A Study on RAM Requirements of Various SHA-3 Candidates on Low-cost 8-bit CPUs

Kota Ideguchi[1], Toru Owada[1], Hirotaka Yoshida[12]

[1] Systems Development Laboratory, Hitachi, Ltd.,
292 Yoshida-cho, Totsuka-ku, Yokohama-shi, Kanagawa-ken, 244-0817 Japan
hirotaka.yoshida.qv@hitachi.com
[2] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium

Abstract. In this paper, we compare the implementation costs of various SHA-3 candidates on low-cost 8-bit CPUs by estimating RAM/ROM requirements of them. As a first step toward this kind of research, in our comparison, we make reasonable estimations of RAM/ROM requirements of them which can be done without implementation.

1 Introduction

A cryptographic hash algorithm is an algorithm that takes input strings of arbitrary length and maps them to short output strings of fixed length.

In November 2007, the US National Institute of Standards and Technology (NIST) opened a public competition to develop a new hash algorithm. The competition is NIST's response to recent advances in the cryptanalysis of hash algorithms. The new hash algorithm will be referred to as SHA-3. In November 2008, the first round of the competition started. NIST received 64 submissions for the competition. After NIST internal reviews of them, 51 were selected for meeting the minimum submission requirements, and accepted as the First Round Candidates. In the first round technical evaluation, NIST performs various tests such as an efficiency analysis on the NIST reference platform (a common PC) and NIST also invites the public to compare results on additional platforms (e.g., 8-bit processors, Digital Signal Processors, dedicated CMOS etc.).

To our knowledge, no work has been demonstrated for comparing SHA-3 candidates performance on 8-bit CPUs So far. In this paper, we compare the implementation costs of various SHA-3 candidates on low-cost 8-bit CPUs by estimating RAM/ROM requirements of them to provide the first impression of their performance on this platform.

As a first step toward this kind of research, in our comparison, we choose SHA-3 candidates which are not colored (meaning that no weakness has been found) in the SHA-3 Zoo website [40], and make reasonable estimations of RAM/ROM requirements of them which can be done without implementation.

Hereafter, we attempt to explain the importance of considering 8-bit CPUs for hash algorithm implementations. Firstly, 8-bit CPUs are really popular,

which is based on a report saying that about 55 % of all CPUs sold in the world are 8-bit microcontrollers and microprocessors and over 4 billion 8-bit controllers were sold in 2006 [43, 44].

In recent years, applications using small portable electronic devices employing low-cost 8-bit CPUs have gained increasing attention from both companies and end users. These devices include low-end smart cards and RFID (Radio frequency identification) tags. They provide users with a convenient storage and processing capability. Smart cards have been considered as one of the most important devises and are used in a wide variety of applications such as electronic commerce and identification.

An important point we would like to make here is that a tradeoff should be made between cryptographic functionality and the cost of the device. The use of asymmetric cryptography (including the coprocessors with it) from which many applications benefit could not be allowed in applications using low-end smart cards and RFID tags. Furthermore, in the RFID security community, although the importance of cryptographic protocols providing confidentiality and integrity are recognized, the cost for implementing cryptographic algorithms is considered as a main issue there. Even some symmetric cryptographic algorithms could be too expensive for some RFID applications.

As for choice of algorithms, block cipher technology appears to be more mature than hash function technology due to the AES development by NIST. RFID tags supporting AES are already available [18]. However, it is pointed out [19] that in RFID security community, it is commonly assumed that hash functions are the better choice than block ciphers from implementation perspective. Therefore, several hash functions protocols for RFID applications have already been proposed [15, 17, 27, 31, 38, 41].

Today some RFID tags providing cryptographic functionality, one of which is component identification enabling a vehicle to check if built-in components like the key are proper. Based on the report in [45] saying that passive RFID tag market is expected to hit $486M in 2013, it is thinkable that in the near future, we will see a wide variety of applications for mobile phones and wireless sensor network, etc.

Considering all this above, we expect that RAM/ROM requirements on low-cost 8-bit CPUs should be considered as an very important factor in comparison of SHA-3 candidates.

Please note that the authors are involved with the submission of a SHA-3 candidate Lesamnta. This is a view from these people.

The outline of this paper is as follows. In Sect. 2, we give the overview of our estimation comparison of various SHA-3 candidates in terms of RAM/ROM. In Sect. 3, we make specific comments on sha-3 candidates. In Sect. 4, we present our results with our classification of these candidates. Our conclusions are given in Sect. 5.

