Non-randomness in the Sarmal Compression Function[⋆]

Nicky Mouha[1,2,⋆⋆], Tor E. Bjørstad[3], and Bart Preneel[1,2,⋆⋆]

[1] Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
[2] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
[3] The Selmer Center, Department of Informatics, University of Bergen, Norway.

Abstract. Sarmal is a hash function submitted to the NIST SHA-3 hash function competition. The design and structure of Sarmal is quite similar to that of ARIRANG, another SHA-3 candidate. We analyse the impact and applicability of recent attacks by Guo et al. on ARIRANG, with respect to Sarmal. Our results indicate that Sarmal is less vulnerable against this line of attack; in particular we were not able to obtain pseudo-collisions for Sarmal faster than using a generic attack. However, we have found that the compression function of Sarmal can be distinguished from a pseudorandom function with probability one, using only two compression function calls. This result is specific to the compression function, and does not seem extensible to the full hash function.

Keywords: Cryptanalysis, hash function, Sarmal, pseudocollision, non-randomness.

1 Introduction

Sarmal [8], designed by Varıcı, Özen and Kocair, is a candidate for the SHA-3 hash function competition [7]. Another SHA-3 candidate, ARIRANG [2], was recently attacked by Guo et al. [3]. Because the round function of ARIRANG and Sarmal have similar structure (the designs are in both cases influenced by that of the hash function FORK-256 [4]), we have examined the security of Sarmal with respect to this type of attack.

2 Brief Description of the Sarmal Compression Function

Sarmal is built upon the HAIFA [1] framework, and produces a digest size of either 224, 256, 384 or 512 bits. All operations in Sarmal are on 64-bit words. To hash a message, it is padded and split into 1024-bit blocks, which are iteratively processed by the Sarmal compression function f. The final hash value is obtained by truncating (for digest sizes less than 512 bits) the final hash state. The only differences between the different digest sizes are the IVs, the constants, the number of rounds of the compression function, and which bits are truncated at the end.

The compression function f operates on two independent, parallel branches, which are combined at the end using xor. Finally, a Davies-Meyer feed-forward is applied to produce the next chaining value. Each branch consists of 16 or 20 calls to the round function, denoted G, shown in Fig. 1. In each round, four reordered words of the message block M_i are used as input. Earlier cryptanalysis of Sarmal [5, 6] has found that the branches are combined in a way which is vulnerable to the generalised birthday attack [9]. To the best of the authors' knowledge, no other independent cryptanalysis of Sarmal exists.

As can be seen from Fig. 1, the round function G contains two calls to an inner function g. This g-function provides a large part of the nonlinearity in Sarmal. It consists of a layer of eight

[⋆] The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

[⋆⋆] This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

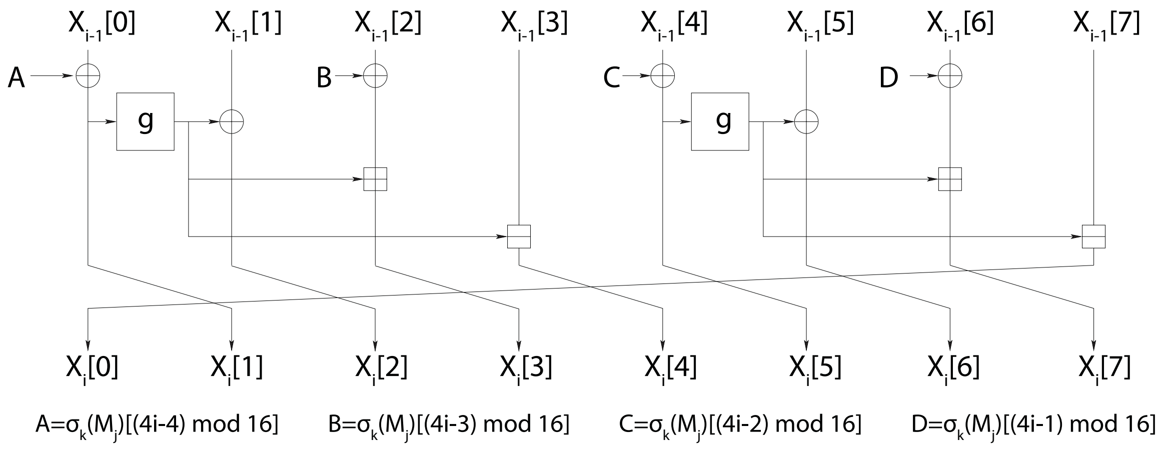

Fig. 1. The Sarmal G-function

parallel 8-bit S-boxes, and an 8x8 MDS matrix. This is essentially the same construction as the $G^{(512)}$ function in ARIRANG-512, although the particular S-box and MDS matrix in Sarmal are both different. For further information about Sarmal, we refer to the full specification [8].

