A Note on Linear Approximations of BLUE MIDNIGHT WISH Cryptographic Hash Function

Vlastimil Klima[1] and Petr Susil[2]

[1] Independent cryptologist - consultant, Czech Republic
v.klima@volny.cz
[2] EPFL PhD student
susil.petr@gmail.com

Abstract. BLUE MIDNIGHT WISH hash function is the fastest among 14 algorithms in the second round of SHA-3 competition [1]. At the beginning of this round authors were invited to add some tweaks before September 15th 2009. In this paper we discuss the tweaked version (BMW). The BMW algorithm [3] is of the type AXR, since it uses only operations ADD (sub), XOR and ROT (shift). If we substitute the operation ADD with operation XOR, we get a BMW_{lin}, which is an affine transformation. In this paper we consider only a BMW_{lin} function and its building blocks.
These affine transformations can be represented as a linear matrix and a constant vector. We found that all matrices of main blocks of BMW_{lin} have a full rank, or they have a rank very close to full rank. The structure of matrices was examined. Matrices of elementary blocks have an expected non-random structure, while main blocks have a random structure. We will also show matrices for different values of security parameter $ExpandRounds_1$ (values between 0 and 16). We observed that increasing the number of rounds $ExpandRounds_1$ tends to increase randomness as was intended by designers. These observations hold for both $BMW256_{lin}$ and $BMW512_{lin}$. In this analysis we did not find any useful property, which would help in cryptanalysis, nor did we find any weaknesses of BMW. The study of all building blocks will follow.

1 Introduction

The BMW algorithm [3] is of the type AXR, since it uses only operations XOR, ADD (sub) and ROT (shift). If we substitute the operation ADD with operation XOR, we get a BMW_{lin}, which is an affine transformation. In this paper we consider only a BMW_{lin} and its building blocks. These are affine transformations, which can be represented by matrices. We present the rank and the structure of these binary matrices. Main

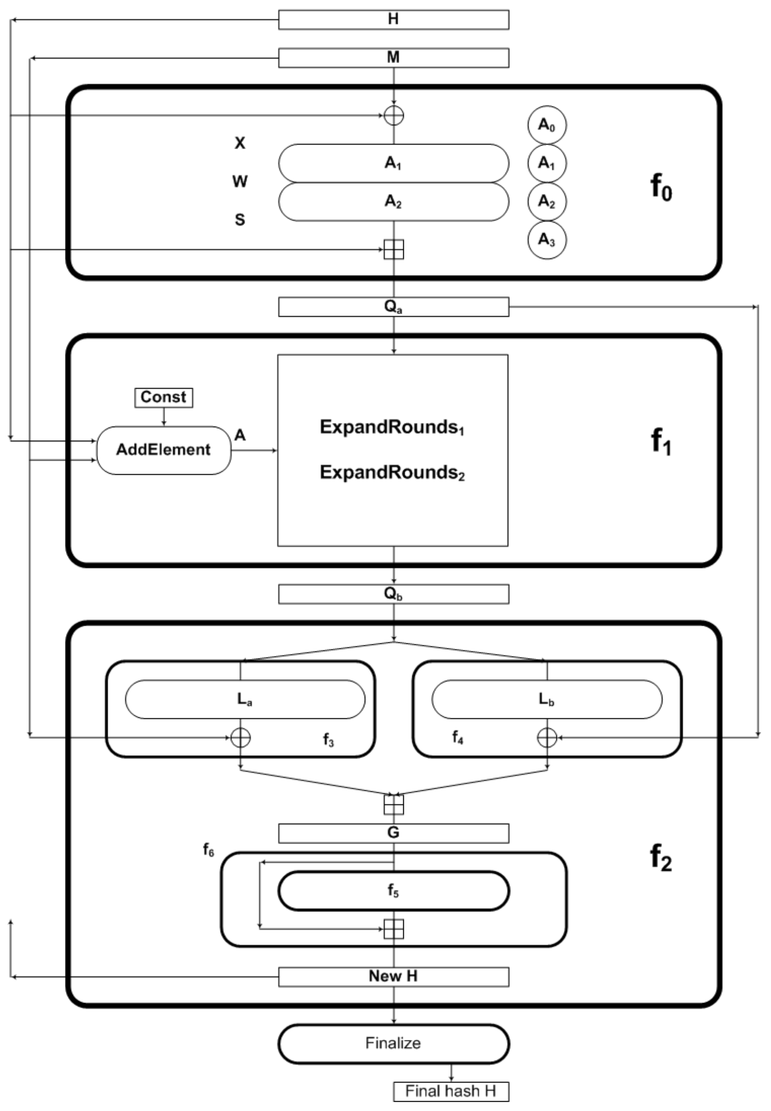

Fig. 1. The scheme of (tweaked) BMW [2]

blocks of BMW are f_0, f_1, f_2, they produce three intermediate values Q_a,Q_b, G, and a chaining value H. All these variables consist of 16 w-bit words. The variables depend only on the input block M and an old hash value H. These dependencies will be shown in matrices. Note the rank and the structure for each matrix. We will also present matrices for different values of security parameter $ExpandRounds_1$ (values between

