

Small Scale Variants of the Secure Hash Standard

Marco Macchetti¹ and Philippe Rivard²

¹ Politecnico di Milano, Milan, Italy

² ALaRI-USI, Lugano, Switzerland

macchett@elet.polimi.it

philippe.rivard@alari.ch

Abstract. In this paper we present effective small scale formulations of the Secure Hash Standard; we focus on the SHA-2 family of algorithms, introducing new compact instances baptized SHA-16, SHA-32, and SHA-64. These may be useful for computing hashes and Message Authentication Codes (MACs) on small platforms where only 8-bit processors are available, such as in the case of Radio Frequency Identification (RFID) devices and embedded systems. To prove the soundness of our scaling approach, we analyze the cryptographic properties of the proposed constructions in terms of adherence to the Strict Avalanche Criterion (SAC) and of robustness to birthday attacks, by also comparing the results with the expected values from random functions. As an additional contribution, we complete the theoretical results for the balance property of random functions, thereby also calculating the expected robustness of the original SHA-2 family versus birthday attacks.

Keywords: hash functions, balance, SAC, small scale, RFID.

1 Introduction

The ever-growing availability of mobile and embedded devices poses new challenges with regards to performance, quality of service and, most of all, security and privacy. Research on ad-hoc networks and especially sensor networks has increased significantly in the last few years, proposing interesting solutions but also introducing new questions. The reader is referred to [1] for a good survey on the topic.

There is a clear need for strong authentication primitives to be introduced in sensor and RFID devices. In fact, the former may be used in large quantities to monitor environmental changes, or dangerous events such as landslides, tsunamis, and earthquakes; in this case an adversarial forgery of messages can lead to false alarms, which eventually may decrease the level of trust in these services and may constitute an indirect, but effective, terroristic act. RFID tags can be conveniently used to monitor goods or can be embedded into personal documents (e.g. passports) to strengthen authentication procedures; it is clear that in this context strong authentication means increased difficulty in forging false documents or in counterfeiting goods.

The flow of information to and from these embedded devices can be secured with known techniques, such as by introducing MACs. These can be built starting from available primitives like block ciphers, e.g. with the CBC-MAC [2], or cryptographic hash functions, e.g. with the HMAC [3] or the UMAC [4] constructions. There are several related proposals for lightweight security layers, among those is TinySec [5]. The designers of TinySec correctly point out [6] that MACs are necessary, and that small sized ones are probably sufficient in the foreseen use-cases, as they make up for a good compromise between security and other desirable properties (e.g. battery duration, latency overhead).

While the panorama of block ciphers is relatively quiet, with the Advanced Encryption Standard (AES) [7] being an efficient and secure solution, the hash functions world has been recently shaken by the publishing of successful attacks against the MD5 [8] and SHA-1 [9] algorithms. It seems today that the SHA-2 family [10] is an appealing replacement, as it benefits from a strengthened structure with regards to SHA-1. It is also probable that the research community will introduce new, open and secure hash functions in the near future.

The SHA-2 family of algorithms was designed specifically for 32-bit and 64-bit processors, and the hash lengths are 256, 384 or 512 bits. These algorithms are too cumbersome to manage on 8-bit embedded platforms. It is always possible to compute a 256-bit MAC and truncate it to the desired length, but this is not an efficient approach.

We show that a small scale design approach can be adopted and conveniently applied to the SHA-2 algorithm; small scale versions of cryptographic algorithms can be useful to test the quality of the design methodologies and also, in our case, to define and validate reasonably-sized hash functions that can be conveniently used in RFID and sensor platforms. A similar approach was used in [11] to test the expected robustness of the AES versus algebraic attacks.

In Section 2, we present our small scale SHA variants, discussing the parts of the original algorithm that scale quite easily and the parts that instead need to be customized; since no details are known about the original design strategy, we make, and explain why we think this is the case, the assumption that the robustness properties are inherited from the older siblings in the family. We examine the possible solutions to the scaling problem, giving guidelines that may also shed some light on the original design rationale (not published by NIST). These small scale variants are named SHA- X , where X is the bit size of the final hash, following the notation used in the original specification.