2 Overview of Our Estimation Comparison of RAM/ROM

Here we present the overview of our estimation comparison of various SHA-3 candidates in terms of RAM/ROM. In low-cost 8-bit CPU applications, hash algorithms should require limited resources, memory and computation time. Although there are some applications that need fast hash function computations on an 8-bit CPU such as a smart card used as train tickets, the speed is not critical because what is the main factor in performance is the other applications, not the hash function computations.

The most important constraints are basically the limited RAM and ROM. ROM is used to store the program of the cryptographic algorithm and RAM is used to store intermediate results. In our comparison, we consider area-optimized implementations rather than speed-optimized implementation. Therefore we only consider hash algorithms where the message digest size is 256 bits. We assume that the only places where hash algorithms uses area are RAM and ROM. We here focus on studying RAM for the following two reasons: RAM is typically more expensive than ROM and well-designed hash algorithms typically fit in ROM more easily than in RAM.

As explained in Sect. 1, what we consider here is low-end smart cards and RFID tags where coprocessors could not be allowed and even some symmetric cryptographic algorithms could be too expensive. To illustrate out target CPUs, we present the current 8-bit technology in Table 1.

Table 1. 8-bit Microcontroller

RAM (bytes)	Microchip Technology™	Freescale®	National Semiconductors®	NXP®	Atmel®	Renesas®
- 255	PIC®10 PIC®12 PIC®16	RS08™ HC08™	COP8®	80C51™		
256 - 511	PIC®16 PIC®18	HC08™ HCS08™	COP8®	80C51™		
512 -	PIC®16 PIC®18	HC08™ HCS08™	COP8®	80C51™	megaAVR®	H8®

Hereafter are what we consider and how we do in estimating RAM/ROM requirements of hash algorithms.

2.1 A Treatment of Our Estimation of RAM

1. We estimate memory areas for the most consuming components of algorithms: an area for internal state and an area for feed forward. Hence, we do

not estimate memory area for salt, counter and temporary area unless the area requires considerable amount of memory.
2. In case that a hash algorithm uses a block-cipher-like component, we view this as a component consisting of key schedule and mixing function and then we estimate RAM requirements for each of them.
3. We assume that a memory area for the message block is not in the RAM area which a hash algorithm uses. We assume that this area is in the RAM area for applications executing the hash algorithm.

2.2 A Treatment of Our Estimation of ROM

As mentioned before, we focus on estimating RAM requirements. Our estimation of ROM requirements could be more rough than one could expect.

1. We do not estimate the code size.
2. We estimate the ROM requirement for the lookup tables for IV, constants and other data like S-box.
3. We do not consider using on-the-fly generation of IV, constants and other data. However, if our estimate of ROM requirement exceeds 500 bytes, we estimate ROM size by using on-the-fly generation.
4. We estimate ROM requirement by using on-the-fly generation if our first estimates of ROM size exceeds 500 bytes since we think that ROM requirement of 500 bytes of a hash algorithm may impact in the case of some class of low-cost 8-bit CPUs, which may be illustrated by the fact that several 8-bit CPUs in Table 1 have only 2K bytes of ROM. However, if the generator is complex like using the digit of π or e, we do not estimate by using on-the-fly generation.
5. In case of using 4-bit data, we estimate two 4-bit data are stored in one byte area.

3 Specific Comments on Candidates

We estimate RAM requirements of various SHA-3 candidates. In our estimation, the numbers do not include the requirement for the message block.

We also estimate ROM requirements of these SHA-3 candidates. However, we think that it is more difficult to estimate ROM requirements than RAM requirements because it is very difficult to estimate the code size of a hash algorithm, which should be take into account if one needs to know ROM requirements of a hash algorithm. We do not measure the code size of these SHA-3 candidates here because we did not implement them. Instead we estimate how much ROM they require for the initial vector, the constant, and the lookup table in order to see if there is any concern or not.