0 and 16). Let us remind the reader that after $ExpandRounds_1$ rounds of the function $expand_1$ there follow $(16 - ExpandRounds_1)$ rounds of the function $expand_2$ [3]. For the simplicity of this paper we suppose that the reader is familiar with the basic description of BMW [3]. We use a simple notation H, but whenever a misunderstanding is possible we distinguish between $oldH$ and $newH$. Our observations and conclusions hold for both $BMW256_{lin}$ and $BMW512_{lin}$, but we present results only for $BMW256_{lin}$.

In the following picture we can see linear dependencies between M and Q_a. There are two matrices of the type 512×512 (separated by a black stripe) which we denote $(M,\, Q_a)$. The columns represent 512 variables (bits) of M on the left side and 512 variables (bits) of Q_a on the right side. Lines represents linear equations between bits on the left side (M) and on the right side (Q_a). Since the matrix on the right is an identity matrix, it gives us direct expressions for bits of Q_a with variables from M. Every line can be expressed as a linear equation: $\bigoplus_{i \in Left} x_i = \bigoplus_{j \in Right} y_j$. Indexes of bits which are included in the equation are denoted as black dots. In the picture we can see (among other things) dependencies of the first word of Q_a on five words of M, which follows from the transformation A_1 in [3].

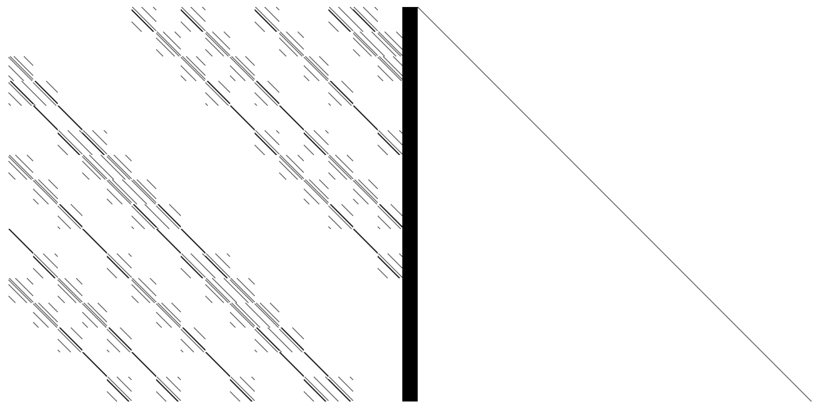

Fig. 2. Example of dependencies between variables (bits) of M and Q_a, denoted as $(M,\, Q_a)$

In the Fig. 2 we can see the dependency of Q_a on M. To get the dependency of M on Q_a, we need to find an inverse matrix. As we know from linear algebra, xoring one line with another or changing the order of lines does not change the rank of the matrix. Using these two operation in our elimination algorithm we transform the pair of matrices (A, I) to (I', A'), where I is an identity matrix and I' is identity matrix if A has a full rank. If I' is an identity matrix then matrix A' gives dependencies of M on Q_a, as is shown in the following picture. If the matrix A is singular, the matrix I' will have nonempty columns above the diagonal. The variable (bit), which corresponds to this nonempty column, creates linear dependencies only with other variables (bits) in the matrix, see for instance Fig. 4 (Zoom on matrix with rank = MAX-1).

The most important dependencies are shown in the following section.

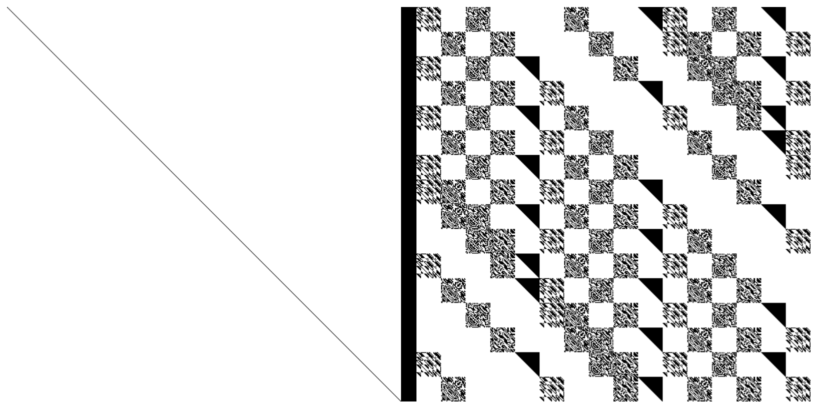
Fig. 3. (M, Q_a) - dependency between M and Q_a