In Section 3, we tackle the problem of how to validate the robustness of the proposed constructions; we also improve on the current results for the balance metric [12], deriving a simple formula that can be used to estimate the average and minimum expected balance values for an arbitrarily-sized random function. As a side result, we prove that for functions with the input/output size of SHA-256 the difference between random and regular functions vanishes, and quantify the minimum expected effort to produce collisions for the SHA-256 function.

In Section 4, we present the results of our experiments, which validate the quality of our design approach. Section 5 concludes the paper.

2 Small Scale Formulations

The concept of small scale variants of cryptographic algorithms is not new. They have been used on block ciphers as learning tools to understand the functioning of such algorithms, and as tools in the evaluation of techniques which could be used to break them [11]. To our knowledge, no small scale variants of hashing constructs have been built for this purpose.

2.1 The Scaling Strategy

The first useful observation that can be made about SHA-256 is that it seems to scale up quite easily, as NIST has also presented SHA-512, an adaptation of SHA-256 which produces a 512-bit hash [10]. The two algorithms are virtually identical, with the exception that SHA-256 uses 32-bit operands, while SHA-512 uses 64-bit operands. However, it is important to note that the choice of the so-called *sigma* functions is different, and that NIST has not provided any explanation as to their choices in the design of either of these algorithms.

It is clear that the algorithm can also be scaled down, as the primitive operations it bases its security on can also scale easily. Hence, by simply reducing the size of the registers used in the function, most of the algorithm will scale along, as modulo addition, and the choice and majority bitwise non-linear functions can be defined over any operand size. Our small scale variants of SHA-256 can keep the exact same structure with smaller register sizes, as long as we change the sigma functions to reflect the new operand sizes. Also, the constants that are used in the main loop and the initial values of the chaining variable can be adapted by simply truncating the originally specified values to the desired bit sizes.

For very small register sizes, the number of unique operations which can be combined to form the sigma functions is quite limited. In this case, we can simply explore the full design space in order to find the optimal combination of these functions and set these as the sigma functions of our small scale variant. However, as the word size increases, such an approach becomes infeasible. We now turn our attention to the known properties of the sigma functions of SHA-256 and SHA-512, in an effort to minimize this design space exploration.

The constants chosen for the sigma functions in SHA-256 and SHA-512 make each sigma function invertible. Hence, we can first choose our constants in such a way that the resulting function is invertible. Small scale variants of SHA-256 have relatively small operand sizes, and thus, such functions can easily be tested with a brute force approach. It is important to note that there exist more formal methods of testing for invertibility.

It has been reported that the choice of the constants in the sigma functions results in a faster convergence of the hash function towards the strict avalanche criterion [13]. To our knowledge, there exists no formal methodology to obtain a shape for these functions which can guarantee the speedup of this process. However, the adherence of a hash function to the strict avalanche criterion can easily be measured. We can compare the adherence of a potential small scale

variant to the SAC, with that of a subset of the bits of the output of SHA-256. We can also compare this result with the expected value of a purely random function, obtained by using simple discrete mathematics, as can be seen in Section 3.

It is interesting to note that individual subsections of SHA-256 behave in a very similar way in the SAC test to the results expected from a random function. We can thus say that SHA-256 behaves in a pseudo-random manner from this point of view. A second interesting observation that can be made is that by replacing the sigma functions in the full version of SHA-256 with an identity operation, its new outputs begin to diverge from that of a true random function in the SAC test.

Hence, it is clear that close adherence of a potential small scale variant of SHA-256 to the expected results of a random function is a very desirable property. It will be shown that small scale variants also exhibit a divergence from the random case when their sigma functions are replaced by identity operations.

The sigma functions in SHA-256 and in SHA-512 also have some particular properties with respect to their number of fixed points. In the case of SHA-256, σ_0 and σ_1 each have two fixed points, while Σ_0 has two fixed points and Σ_1 has eight. In the case of SHA-512, σ_0 and σ_1 have a unique fixed point corresponding to the null input, while Σ_0 and Σ_1 have two.