After our estimation of RAM requirements of SHA-3 candidates and we compare them with those of SHA-256 and SHA-512. We estimate that the RAM usage of SHA-256 can be 128 bytes and that the RAM usage of SHA-512 can be 256 bytes. Considering RAM requirements of SHA-256 and SHA-512, we make the following classification of SHA-3 candidates:

1. "Small" class contains hash algorithms with RAM requirement smaller than that of SHA-256, more precisely, the RAM requirement is less than 100 bytes.
2. "Middle" class algorithm: hash algorithms with RAM requirement comparative to that of SHA-256, more precisely, the RAM requirement is between 100 bytes to 250 bytes.
3. "Large" class contains hash algorithms with RAM requirement larger than that of SHA-512, more precisely, the RAM requirement is more than 250 bytes.

3.1 ARIRANG

We estimate that RAM requirement of ARIRANG is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 32 bytes of RAM respectively. The RAM usage of ARIRANG can be small (64 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 64, and 296 bytes of ROM respectively.

3.2 BLAKE

We estimate that RAM requirement of BLAKE is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 64 bytes of RAM respectively. The RAM usage of BLAKE can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 64, and 80 bytes of ROM respectively.

3.3 BMW

We estimate that RAM requirement of BMW is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 64 bytes of RAM respectively. The RAM usage of BMW can be small (192 bytes).

We estimate that the initial vector, the constant, and the lookup table require 64, 128, and 0 bytes of ROM respectively.

3.4 CHI

We estimate that RAM requirement of CHI is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 64 and 48 bytes of RAM respectively. The RAM usage of CHI can be small (160 bytes).

We estimate that the initial vector, the constant, and the lookup table require 48, 320, and 0 bytes of ROM respectively.

3.5 CubeHash

We estimate that RAM requirement of CubeHash is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 128 bytes of RAM respectively. The RAM usage of CubeHash can be small (128 bytes).

We estimate that the initial vector, the constant, and the lookup table require 128, 0, and 0 bytes of ROM respectively.

3.6 ECHO

We estimate that RAM requirement of ECHO is larger than that of SHA-512. We estimate that the key schedule and the mixing function require 0 and 256 bytes of RAM respectively. The RAM usage of ECHO can be small (320 bytes).

We estimate that the initial vector, the constant, and the lookup table require 64, 0, and 256 bytes of ROM respectively.

3.7 ESSENCE

We estimate that RAM requirement of ESSENCE is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 32 and 32 bytes of RAM respectively. The RAM usage of ESSENCE can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 0, and 0 bytes of ROM respectively. The size of the lookup table is obtained by using the generator.

3.8 FSB

We estimate that RAM requirement of FSB is larger than that of SHA-512. We estimate that the key schedule and the mixing function require 0 and 128 bytes of RAM respectively. The RAM usage of FSB can be small (389 bytes).

We estimate that the initial vector, the constant, and the lookup table require 0, 271616, and 0 bytes of ROM respectively. Based on this very large ROM requirement, we feel some concern about its implementation on low-cost 8-bit CPUs.

3.9 Fugue

We estimate that RAM requirement of Fugue is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 120 bytes of RAM respectively. The RAM usage of Fugue can be small (120 bytes).

We estimate that the initial vector, the constant, and the lookup table require 120, 0, and 256 bytes of ROM respectively.

3.10 Grøstl

We estimate that RAM requirement of Grøstl is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 64 bytes of RAM respectively. The RAM usage of Grøstl can be small (128 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 0, and 256 bytes of ROM respectively.

3.11 Hamsi

We estimate that RAM requirement of Hamsi is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 64 bytes of RAM respectively. The RAM usage of Hamsi can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 128, and 2056 bytes of ROM respectively. The size of the lookup table is obtained by using the generator.

3.12 JH

We estimate that RAM requirement of JH is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 128 bytes of RAM respectively. The RAM usage of JH can be small (128 bytes).

We estimate that the initial vector, the constant, and the lookup table require 128, 0, and 16 bytes of ROM respectively.

3.13 Keccak

We estimate that RAM requirement of Keccak is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 200 bytes of RAM respectively. The RAM usage of keccak can be small (200 bytes).

We estimate that the initial vector, the constant, and the lookup table require 0, 144, and 0 bytes of ROM respectively.

3.14 LANE

We estimate that RAM requirement of LANE is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 96 bytes of RAM respectively. The RAM usage of LANE can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 3072, and 256 bytes of ROM respectively. The size of the constant is obtained by using the generator.

3.15 Lesamnta

We estimate that RAM requirement of Lesamnta is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 32 and 32 bytes of RAM respectively. The RAM usage of Lesamnta can be small (64 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 0, and 256 bytes of ROM respectively. The size of the constant is obtained by using the generator.