Elimination algorithm - transformation of (A, I) to (I', A'):

```
Input:  boolean matrix A[LEN][LEN]
Output: pair of boolean matrices (I',A')
var boolean matrix I[LEN][LEN], where I[i][j]=1 if i==j and I[i][j]=0 otherwise

for i=1 to LEN
begin
      if A[i][i] then
        for all j != i and A[j][i] //add line i to line j in (A, I)
           for all k
               A[j][k] = A[j][k] + A[i][k] and I[j][k] = I[j][k] + I[i][k];
      else
```

```
        for all j > i
           if A[j][i]    //switch line j and i in (A, I)
              for all k
                  switch values A[j][k] and A[i][k] and I[j][k] and I[i][k]
              go to begin (without increasing i) or go to end (if i = LEN)
end
output (A,I)
```

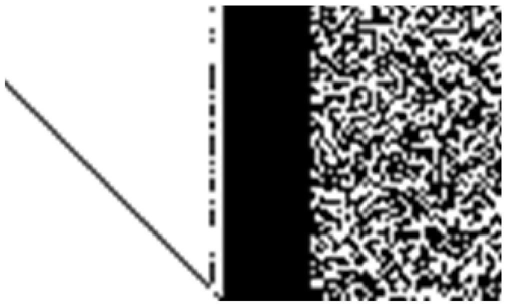

Fig. 4. Zoom on matrix with rank = MAX-1

2 Dependency of Q_a on M

We can see the dependency in the Fig. 2 and 3. Note that matrices do not depend on the value $ExpandRounds_1$, since Q_a is created before the function f_1. Rank of matrix (M, Q_a) is 512 i.e. the matrix has a full rank.

3 Dependency of Q_a on H

Also rank of matrix $(H,\ Q_a)$ is 512. Note that the matrix expresses the dependence in the block f_0, so it does not depend on the value $ExpandRounds_1$.

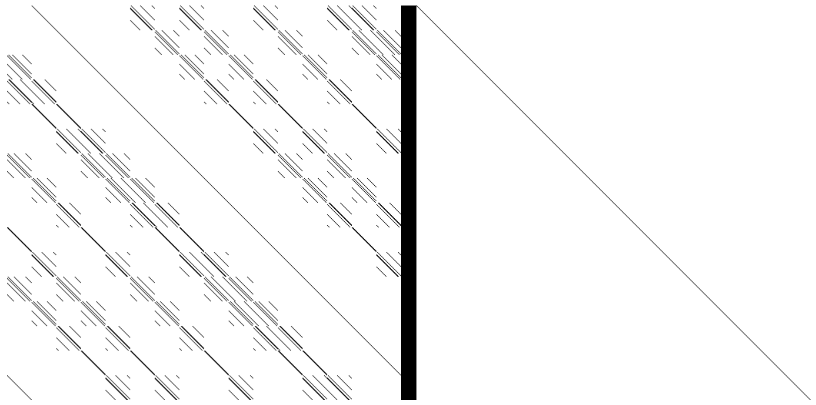

Fig. 5. $(H,\ Q_a)$

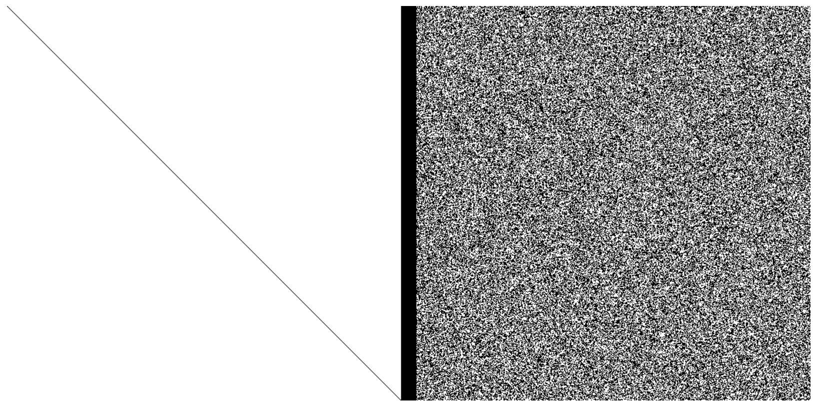

Fig. 6. $(H,\ Q_a)$

4 Dependency of Q_b on M

We can see here for the first time that matrices do not have full rank and that increasing the number of rounds $ExpandRounds_1$ tends to increase the randomness in dependencies. Since all other dependencies (among other variables) are similar, in next sections we will show only ranks of matrices.