3 Choosing the Type of Difference

In the recent attack on ARIRANG [3], the authors use symmetric differences; in other words, each message or state word either contains no difference, or $\Delta = \texttt{0xFF...FF}$. Restricting oneself to this pair of differences has several nice properties: they are invariant under rotation, are a closed set under xor, and pass through the S-box and MDS layer of ARIRANG with a "good" probability (2^{-28} for ARIRANG-256).

Unlike ARIRANG, Sarmal does not contain any rotations, so we can in principle use any fixed pair $(0, \delta)$ in the same manner. However, the only xor difference that always passes through the modular additions and subtractions in Sarmal[4], is a difference in the most significant bit, `0x80...00`. Because the MDS layer in the g-function spreads any single-byte difference to all eight bytes, this difference is not useful (in this context) for characteristics that contain active g-functions.

The selection heuristic for a good differential characteristic should thus be to minimise the number of active g-functions, as well as the number of active additions and subtractions, since these will all have to be satisfied either probabilistically or through message modification. As a first-order estimate of the overall attack complexity, we have only considered the number of active g-functions, and assumed that the modular operations behave differentially as xor.

Finding a good differential for the g-function can be done by examining the MDS and S-box layers separately. The MDS matrix used in Sarmal has only one eigenvalue, `0x0D`, with multiplicity 8 and the corresponding eigenvector $[\texttt{0x01}, \texttt{0x01}, \texttt{0x01}, \texttt{0x01}, \texttt{0x01}, \texttt{0x01}, \texttt{0x01}, \texttt{0x01}]^T$. We may therefore consider word differences that have the same (fixed) value in every byte, and where the value of this byte is chosen to maximise the probability of passing it through the S-box. Assuming that inputs to the g-function are distributed uniformly at random, the byte differences `0x55`, `0x7A` and `0xA8` will pass through the S-box with probability $6/256$. Hence we may assume that the difference `0xA8...A8` (as well as the other two) is passed through an active g-function with probability $p = (6/256)^8 \approx 2^{-43.32}$.

If we assume that there is some δ that passes through the g-function unchanged with probability p, and that all additions and subtractions behave differentially as xor, we can easily search

[4] The ARIRANG compression function does not use modular additions or subtractions.