3.16 Luffa

We estimate that RAM requirement of Luffa is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 96 bytes of RAM respectively. The RAM usage of Luffa can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 96, 192, and 8 bytes of ROM respectively.

3.17 MD6

We estimate that RAM requirement of MD6 is larger than that of SHA-512. We estimate that the key schedule and the mixing function require 0 and 712 bytes of RAM respectively. The RAM usage of MD6 can be small (712 bytes).

We estimate that the initial vector, the constant, and the lookup table require 0, 224, and 0 bytes of ROM respectively.

3.18 SANDstorm

We estimate that RAM requirement of SANDstorm is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 64 and 160 bytes of RAM respectively. The RAM usage of SANDstorm can be small (224 bytes). RAM requirement of SADNstorm changes depending on the number of message block. It is 224 bytes when one message block is processed. And it is 672 bytes when more than 1002 message blocks are processed.

We estimate that the initial vector, the constant, and the lookup table require 32, 256, and 256 bytes of ROM respectively. The size of the initial vector is obtained by using the generator.

3.19 Shabal

We estimate that RAM requirement of Shabal is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 0 and 176 bytes of RAM respectively. The RAM usage of Shabal can be small (176 bytes).

We estimate that the initial vector, the constant, and the lookup table require 0, 0, and 0 bytes of ROM respectively.

3.20 SHAvite-3

We estimate that RAM requirement of SHAvite-3 is as comparative as that of SHA-256. We estimate that the key schedule and the mixing function require 64 and 32 bytes of RAM respectively. The RAM usage of SHAvite-3 can be small (128 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 0, and 256 bytes of ROM respectively.

3.21 SIMD

We estimate that RAM requirement of SIMD is larger than that of SHA-512. We estimate that the key schedule and the mixing function require 256 and 64 bytes of RAM respectively. The RAM usage of SIMD can be small (384 bytes).

We estimate that the initial vector, the constant, and the lookup table require 64, 0, and 32 bytes of ROM respectively.

3.22 Skein

We estimate that RAM requirement of Skein is smaller than that of SHA-256. We estimate that the key schedule and the mixing function require 64 and 32 bytes of RAM respectively. The RAM usage of Skein can be small (96 bytes).

We estimate that the initial vector, the constant, and the lookup table require 32, 16, and 0 bytes of ROM respectively.

3.23 SWIFFTX

We estimate that RAM requirement of SWIFFTX is larger than that of SHA-512. We estimate that the key schedule and the mixing function require 0 and 256 bytes of RAM respectively. The RAM usage of SWIFFTX can be small (451 bytes).

We estimate that the initial vector, the constant, and the lookup table require 65, 0, and 12544 bytes of ROM respectively. Based on this very large ROM requirement, we feel some concern about its implementation on low-cost 8-bit CPUs.

4 Our Results

In the previous section, we estimated the RAM/ROM requirements of SHA-3 candidates and compare them with those of SHA-256 and SHA-512.

Using the classification of SHA-3 candidates made in Sect. 3, we present our result in Table 2.

Table 2. RAM Requirements of Various SHA-3 Candidates on Low-cost 8-bit CPUs

	Algorithm	RAM(bytes) (Our estimates)	RAM(bytes) (Submittors estimates)	Our classification
1	ARIRANG[14]	64	448	Small
2	BLAKE[3]	96	274	Small
3	BMW[23]	192	264	Middle
4	CHI[25]	160	198	Middle
5	CubeHash[5, 6]	128	128	Middle
6	ECHO[4]	320	≤ 500	Large
7	ESSENCE[33, 34]	96	< 16000	Small
8	FSB[2]	389	-[*1]	Large
9	Fugue[24]	120	≤ 200	Middle
10	Grøstl[22]	128	< 100	Middle
11	Hamsi[30]	96	-[*1]	Small
12	JH[42]	128	-[*1]	Middle
13	Keccak[7, 8]	200	≤ 512	Middle
14	LANE[28]	96	172	Small
15	Lesamnta[26]	64	68	Small
16	Luffa[11, 12]	96	132	Small
17	MD6[37]	712	≤ 1024	Large
18	SANDstorm[39]	224[*2]	256	Middle
19	Shabal[13]	176	192	Middle
20	SHAvite-3[9]	128	-[*1]	Middle
21	SIMD[32]	384	-[*1]	Large
22	Skein[21]	96	-[*1]	Small
23	SWIFFTX[1]	451	-[*1]	Large
	SHA-256[36]	128	-[*1]	Middle
	SHA-512[36]	256	-[*1]	Large

[*1] RAM estimates are not given in the proposal document.
[*2] The figure is correct only when only one block message is processed.