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8
$Rank(M, Q_b)$	511	512	512	510	511	512	512	511	510

$ExpandRounds_1$	9	10	11	12	13	14	15	16
$Rank(M, Q_b)$	511	511	512	512	510	512	511	510

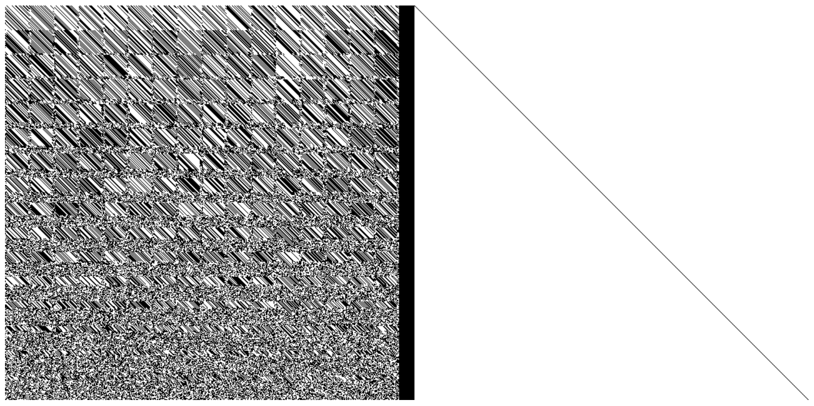

Fig. 7. (M, Q_b), $ExpandRounds_1$=0

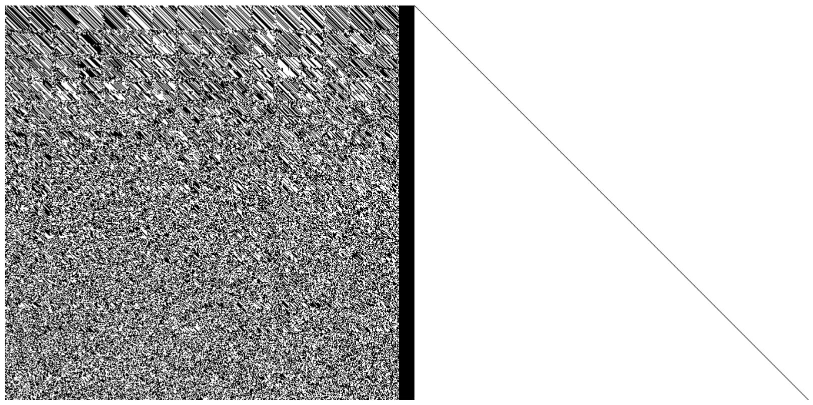

Fig. 8. $(M,\, Q_b)$, $ExpandRounds_1$=2

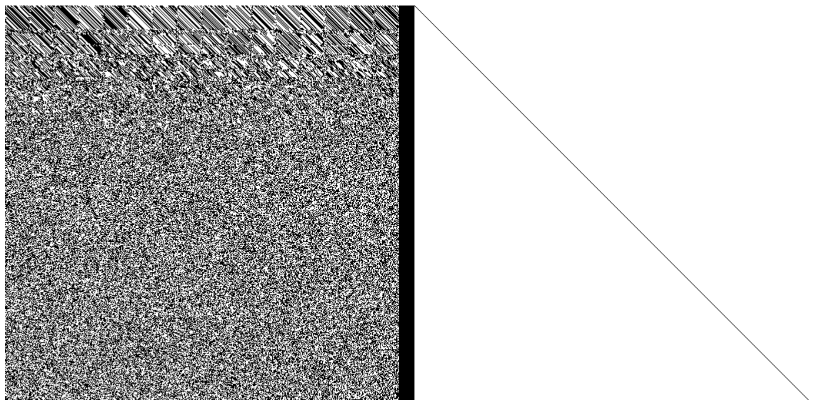

Fig. 9. $(M,\, Q_b)$, $ExpandRounds_1$=16