2.2 Choice of Parameters

The small scale variants obtained by following the few design tips available in Section 2.1 are denoted as SHA- X , where X is the number of output bits of the hash function, and must be a multiple of 8, as we are primarily decreasing the word length of the operands in the original algorithm. For practical purposes, we will limit ourselves to 3 different word sizes (2, 4, and 8), which will yield small scale variants named SHA-16, SHA-32 and SHA-64. The sigma parameters selected for each of these variants are shown in Table 1, where SH_R^X (RT_R^X) denotes right shift (rotate) by X bit positions.

Table 1. Sigma functions of selected small scale variants

Sig.	SHA-16	SHA-32	SHA-64
Σ_0	$\{x_2, x_1 \oplus x_2\}$	$RT_R^3(x) \oplus SH_R^1(x) \oplus SH_R^2(x)$	$RT_R^1(x) \oplus RT_R^3(x) \oplus RT_R^4(x)$
Σ_1	$\{x_1, x_1 \oplus x_2\}$	$x \oplus RT_R^1(x) \oplus SH_R^3(x)$	$RT_R^2(x) \oplus RT_R^5(x) \oplus RT_R^6(x)$
σ_0	$\{x_1 \oplus x_2, x_2\}$	$x \oplus RT_R^1(x) \oplus SH_R^1(x)$	$RT_R^1(x) \oplus RT_R^6(x) \oplus SH_R^4(x)$
σ_1	$\{x_1 \oplus x_2, x_1\}$	$x \oplus RT_R^1(x) \oplus RT_R^3(x)$	$RT_R^2(x) \oplus RT_R^7(x) \oplus SH_R^5(x)$

The design options for the sigma functions in the case of SHA-16 are quite limited, as the small word size restricts the number of functions which can be considered. Furthermore, it is important to note that the sigma functions of SHA-16 exhibit symmetric behaviour, which could potentially decrease the overall strength of the construction. The small size of the output would make such

a construction highly ineffective for most cryptographic applications, and this formulation is therefore considered to be a purely academic exercise.

However, the number of possible sigma function choices increases drastically as we move to larger word sizes. It is thus possible to attempt to mimic the behaviour of the original sigma functions in the small scale versions with larger operand sizes. The sigma function choices for SHA-32 and SHA-64 reflect this approach.

2.3 A Note About SHA-64

SHA-64 is probably the most interesting construction from a practical point of view; a calculation of a 64-bit hash can be computed very efficiently on 8-bit microcontrollers, such as the widely used ATMEL devices, with an effort roughly 1/4 of that required to run a full SHA-256 computation.

Even if a 64-bit hash does not protect from collision attacks, in some applications the real objective is the calculation of a relatively short MAC. A possible method to obtain a secure 64-bit MAC is as follows. First, a 128-bit key k is chosen (possibly the result of a key distribution undertaken previously); the length of the key is chosen in order to rule out the risk of exhaustive searches onto the key-space. The key is used to replace the initial constants in the SHA-64 algorithm, following the NMAC construction [3]:

$$\text{MAC}_{64} = \text{SHA-64}_{k_l}(\text{SHA-64}_{k_h}(m)) \quad (1)$$

where m is the message and $k = (k_h || k_l)$.

To obtain a fully usable specification, there are some padding issues to be solved with regards to SHA-64. We propose to use the same padding strategy used for the full SHA-256, the only difference being that the last 4 message bytes are reserved for the message length. This allows for messages of length up to 2^{32} bits to be hashed, corresponding to a maximum length of 512 Mbytes. Considering the typical usage on embedded platforms, this should be high enough.

In some applications, e.g. sensor devices, the limit may be safely decreased, by reserving less space for padding (2 bytes allowing the computation of MACs on messages with length up to 8 Kbytes). It is important to note that such customizations always preserve the relative security of the full construction, the parameters being only scaled down to the desired amount.

