Comparative Analysis of Energy Usage of Hash Functions in Secured Wireless Sensor Networks

Henry Nunoo-Mensah
Department of Computer Engineering
KNUST, Kumasi - Ghana

Kwame Osei Boateng
Department of Computer Engineering
KNUST, Kumasi - Ghana

James Dzisi Gadze
Department of Electrical and Electronics Engineering
KNUST, Kumasi - Ghana

ABSTRACT
Security in wireless sensor network (WSN) has become an increasing necessity due to the diverse application areas being implemented. Application areas such as military surveillance and environmental monitoring need to be guided against node tampering and node subversion. Works carried out by wireless sensor network researchers pertaining to increasing the security of the network is significant. Node authentication is a suitable technique against node tampering and the introduction of false nodes. A way of authenticating nodes is by using Message Authentication Code (MAC); this is implemented using hash functions. The limited energy available to sensor nodes, have to be considered when selecting a hash function for implementation. In this paper, comparative analysis of some hash functions (MD-5, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512) were carried out. The functions were selected based on their popularity. The analysis was done to identify a short-list of hash functions that can be used when WSN hash related security techniques are being designed. The short-listed functions so identified were further analyzed in terms of their complexities. This was carried out by executing compiled codes and averaging the CPU time spent in executing a common scenario. The result of the analyses showed SHA-224 as the best hash function to be used when designing energy-conscious secured WSNs.

General Terms
Wireless Sensor Networks, Security, Hash Functions

Keywords
Hash Functions, Wireless Sensor Network, Security, Execution time, Complexity, SHA-224, SHA-1, Analysis

1. INTRODUCTION
Wireless Sensor Networks have gained tremendous pace considering the rate at which they are being used for many challenging applications. The ubiquitous and disposable nature of these sensor nodes, make them easier and less costly to be deployed in harsh and inaccessible areas. The deployed nodes are able to measure and transmit data from the field for numerous environmental and strategic reasons. Sensor nodes deployed in sensor networks, are mostly deployed with very little or no supervision. Under the above mentioned circumstances, the nodes are physically made accessible to possible adversaries and are more vulnerable to security breaches. WSNs are subject to security threats at virtually all layers of the communication protocol stack [1].

Cryptography is a potent remedy to a lot of the security issues faced by network communications. Concerns such as Confidentiality, Integrity and Authentication are tackled using cryptography. Security techniques implemented in traditional wireless networks cannot be easily ported to wireless sensor networks due to the energy constraints and other unique characteristics such as the infrastructureless nature of WSNs [2].

Hashes are able to provide authentication of communication entities and also help in checking the integrity of messages traversing the network. Attacks carried out by adversaries with the intent of altering data traversing the network can be mitigated by, using Hash Message Authentication Code (HMAC). In situations not involving confidentiality; the use of symmetric and asymmetric encryption algorithms can be a drain on the energy available to the sensor nodes. When HMACs are to be used, it is prudent that energy efficient hash functions are selected to help increase the lifespan of deployed nodes in the field. Fig. 1 shows the high-level view of hash function whilst fig.2 shows the detailed view of the workings of a typical hash function.