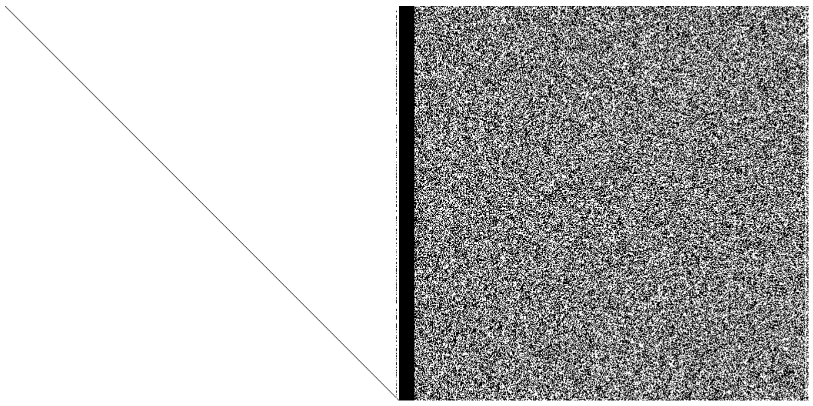

Fig. 10. (M, Q_b), $ExpandRounds_1$=0

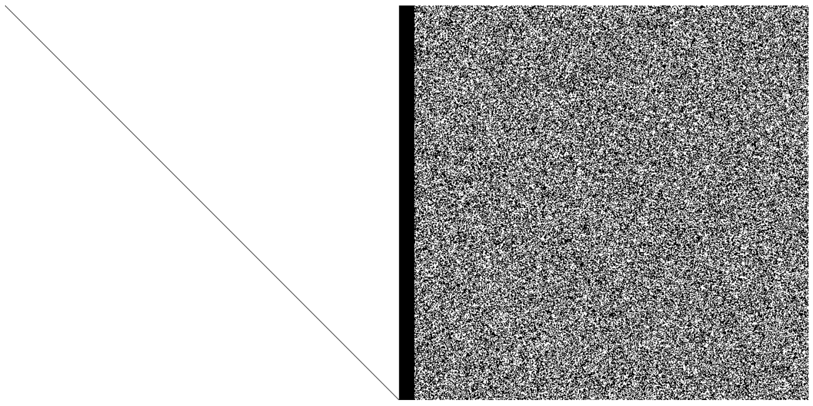

Fig. 11. (M, Q_b), $ExpandRounds_1$=2

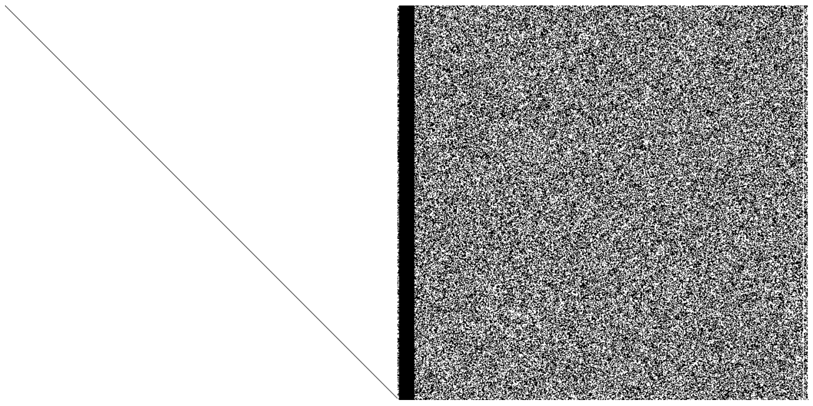

Fig. 12. (M, Q_b), $ExpandRounds_1$=16

5 Dependency of Q_b on $oldH$

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8
$Rank(oldH, Q_b)$	512	510	509	511	511	511	511	510	512

$ExpandRounds_1$	9	10	11	12	13	14	15	16
$Rank(oldH, Q_b)$	511	511	512	510	511	512	511	511