3 Benchmarking Techniques

Even if there are good reasons to believe that the cryptographic robustness is inherited from the original algorithms, we need to test the cryptographic properties of the proposed small-scale variants of SHA-256. This will confirm our assumptions and possibly shed some light on the role that the different components play in the global construction.

We will not investigate the robustness to differential [14] and linear [15] cryptanalysis techniques. The first reason is that, due to space and time constraints,

we prefer to focus on basic properties, such as the adherence to the SAC, and the collision resistance. A second reason is that, since the structure of the original algorithm is essentially preserved, attacks of this kind that prove successful on small-scale variants are probably also applicable, with some scaling effort, to SHA-256, and vice versa. These results would exceed the goals of this paper.

3.1 Adherence to SAC

The Strict Avalanche Criterion is a basic, but important tool that can be used to test the quality of cryptographic constructions. The approach is to collect statistics for the number of bit-flips in the hash values, when single bits are flipped in the input messages. The results are then compared with what is expected if the function is randomly picked from the set of all functions with the same domain and range sizes; if the gain of the adversary in distinguishing the two cases is small, i.e. if the data series are in good agreement, we can say that the hash function behaves like a good pseudo-random function (PRF), at least with regard to the SAC. This is only a necessary robustness condition for cryptographic functions.

For a randomly-picked function $f : \{0, 1\}^m \Rightarrow \{0, 1\}^n$, and a set of pair-wise distinct input values $D_s = \{d_i, 1 \leq i \leq 2^s\}$, considering the corresponding set of output values $R_s = \{r_i, 1 \leq i \leq 2^s\}$ we have that:

$$\text{Prob}[w(r_i \oplus r_{i+1}) = k] \approx \frac{n^k \sqrt{e^{-k}}}{2^k k!} \quad (2)$$

where $w(x)$ denotes the Hamming weight of x . This is the expected distribution because the whole process is a Poisson experiment, and since the output bits have a 50% probability of being flipped, the expected number of bit flips is $\frac{n}{2}$. The validity of (2) extends to the case when

$$D_s = \{d_i : w(d_i \oplus d_{i+1}) = 1\} \quad (3)$$

thereby offering a comparison ground for the SAC adherence of real functions.

3.2 The Balance Metric

The balance property is, roughly speaking, an estimate of the regularity of a vectorial Boolean function f ; it is a real number lying in the $(0, 1)$ interval, equal to 1 if all pre-image classes of f have the same cardinality, and equal to 0 if f is a constant function. The exact formulation is given in [12], along with the proof that the complexity of birthday attacks on a given hash function is directly linked to its balance value.

It is still an open problem to derive an expected range of balance values for random functions with domain and range sizes comparable to those of SHA-256. In the following, we will obtain a simple expression for the latter, which can be used to test our small-scale variants and also to calculate the collision resistance of functions like the full SHA-256.

A first thing to clear out is that, in general, there is no link between the balance value of a function and its randomness, since it is easy to see that there exist highly balanced functions that are not at all random, and any random function has a finite probability of having any balance value between 0 and 1. Thus the two properties seem to be orthogonal, at least in general.

However, we are justified to say that a function randomly picked from the set of all functions with domain size $|D| = 2^m$ and range size $|R| = 2^n$ is expected to have a definite balance value. This is nothing but the expected value (mean) of the distribution of balance values for all functions in the set. If the variance of this distribution is very small, the claim has a strong basis, and if we have good reasons to believe that a function is behaving randomly, then we can say that its balance is likely to be very near to the mean value of the balance for the set of functions with the same domain and range size.

If f is randomly chosen, we can model its behaviour with the following process. There is a basket which contains 2^n balls, each one indexed by a unique number (for instance in the range $0 \dots 2^n - 1$). The index of each ball corresponds to the output of f . We randomly pick one ball from the basket, we read the index and we increment a corresponding counter; the ball is then put back into the basket. We repeat the extraction 2^m times and in the end we calculate the expected balance from the values of all the counters.