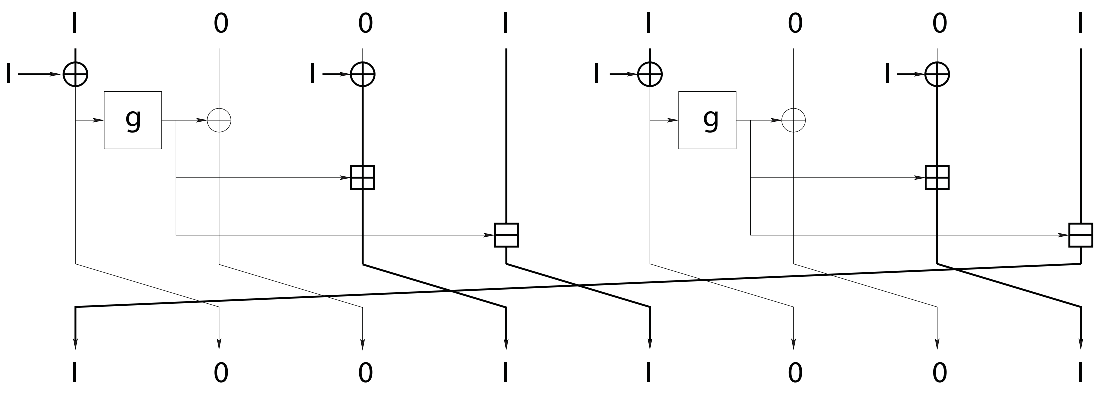

Fig. 2. An iterative characteristic for the Sarmal round function

through all possible characteristics, much in the same way as for ARIRANG in [3]. With a single exception, all characteristics obtained in this way will have a number of active g-functions present. These can be satisfied either through the use of message modification techniques (in the same style as for the attack on ARIRANG, see [3] and Appendix A.1 for details), or probabilistically. While the exact analysis (involving the number of equivalent compression function calls) is more involved, a good working estimate is that one may satisfy $\{2, 3, 4, 6\}$ of the active g-functions probabilistically for Sarmal-$\{224, 256, 384, 512\}$, while keeping the overall cost of the characteristic below generic attacks.

4 Non-random Behavior of the Compression Function

A curious observation about the Sarmal compression function is that it is possible to obtain an iterative differential characteristic for the round function G that holds with probability one.

Consider a single application of G, with state input $X_{j-1} = (X_{j-1}[0], \ldots, X_{j-1}[7])$ and message input $m_j = (A, B, C, D)$. Now suppose that we have input differences $\Delta X_{j-1} = (\delta, 0, 0, \delta, \delta, 0, 0, \delta)$, and $\Delta m_j = (\delta, \delta, \delta, \delta)$. This means that none of the g-functions will be active, since the differences in $X_{j-1}[0]$ and $X_{j-1}[4]$ are cancelled by the differences coming from A and C. If the two additions and the two subtractions also behave as xor, we find that $\Delta X_j = \Delta X_{j-1}$. When $\delta = I \triangleq$ `0x80...00`, this occurs with probability one. This is illustrated in Fig. 2.

By using this characteristic, it is simple to distinguish the full compression function of Sarmal from a pseudorandom function, with only two compression function calls (having suitable differences in the inputs). For the particular mode of operation used in Sarmal, this corresponds to having differences (in the most significant bit) of not only all the message blocks M_i and some of the incoming chaining values h_{i-1}, but also the salt s, and the HAIFA counter value t which denotes the number of bits hashed. This is somewhat unreasonable in practice, and hence we have not been able to apply this result to attack the full hash.

5 Finding the Best Characteristics

We performed an exhaustive search for characteristics leading to collisions and pseudo-collisions, similar to the one in [3]. The round function takes 16 message words and 8 state words as input, so checking all possible inputs with respect to the input differences (0,`0xA8...A8`) requires only 2^{24} trials. Our search heuristic was to look for characteristics with the least number of active g-functions.

The "best" characteristics found in this way did not contain differences in any of the message words, but only in part of the chaining value h_{i-1}. Common for these characteristics is that only one of the two parallel branches becomes active. The single best characteristic found for Sarmal-256 has 9 active g-functions, and is shown in Fig. 3. For Sarmal-224, the best characteristic is also unique, but more expensive (10 active g-functions), due to the way output truncation is performed. In the case of Sarmal-$\{384, 512\}$, there are several different characteristics with the minimal number of active g-functions, 12. Characteristics leading to collisions, as opposed to pseudo-collisions, are significantly more expensive. The results of our search are summarised in Table 1.

Finally, we applied message modification strategies in the style of [3] to all the most promising characteristics. While we (in principle) control more input words (24) than the number of active constraints on the pseudo-collision characteristic, the message schedule makes it difficult to satisfy enough g-functions. Our best result was for Sarmal-512, but even here we would still need to satisfy 7 of the g-functions probabilistically, yielding an attack complexity well above 2^{-256}. We were therefore not able to construct pseudo-collisions for any of the Sarmal digest sizes. A possible strategy for message modification is explained in the appendix.

Table 1. Complexity of the best pseudo-collision and collision characteristics for all digest sizes

Digest size	224	256	384	512
Active g-functions for pseudo-collision	10	9	12	12
Active g-functions for collision	17	17	25	26

Since we in this analysis neglected the cost of passing our differential through the modular arithmetic, the real cost of any attack along these lines is likely to be even higher.

6 Conclusion and Acknowledgements

We studied the impact of the recent ARIRANG attacks [3] for Sarmal, a hash function with a similar design. We find that Sarmal appears less vulnerable to these attacks. Using the techniques developed for ARIRANG, we could not construct pseudo-collisions nor collisions for any digest size.