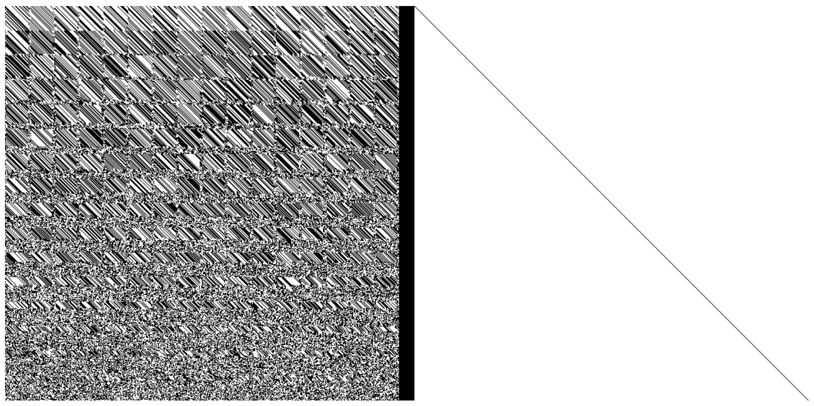

Fig. 13. ($oldH$, Q_b), $ExpandRounds_1$=0

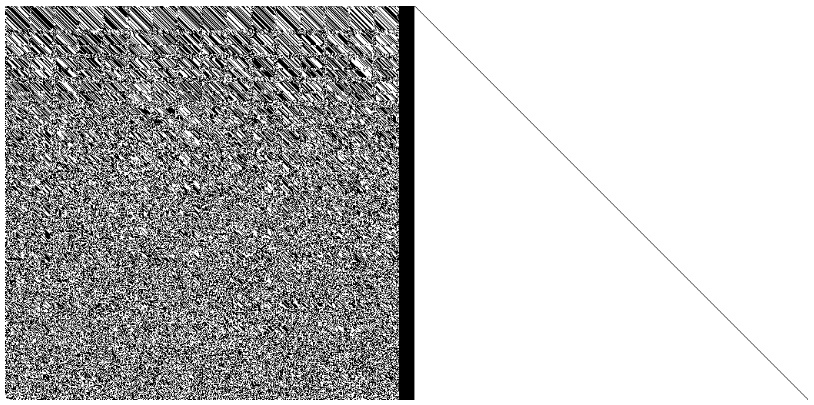

Fig. 14. $(oldH,\ Q_b)$, $ExpandRounds_1{=}2$

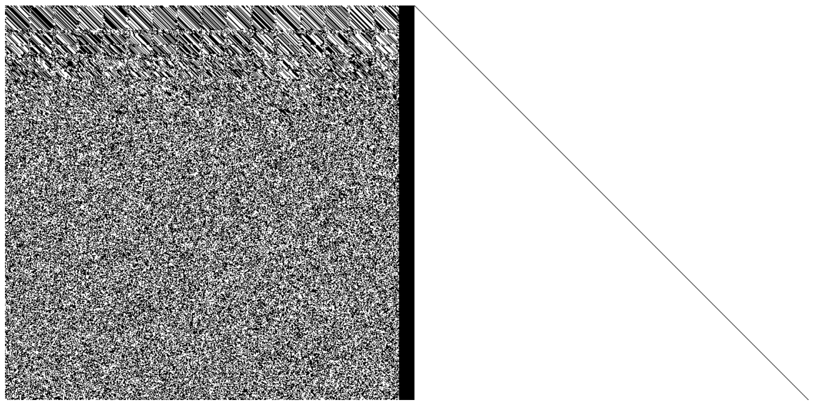

Fig. 15. $(oldH,\ Q_b)$, $ExpandRounds_1{=}16$

6 Dependency of G on M

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8
$Rank(M, G)$	512	510	511	512	510	511	510	512	511

$ExpandRounds_1$	9	10	11	12	13	14	15	16
$Rank(M, G)$	510	510	510	511	512	512	510	511

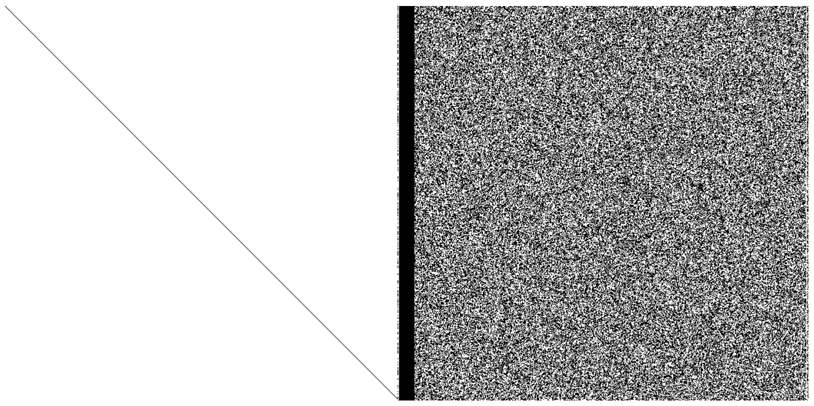

Fig. 16. (M, G), $ExpandRounds_1$=2

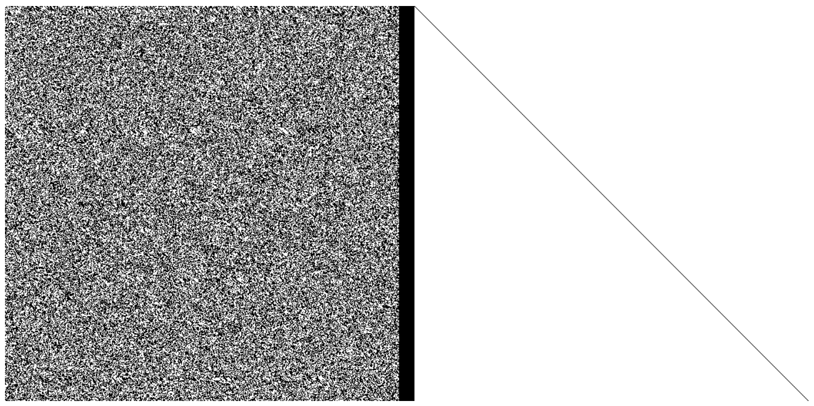

Fig. 17. (M, G), $ExpandRounds_1$=2

7 Dependency of G on $oldH$

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$Rank(oldH, G)$	511	511	510	511	511	511	512	511	511	512	510	510	511	512	512	511	511