The problem can be reduced to calculating the expected values of the counters. In the above experiment, the chance that a ball with a given index is extracted k times is P_k ; also, $z = m - n \geq 0$ and we may assume that $m, n \geq 8$.

$$\begin{aligned}
P_k &= \binom{2^m}{k} \left(\frac{1}{2^n}\right)^k \left(\frac{2^n - 1}{2^n}\right)^{2^m - k} = & (4) \\
&= \frac{2^m!}{k!(2^m - k)!} \frac{1}{2^{nk}} \left(\frac{2^n - 1}{2^n}\right)^{2^m} \left(\frac{2^n - 1}{2^n}\right)^{-k} = \\
&= \frac{\prod_{i=0}^{k-1} (2^m - i)}{k!} \frac{1}{(2^n - 1)^k} \left(\left(\frac{2^n - 1}{2^n}\right)^{2^n}\right)^{2^z} \simeq \\
&\simeq \frac{2^{mk}}{k!} \frac{1}{2^{nk}} \left(\frac{1}{e}\right)^{2^z} = \frac{(2^z)^k}{k!} \left(\frac{1}{e}\right)^{2^z} & (5)
\end{aligned}$$

As expected, the distribution approaches a Poisson distribution [16], and thus the probabilities depend only on the difference between m and n . The above distribution of probability is true for each counter value. From this, one can calculate the balance with the usual formula, taking into account that the mean value and the variance of distribution (5) are:

$$\mu_p = E[P_k] = 2^z \tag{6}$$

$$\sigma_p^2 = E[(P_k - \mu_p)^2] = 2^z \tag{7}$$

We can write the balance formula considering that every counter value c_i will be distributed according to (6) and (7), and indicating the distance of each counter

value from the mean value with d_i :

$$\begin{aligned}
B &= \log_{2^n} \left(\frac{2^{2m}}{\sum_{i=1}^{2^n} c_i^2} \right) = \\
&= \log_{2^n} \left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p + d_i)^2} \right) = \\
&= \log_{2^n} \left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p^2 + d_i^2 + 2\mu_p d_i)} \right) \tag{8}
\end{aligned}$$

In any realization of the above process, one has to remember that the values of the counters are not independent, since it must hold that $\sum_{i=1}^{2^n} c_i = 2^m$. This implies that the third term in the sum of (8) can be ignored, since:

$$\begin{aligned}
\sum_{i=1}^{2^n} 2\mu_p d_i &= \sum_{i=1}^{2^n} 2\mu_p (c_i - \mu_p) = \\
&= 2\mu_p \sum_{i=1}^{2^n} c_i - 2^{2n} \mu_p^2 = \\
&= 2^{2m-n} - 2^{2n} \mu_p^2 = 0 \tag{9}
\end{aligned}$$

Moreover, if we want to get an expected value of the balance, we can use the expected value of d_i^2 which is exactly σ_p^2 (this is very easy to see). Elaborating on (8) we obtain:

$$\begin{aligned}
B_{exp} &= \log_{2^n} \left(\frac{2^{2m}}{\sum_{i=1}^{2^n} (\mu_p^2 + \sigma_p^2)} \right) = \tag{10} \\
&= \log_{2^n} \left(\frac{2^{2m}}{2^n (2^{2z} + 2^z)} \right) = \\
&= \log_{2^n} \left(\frac{2^{2m}}{2^{2m-n} + 2^m} \right) = \\
&= -\log_{2^n} \left(\frac{1}{2^m} + \frac{1}{2^n} \right) \tag{11}
\end{aligned}$$

which is an elegant and simple formulation of the expected balance of a random function with the given domain and range sizes.

For interesting values of m, n the value of B_{exp} will be very near to 1, which is always an upper bound on the balance value and is reached by regular functions. What is an acceptable tight lower bound for B for a random function? We can easily answer starting from (10) and using the Chebyshev inequality. For each counter value, we know that:

$$\text{Prob}[(\mu_p - c_i)^2 \geq \phi^2] \leq \frac{\sigma_p^2}{\phi^2} \tag{12}$$

$$\text{Prob}[d_i^2 \geq 2^{z+1}] \leq \frac{1}{2} \tag{13}$$

three formulations behave identically to their parent. However, it is important to note that there is a noticeable difference in the behaviour of the formulations with that of their parent (and the random case) when the sigma functions are replaced by identity transformations (NS stands for no sigma).