The Sarmal compression function can, however, be distinguished from a pseudorandom function by using a differential characteristic that holds with probability one. This result does not seem extensible to the full hash function, and hence does not seem to invalidate the security claims made by the designers of Sarmal.

The authors would like to thank Sebastiaan Indesteege, Christophe De Cannière, Vesselin Velichkov as well as Kerim Varıcı for useful discussions. Tor E. Bjørstad would like to thank the COSIC research group at Katholieke Universiteit Leuven for hosting him while this research was done.

References

1. E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions — HAIFA. In *Second NIST Cryptographic Hash Workshop*, 2007.
2. D. Chang, S. Hong, C. Kang, J. Kang, J. Kim, C. Lee, J. Lee, J. Lee, S. Lee, Y. Lee, J. Lim, and J. Sung. ARIRANG. Submitted to the NIST SHA-3 hash function competition, 2008. Available: `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/ARIRANGUpdate.zip`.
3. J. Guo, K. Matusiewicz, L. R. Knudsen, S. Ling, and H. Wang. Practical pseudo-collisions for hash functions ARIRANG-224/384. Available online, 2009.
4. D. Hong, D. Chang, J. Sung, S. Lee, S. Hong, J. Lee, D. Moon, and S. Chee. A New Dedicated 256-Bit Hash Function: FORK-256. In M. J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2006.

5. F. Mendel and M. Schläffer. Collisions and Preimages for Sarmal. Available online, 2008.
6. I. Nikolić. Preimage attack on Sarmal-512. Available online, 2008.
7. N. I. of Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. *Federal Register*, 27(212):62212–62220, November 2007. Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (2008/10/17).
8. K. Varıcı, O. Özen, and Ç. Kocair. Sarmal: SHA-3 Proposal. Submitted to the NIST SHA-3 hash function competition, 2008. Available: `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Sarmal.zip`.
9. D. Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

A Appendix

A.1 A Pseudo-collision Characteristic for Sarmal-256

Using the characteristic in Fig. 3, we construct a strategy for message modification to find a pseudo-collision for Sarmal-256.

– Single-message modification can be used for $M_i[0]$ in round 1.
– We can use single message modification for $M_i[15]$ in round 5 as well. Although this changes $M_i[15]$ in round 4, the change will not propagate through a g-function, and as such still follows the characteristic.
– We change $M_i[12]$ in round 6. This will also change $M_i[12]$ in round 4. By correcting $M_i[7]$ in round 2 as well, the output after round 4 will not be changed.
– Changing $M_i[13]$ in round 7 will also affect $M_i[13]$ in round 4. To keep the output after round 4 the same, we modify $M_i[4]$ in round 2. As this will change the output of the g-function in round 2, we must modify $h_{i-1}[0]$, $h_{i-i}[1]$ and $h_{i-1}[2]$ as well.
– Changing $M_i[3]$ in round 7 will affect $M_i[3]$ in round 1 as well. Correcting this change is not possible, as we cannot change the counter value $c[0]$. As message modification cannot be used, $M_i[3]$ in round 7 will only have the right value with some probability.
– Changing $M_i[14]$ in round 10 also changes $M_i[14]$ in rounds 4 and 5. To correct these changes, we must also change $M_i[5]$ in round 2 and $M_i[8]$ in round 3. As the change of $M_i[8]$ in round 3 is at the input of a g-function, we must change several other words of the message and chaining value in earlier steps as well. We must also correct the changes of $M_i[5]$ and $M_i[8]$ in rounds 8 and 7 respectively. As we do not have the freedom to do this, we can only satisfy $M_i[14]$ in round 10 probabilistically.
– Message modification techniques for $M_i[6]$ in round 11, $M_i[1]$ in round 15 and $M_i[9]$ in round 16 are not possible for similar reasons. They will therefore only have the correct values with some probability.

Using this strategy for message modification, we need to satisfy at least five g-function inputs probabilistically. As was shown in Sect. 3, the attack complexity will never be lower than the generic birthday bound if the number of g-functions that we cannot control is more than three. For other characteristics, our search program showed that the complexity is even worse. Similar results were obtained for the other digest sizes of Sarmal. We were therefore not able to produce a pseudo-collision attack for any digest size of Sarmal, using the techniques of [4].