5 Conclusion

In this paper, we compared the performance of various SHA-3 candidates on low-cost 8-bit CPUs by estimating RAM/ROM requirements. We confirmed that there is some range of RAM requirements of SHA-3 candidates would require. We pointed out that a few algorithms have some concerns in terms of ROM. We believe that our estimation comparison could be useful for a choice of hash algorithms used in (future) security applications using low-cost 8-bit CPUs.

6 Acknowledgments

We would like to thank Yasuko Fukuzawa, Shoichi Hirose, Hidenori Kuwakado, Kazuo Ota, Kazuo Sakiyama, Lei Wang, Dai Watanabe for fruitful discussions. We would like to thank Satoshi Kawanami and Yuji Matsuo who helped us verify

performance figures. This work was partially supported by the National Institute of Information and Communications Technology.

7 Trademarks

- 8-bit PIC® microcontrolloer is a registered trademark and Microchip Technology™ is a trademark of Microchip Technology Inc. in the United States and/or other countries.
- Freescale® is a registered trademark and RS08™ HC08™ and HCS08™ are trademarks of Freescale Semiconductor Inc. in the United States and/or other countries.
- National Semiconductors® and COP8® are registered trademarks of National Semiconductor Corporation in the United States and/or other countries.
- NXP® is a registered trademark and 80C51™ is a trademark of NXP Semiconductors in the United States and/or other countries.
- Atmel® and megaAVR® are registered trademarks of Atmel Corporation in the United States and/or other countries.
- Renesas® and H8® are registered trademarks of Renesas Technology Corp. in the United States and/or other countries.

References

1. Y. Arbitman, G. Dogon, V. Lyubashevsky, D. Micciancio, C. Peikert and A. Rosen, "SWIFFTX: A Proposal for the SHA-3 Standard", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
2. D. Augot, M. Finiasz, P. Gaborit, S. Manuel and N. Sendrier, "SHA-3 proposal: FSB", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
3. J.-P. Aumasson, L. Henzen, W. Meier and R. C.-W. Phan, "SHA-3 proposal BLAKE", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
4. R. Benadjila, O. Billet, H. Gilbert, G. Macario-Rat, T. Peyrin, M. Robshaw and Y. Seurin, "SHA-3 Proposal: ECHO", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
5. D. J. Bernstein, "CubeHash Specification (2.B.1)", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
6. D. J. Bernstein, "CubeHash features (2.B.6)", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
7. G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, "Keccak specifications", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
8. G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, "Keccak sponge function family main document", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html

9. E. Biham and O. Dunkelman, "The SHAvite-3 Hash Function", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
10. J. Borst, Bart Preneel and V. Rijmen, "Cryptography on smart cards", Computer Networks:The International Journal of Computer and Telecommunications Networking, vol. 36, pp, 432-435, 2001.
11. C. D. Canniére, H. Sato and D. Watanabe, "Hash Function Luffa – Specification", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
12. C. D. Canniére, H. Sato and D. Watanabe, "Hash Function Luffa – Supporting Document", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
13. A. Canteaut, B. Chevallier-Mames, A. Gouget, P. Paillier, T. Pornin, E. Bresson, C. Clavier, T. Fuhr, T. Icart, J.-F. Misarsky, M. Naya-Plasencia, J.-R. Reinhard, C. Thuillet and M. Videau, "Shabal, a Submission to NISTs Cryptographic Hash Algorithm Competition", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
14. 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", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
15. E. Y. Choi, S. M. Lee, and D. H. Lee. "Efficient RFID Authentication Protocol for Ubiquitous Computing Environment", In Embedded and Ubiquitous Computing - EUC 2005 Workshops, volume 3823 of LNCS, pages 945954. Springer, December 2005.
16. J. Daemen and V. Rijmen, *The Design of Rijndael: AES -Advanced Encryption Standard.* Springer-Verlag, 2002.
17. T. Dimitriou, "A Lightweight RFID Protocol to protect against Traceability and Cloning attacks", In First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2005), pages 5966, Athens, Greece, September 2005. IEEE Computer Society.
18. M. Feldhofer, S. Dominikus, J. Wolkerstorfer "Strong Authentication for RFID Systems Using the AES Algorithm", Cryptographic Hardware and Embedded Systems - CHES 2004 (2004), pp. 357-370.
19. M. Feldhofer and C. Rechberger, "A Case Against Currently Used Hash Functions in RFID Protocols", On the Move to Meaningful Internet Systems 2006, Lecture Notes in Computer Science, pp. 327-381, Springer, 2006.
20. N. Ferguson, S. Lucks, D. Whiting, M. Bellare, T. Kohno, J. Callas, and J. Walker, "The Skein hash function faimily," 2008. http://www.schneier.com/skein.pdf.
21. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas and J. Walker, "The Skein Hash Function Family", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
22. P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer and S. S. Thomsen, "Grøstl – a SHA-3 candidate", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
23. D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, J. Amundsen and S. F. Mjølsnes, "Cryptographic Hash Function BLUE MIDNIGHT WISH", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html

