin{pmatrix} 00000011 & 00000000 & 00000000 & 00000000 \\ 20220202 & 00000000 & 00000000 & 00000000 \\ 11010100 & 00000000 & 00000000 & 00000000 \\ 11000100 & 00000000 & 00000000 & 00000000 \end{pmatrix} \text{(weight 14)}$$

$$\text{diagonal step} \begin{pmatrix} 00000101 & 10001001 & 10011010 & 02202000 \\ 40040040 & 22022220 & 00202202 & 00222020 \\ 01110010 & 20020222 & 01111101 & 00111101 \\ 01110001 & 10100110 & 22002200 & 01001101 \end{pmatrix} \text{(weight 65)}$$

$$\text{column step} \begin{pmatrix} 54500415 & 13012131 & 02002022 & 20331103 \\ 2828A0A8 & 46222006 & 04006046 & 64646022 \\ 00045140 & 30131033 & 12113132 & 10010011 \\ 00551045 & 23203003 & 03121212 & 01311212 \end{pmatrix} \text{(weight 125)}$$

$$\text{diagonal step} \begin{pmatrix} 35040733 & 67351240 & 24050637 & B1300980 \\ 27472654 & 8AE6CA08 & EE4A6286 & E08264A8 \\ 03531247 & 1AB89238 & 54132765 & 55051040 \\ 14360705 & 73540643 & 89128902 & 70030514 \end{pmatrix} \text{(weight 186)}$$

The higher weight in the original model is due to the addition carries induced by the constants c0, . . . , c15 . A technique to avoid carries at the first round and get a low-weight output difference is to choose a message such that m0 = c0, . . . , m15 = c15 . At the subsequent rounds, however, nonzero words are introduced because of the different permutations.

$$(80000000, 00000000, 80000000, 80008000) \mapsto (80000000, 0, 0, 0)$$

Diffusion can be delayed a few steps by combining high-probability and low-weight dif- ferentials of G, using initial conditions, neutral bits, etc. For example, applying directly the differentialthe diffusion is delayed one step, as illustrated below:

$$\text{column step} \begin{pmatrix} 80000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \end{pmatrix} \text{(weight 1)}$$

$$\text{diagonal step} \begin{pmatrix} 800003E8 & 00000000 & 00000000 & 00000000 \\ 00000000 & 0B573F03 & 00000000 & 00000000 \\ 00000000 & 00000000 & AB9F819D & 00000000 \\ 00000000 & 00000000 & 00000000 & E8800083 \end{pmatrix} \text{(weight 49)}$$

$$\text{column step} \begin{pmatrix} 8007E4A0 & 2075B261 & 18E78828 & 9800099E \\ 5944FE53 & F178A22F & 86B0A65B & 936C73CB \\ A27F0D24 & 98D6929A & 4088A5FB & 2E39EDA3 \\ A08FFF64 & 2AD374B7 & 2818E788 & 1E9883E1 \end{pmatrix} \text{(weight 236)}$$

$$\text{diagonal step} \begin{pmatrix} 4B3CBDD2 & 0290847F & B4FF78F9 & F1E71BA3 \\ 3A023C96 & 49908E86 & F13BC1D7 & ADC2020A \\ 9DCA344A & 827BF1E5 & B20A8825 & FE575BE3 \\ FC81FE81 & D676FFC9 & 80740480 & 52570CB2 \end{pmatrix} \text{(weight 252)}$$

In comparison, for a same input difference in the linearized model we have

$$\text{column step} \begin{pmatrix} 80000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \end{pmatrix} \text{(weight 1)}$$

$$\text{diagonal step} \begin{pmatrix} 80000018 & 00000000 & 00000000 & 00000000 \\ 00000000 & 10310101 & 00000000 & 00000000 \\ 00000000 & 00000000 & 18808080 & 00000000 \\ 00000000 & 00000000 & 00000000 & 18800080 \end{pmatrix} \text{(weight 18)}$$

$$\text{column step} \begin{pmatrix} 80000690 & E1101206 & 0801B818 & B8000803 \\ 1D217176 & 600FC064 & 60111212 & 22167121 \\ 90B0B886 & 16E12133 & 00888138 & 83389890 \\ 90803886 & 17E01122 & 18001180 & 83B88010 \end{pmatrix} \text{(weight 155)}$$

$$\text{diagonal step} \begin{pmatrix} 44E4E456 & 1334688D & D8BDA164 & 0F649833 \\ 4E20F629 & 563A9099 & A62F3969 & 7773C0BE \\ FEB6F508 & AABDCBF9 & 32622291 & 87A10D6A \\ 3C2B867B & B603B05C & DA695123 & F88E8007 \end{pmatrix} \text{(weight 251)}$$

These examples show that even in the linearized model, after two rounds about half of the state bits have changed when different initial states are used (similar figures can be given for a difference in the message). Combinations of low-weight differentials and of message modifications may help to attack reduced-round versions of BLAKE. However, differences after more than four rounds seem difficult to control.

Conclusion :

BLAKE partially inherits the security of ChaCha for its compression function, and uses a mode of operation that, although simple, was proved to induce no vulnerability. The recommended number of rounds ensure a comfortable security margin: BLAKE-32 makes ten rounds, which is equivalent to 20 rounds of the stream cipher ChaCha, for which the best attack exploits a truncated differential over only three rounds (so 1.5 of BLAKE-32). The absence of external attack despite the simplicity of the design suggests that even reduced versions of BLAKE are difficult to attack. Finally, one may compare BLAKE with the AES: both operate on a 4×4 matrix column-wise and diagonal-wise, and both achieve full diffusion after two rounds and make ten or 14 rounds depending on the size of the key (for AES) or of the words (for BLAKE).

REFERENCES

[1] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In FSE, 1997. | [2] Joan Daemen and Vincent Rijmen. Rijndael for AES. In AES Candidate Conference, | [3] 2000. Daniel J. Bernstein. ChaCha, a variant of Salsa20. In SASC. ECRYPT, 2008. | [4] Daniel J. Bernstein. The Salsa20 family of stream ciphers. In New Stream Cipher Designs, 2008. See also [29]. | [5] Daniel J. Bernstein. The Rumba20 compression function. See also [31].