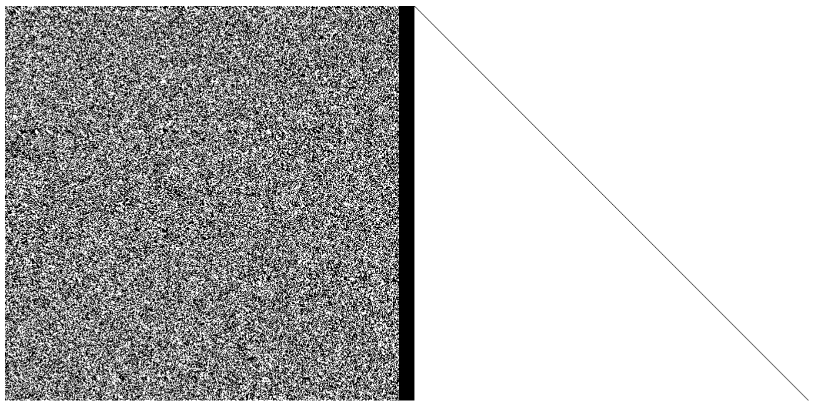

Fig. 18. $(oldH,\, G)$, $ExpandRounds_1$=2

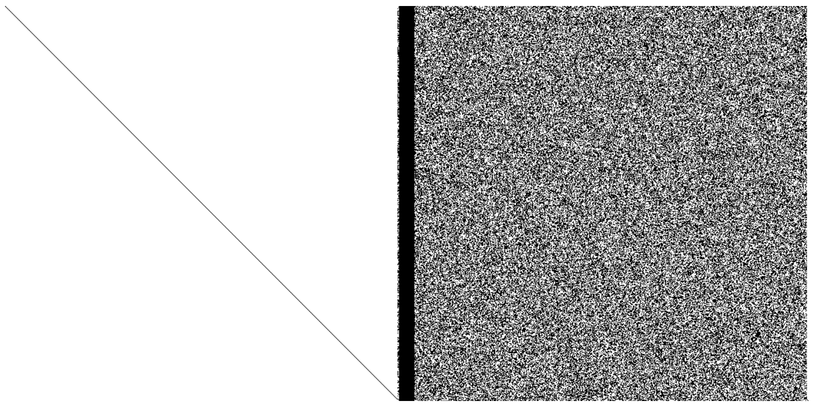

Fig. 19. $(oldH,\, G)$, $ExpandRounds_1$=2

8 Dependency of *newH* on *M*

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8
$Rank(oldH, G)$	512	510	511	512	510	511	510	512	511

$ExpandRounds_1$	9	10	11	12	13	14	15	16
$Rank(oldH, G)$	510	510	510	511	512	512	510	511