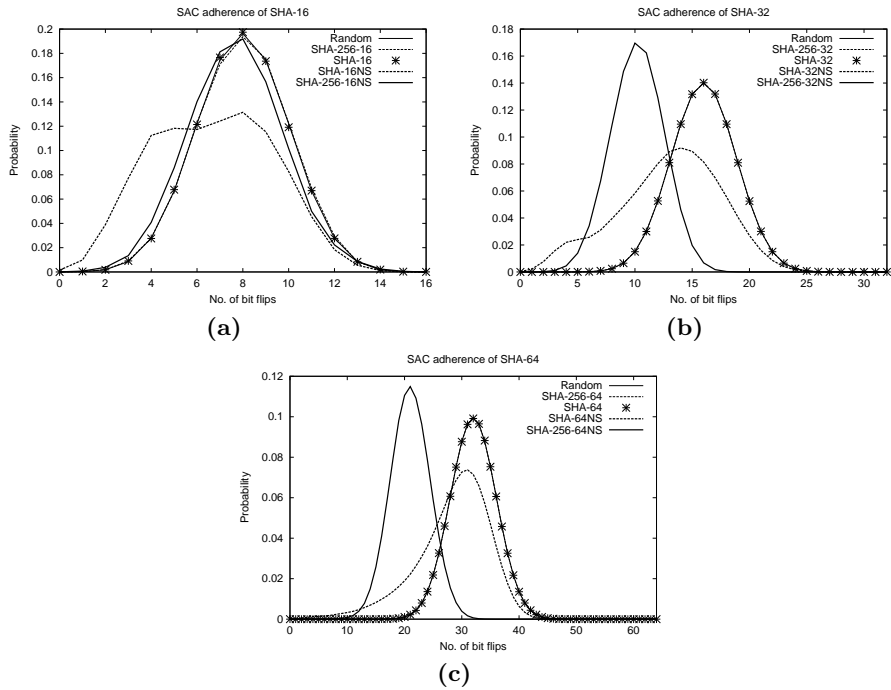


Fig. 1. SAC adherence of SHA-16, SHA-32 and SHA-64

Thus, we can see that careful selection of the sigma functions in each small scale variant is critical. It must be noted that there are many possible design choices resulting in constructions which behave well with respect to this criterion. Since NIST has not made their selection criteria publicly available, and adherence to the SAC alone is not enough to confirm the fitness of the introduced variants, we should consider further fitness tests.

4.2 Balance

The balance of the small scale formulations of SHA-256 was also measured using the method described earlier. The results of this experiment in the case of SHA-16 can be seen in Figure 2. It is interesting to note that also from the balance point of view, SHA-16 behaves exactly like a random function. Furthermore, the version of SHA-16 where all sigma functions are the simple identity

operation does *not* behave randomly. It is clear that any hashing construction which does not meet the minimum bound on balance is not a suitable candidate. The balance of each 16-bit subset of SHA-32 and SHA-64 has also been obtained

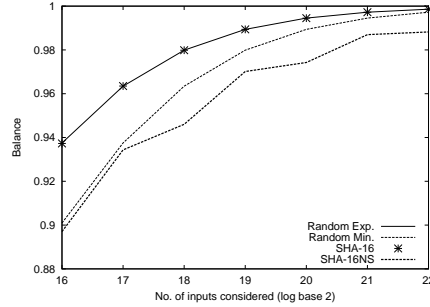


Fig. 2. Balance of SHA-16

experimentally. The size of the segments considered was chosen due to the extensive storage space and execution time required to test larger 8-bit increments thoroughly. These results are shown in Table 2.