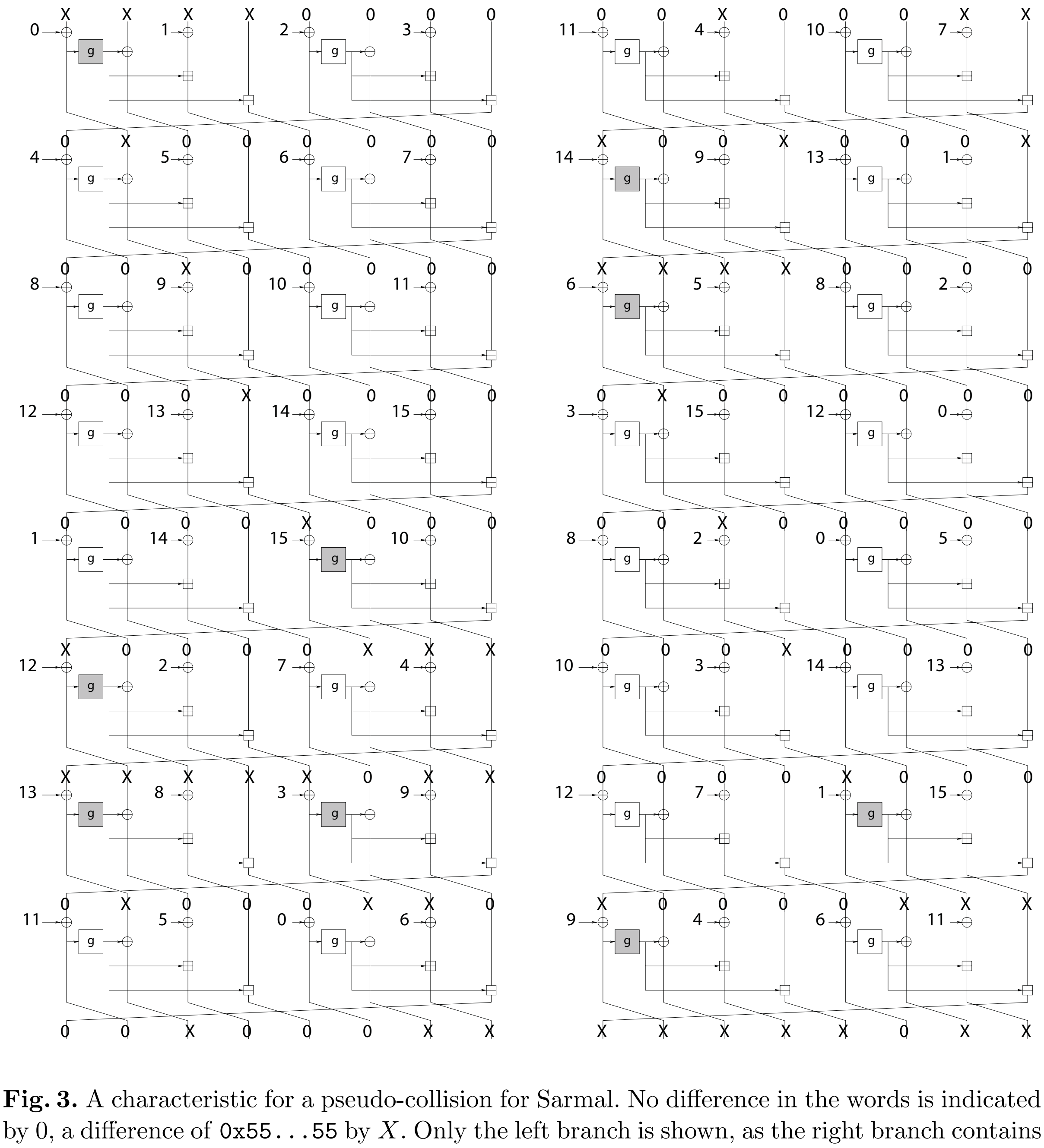

Fig. 3. A characteristic for a pseudo-collision for Sarmal. No difference in the words is indicated by 0, a difference of `0x55...55` by X. Only the left branch is shown, as the right branch contains no differences. No differences are introduced in the message words. A shortened notation is used for the message words that do not include the block index.