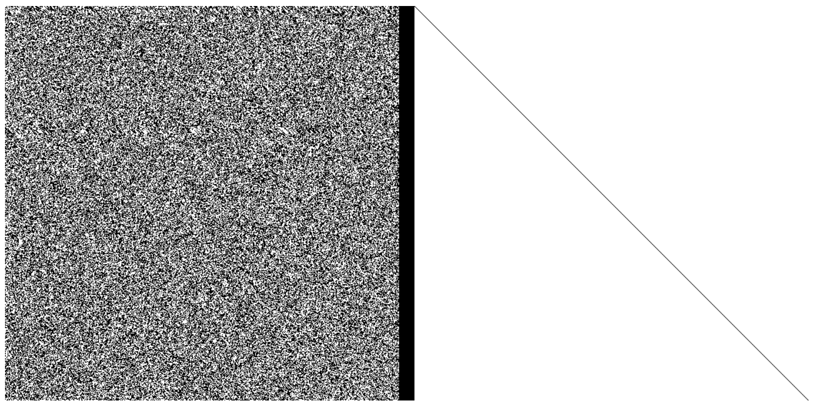

Fig. 20. $(M$, $newH)$, $ExpandRounds_1$=2

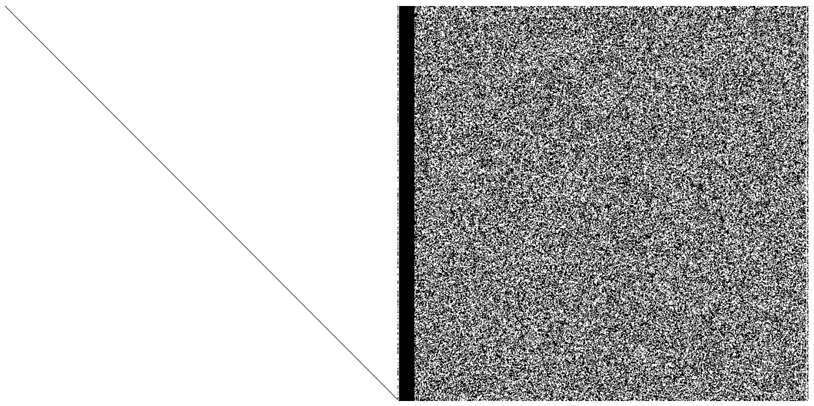

Fig. 21. $(M$, $newH)$, $ExpandRounds_1$=2

9 Dependency of *newH* on *oldH*

$ExpandRounds_1$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$Rank(oldH, newH)$	511	511	510	511	511	511	512	511	511	512	510	510	511	512	512	511	511

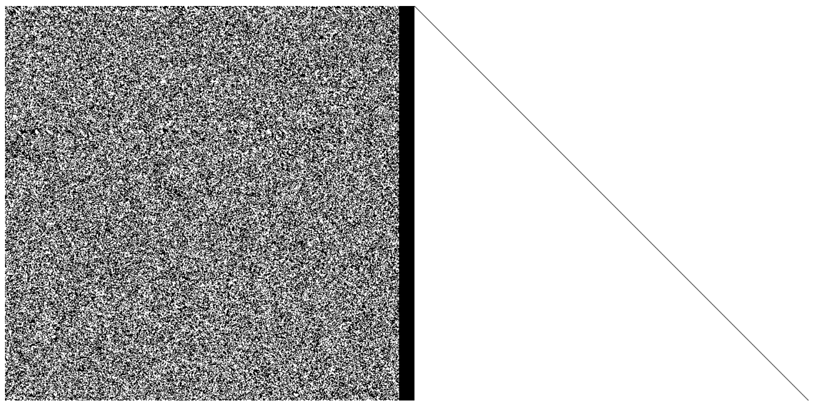

Fig. 22. $(oldH,\ newH)$, $ExpandRounds_1$=2

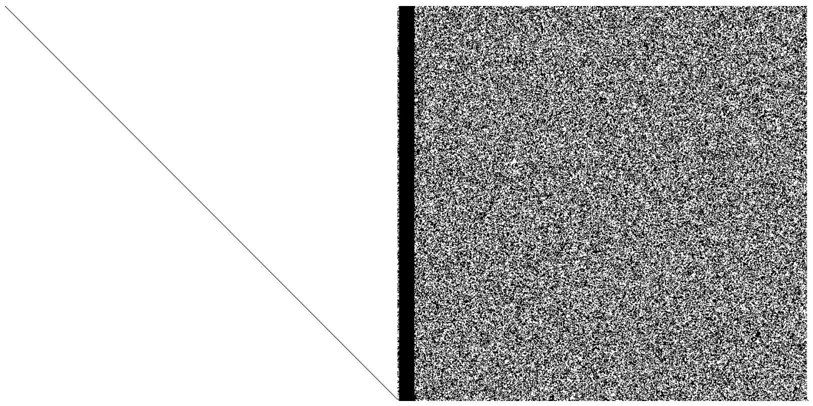

Fig. 23. $(oldH,\ newH)$, $ExpandRounds_1$=2

10 Conclusion

Results presented in this paper are only a small part of analysis which has been performed. As the reader has noticed, only variables M and H have been used as inputs for BMW_{lin}. However, some elementary/main blocks can be analyzed separately, which would give a better insight into the structure of BMW hash function. Results of the complete analysis will follow. In this paper all dependencies form matrices with a full or an almost full rank. The structure of these matrices was expected as well. Matrices of elementary blocks have a non random structure, but they fulfill requirements to mix variables (bits) fast. Matrices of main blocks have a full or an almost full rank and a random structure. We observed that if it is required (and possible) to increase randomness, it can be done by increasing the number of rounds $ExpandRounds_1$. These observations hold for both versions BMW256 and BMW512.

References

1. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family, 2007, NIST, http://csrc.nist.gov/groups/ST/hash/index.html.
2. Vlastimil Klima and Danilo Gligoroski, On BLUE MIDNIGHT WISH decomposition, to be published
3. Danilo Gligoroski and Vlastimil Klima and Svein Johan Knapskog and Mohamed El-Hadedy and Joørn Amundsen and Stig Frode Mjølsnes, Cryptographic Hash Function BLUE MIDNIGHT WISH. Submission to NIST, September 15, 2009. http://people.item.ntnu.no/~danilog/Hash/BMW-SecondRound/Supporting_Documentation/BlueMidnightWishDocumentation.pdf