Table 2. Balance of 16-bit portions of small scale SHA-256 formulations

Inp.	S-16	S-32 _{0..15}	S-32 _{16..31}	S-64 _{0..15}	S-64 _{16..31}	S-64 _{32..47}	S-64 _{48..63}
2 ¹⁶	0.937288	0.937455	0.937589	0.937324	0.937457	0.937544	0.937816
2 ¹⁷	0.963500	0.963503	0.963366	0.963071	0.963195	0.963497	0.963363
2 ¹⁸	0.979860	0.979892	0.979911	0.979846	0.979667	0.979784	0.980001
2 ¹⁹	0.989392	0.989422	0.989307	0.989413	0.989381	0.989332	0.989437
2 ²⁰	0.994510	0.994542	0.994503	0.994516	0.994498	0.994543	0.994547
2 ²¹	0.997213	0.997236	0.997208	0.997223	0.997232	0.997221	0.997221
2 ²²	0.998599	0.998607	0.998592	0.998607	0.998606	0.998593	0.998600

One can see that all 16-bit subsets of SHA-32 and SHA-64 behave in a very similar manner to SHA-16, which was previously shown to behave like a random function with respect to this test. Hence, we now have small scale formulations which behave identically to SHA-256 under two different tests. Further assessments on the quality of these constructions are beyond the scope of this paper. However, one can see from the results of these experiments that it is possible to obtain seemingly well-behaved small scale versions of SHA-256 by using a careful design methodology.

It is also important to note that all experiments were performed on the full 64 rounds of SHA-256. Considering the impact of round reduction on the small scale variants compared to the full SHA-256 would also be interesting.

5 Conclusions and Future Work

In this paper, we have presented scaled versions of the Secure Hash Algorithm that can be used to implement security features on embedded devices; the most appealing variant, named SHA-64, can be used to efficiently calculate MACs on 8-bit microcontrollers.

We have studied the security of the proposed constructions by considering the adherence to the SAC and the balance metrics, also improving on the known results regarding the latter. In particular we calculate the expected minimum effort needed to find collisions for the full SHA-256 hash function.

Further work includes publishing a full specification and reference code of the small scale variants, analysing the effect of reducing the number of rounds and assessing the quality of the scaling method using different randomness tests.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A Survey on Sensor Networks. *IEEE Communications Magazine*, Vol. 40(8), 102–114, 2002.
2. Bellare, M., Kilian, J., Rogaway, P.: The Security of Cipher Block Chaining. *Proceedings of CRYPTO 1994*, 341–358, 1994.
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. *Proceedings of CRYPTO 1996*, 1–15, 1996.
4. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and Secure Message Authentication. *Proceedings of CRYPTO 1999*, 216–233, 1999.
5. TinySec Webpage, <http://www.cs.berkeley.edu/nks/tinysec/>
6. Karlof, C., Sastry, N., Wagner, D.: TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. *Second ACM Conference on Embedded Networked Sensor Systems (SensSys 2004)*, 2004.
7. Daemen, J., Rijmen, V.: *The Design of Rijndael: AES-The Advanced Encryption Standard*. Springer-Verlag, 2002.
8. Klima, V.: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. *3rd International Conference Security and Protection of Information*, 2005.
9. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA1. To appear in *CRYPTO 2005 proceedings*.
10. NIST: Announcing the Secure Hash Standard. *Federal Information Processing Standards Publication 180-2*, 2002.
11. Cid, C., Murphy, S., Robshaw, M.: Small Scale Variants of the AES. *Proceedings of Fast Software Encryption 2005*.
12. Bellare, M., Kohno, T.: Hash Function Balance and Its Impact on Birthday Attacks. *Proceedings of EUROCRYPT 2004*, 401–418, 2004.
13. Gilbert, H., Handschuh, H.: Security Analysis of SHA-256 and Sisters. *Proceedings of SAC 2003*, 175–193, 2003.
14. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
15. Matsui, M.: Linear Cryptanalysis method for DES cipher. *Proceedings of EUROCRYPT 1993*, 386–397, 1994.
16. Preneel, B.: *Analysis and Design of Cryptographic Hash Functions*. Doctoral Dissertation, Katholieke Universiteit Leuven, 1993.