24. S. Halevi, W. E. Hall and C. S. Jutla, "The Hash Function Fugue", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
25. P. Hawkes and C. McDonald, "Submission to the SHA-3 Competition: The CHI Family of Cryptographic Hash Algorithms", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
26. S. Hirose, H. Kuwakado and H. Yoshida, "SHA-3 Proposal: Lesamnta", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
27. D. Henrici and P. Muller, "Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers", In 2nd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2004 Workshops), pages 149153. IEEE Computer Society, March 2004.
28. S. Indesteege, E. Andreeva, C. D. Canniére, O. Dunkelman, E. Käsper, S. Nikova, B. Preneel and E. Tischhauser "The LANE hash function", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
29. T. Jakobsen and L. R. Knudsen, "The interpolation attack on block ciphers," Fast Software Encryption, FSE '97, Lecture Notes in Computer Science, vol. 1267, pp. 28–40, 1997. http://homes.esat.kuleuven.be/~cosicart/ps/LRK-9700.ps.gz.
30. Ö. Küçük, "The Hash Function Hamsi", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
31. Y. Lee and I. Verbauwhede. "Secure and Low-cost RFID Authentication Protocols," In 2nd IEEE Workshop on Adaptive Wireless Networks (AWiN), 2005.
32. G. Leurent, C. Bouillaguet and P.-A. Fouque, "SIMD Is a Message Digest", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
33. J. W. Martin, "ESSENCE: A Candidate Hashing Algorithm for the NIST Competition", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
34. J. W. Martin, "ESSENCE: A Family of Cryptographic Hashing Algorithms", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
35. National Institute of Standards and Technology, Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family, November 2007. http://csrc.nist.gov/groups/ST/hash/documents/SHA-3FRNoticeNov022007-morereadableversion.pdf
36. National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publication 180-2, August 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.
37. R. L. Rivest, B. Agre, D. V. Bailey, C. Crutchfield, Y. Dodis, K. E. Fleming, A. Khan, J. Krishnamurthy, Y. Lin, L. Reyzin, E. Shen, J. Sukha, D. Sutherland, E. Tromer and Y. L. Yin, "The MD6 hash function – A proposal to NIST for SHA-3", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
38. K. Rhee, J. Kwak, S. Kim, and D. Won. "Challenge-Response Based RFID Authentication Protocol for Distributed Database Environment," In Security in Pervasive Computing, Second International Conference, SPC 2005, volume 3450 of LNCS, pages 7084. Springer, 2005.

39. M. Torgerson, R. Schroeppel, T. Draelos, N. Dautenhahn, S. Malone, A. Walker, M. Collins and H. Orman, "The SANDstorm Hash", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
40. The SHA-3 Zoo - The ECRYPT Hash Function Website, http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo
41. S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels. "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," In 1st Annual Conference on Security in Pervasive Computing, volume 2802 of LNCS, pages 201212, 2003.
42. H. Wu, "The Hash Function JH", Submission to NIST, 2008. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
43. Wikipedia, "Microprocessor", ch. Market statistics, http://en.wikipedia.org/wiki/Microprocessor.
44. http://www.semico.com.
45. http://www.infoworld.com/t/networking/passive-rfid-tag-market-hit-486m-in-2013-102.