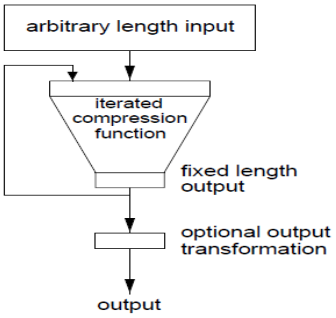

Figure 1: High-level view of an iterative hash function

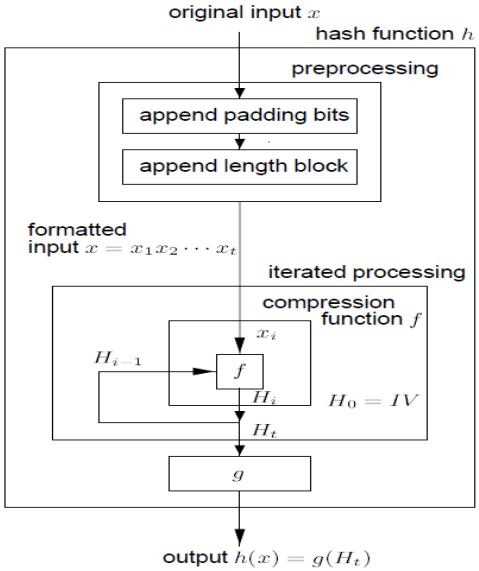

Figure 2: Detailed view of an iterative hash function

Most hash functions (*h)* are designed as iterative processes. These functions hash variable length input data by successively processing input blocks of fixed sizes. A hash input *x* of arbitrary finite length is divided into fixed-length r-bit blocks x_i. The preprocessing stage involves *padding* of the input with extra bits as necessary to attain an overall bit length which is a multiple of the block length *r*. The x_i blocks now serve as inputs to an internal fixed-size hash function *f,* the compression function of *h*, which computes a new intermediate result of bit length *n* for some fixed *n*, as a function of the previous *n*-bit intermediate result and the next input block x_i. H_i denotes the partial result after stage *i*, the general process of an iterated hash function with input $x = x_1x_2x_3...x_i$ can be modeled as:

$H_0 = IV$; $H_i = f(H_{i-1}, x_i)$, $1 \leq i \leq t$; $h(x) = g(H_t)$ [3]

H_{i-1} serves as an *n*-nit *chaining variable* between stage *i-1* and stage *i*. H_0 is a predefined stating value or *initialization value* (IV). There is an optional output transformation g which is used in the final step to map the *n*-bit chaining variable to an *m*-bit result $g(H_t)$; *g* is often the identity mapping $g(H_t) = H_t$. Hash functions are distinguished by the nature of the preprocessing, compression function and the output transformation [3].

This paper conducts a study to determine amongst the MD-5 and SHA family of hash functions, the appropriate hash function for designing secured WSNs that are energy-thrifty. The paper performs analysis on the selected hashes to help future research into security techniques involving hash functions.

This rest of this paper is organized as follows. Section II discusses research works that have been proposed utilizing hashes to provide security in WSNs. Section III describes briefly the selected hash functions and the reason why they have been selected. Sections IV and V outlines the procedure and analysis carried out during the study. Finally section VI concludes this paper by recommending one of the candidate hash functions for possible use by researchers in future works.

2. RELATED WORK

The succeeding paragraphs in this section touch on security related implementations of hashes for wireless sensor network. It is to highlight the fact that hashes are becoming widely used in security schemes. It also shows that, due to their reduced complexity, they constitute the prudent energy efficient option compared to encryption techniques.

Deng et al proposed an intrusion tolerant routing protocol in wireless sensor networks (INENS). It works by adapting a routing-based approach to security in WSNs [4]. The proposed protocol prevents Denial of Service (DoS) attacks by not allowing individual nodes to broadcast to the entire network. The only device allowed to broadcast is the base station which is authenticated using a one-way hash function so as to prevent the possible masquerading of a malicious node. The protocol increases the computation and communication requirements of the base station but not at the field nodes.

Du et al also proposed a one-way hash-function for public key authentication [5]. The proposed schemes is said to be more efficient than signature verification on certificates. The signature verification operation is a very expensive operation for the sensors which also involves a trusted third-party Certificate Authority (CA). The scheme however, requires some hash values to be distributed at a key pre-distribution stage. This phenomenon raises a serious concern against scalability of the network when new nodes are added to the initial network deployment.

Delgado-Mohatar et al proposed an authentication and key establishment scheme that is energy efficient and especially suited to sensor networks [6]. The proposed scheme requires keyed-hash functions (HMAC) and encryption algorithms. The scheme focuses more on confidentiality and authentication. It does not require expensive public key making it light-weight. The scheme is made up of three phases; key pre-distribution phase, network initialization phase and finally authentication protocol. The key pre-distribution requirement hinders the ability of the network to easily scale.

3. HASHING CANDIDATES

The candidate hash functions under consideration in this paper are presented and explained in this section. They are MD-5, SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512. These hash function were selected based on their popularity. Table 1 shows a summary of properties of the candidate hash functions.

3.1 MD-5

Message Digest 5 (MD-5) was developed in 1991 by Ron Rivest. It takes an input of any length and produces a fixed length output digest of 128-bit. The input data that are received by MD-5 is processed in 512-bit block sizes. The blocks are further divided into 16 sub-blocks, each of size 32-bit. The weakness of MD-5 is pointed out in 2004 [7]. The weakness identified by [7] concerning the MD-5 algorithm was that collisions frequently occur on MD-5 hashes. The design of MD-5 with unlimited input message sizes makes it collision-prone.

3.2 SHA-1

Secure Hash Algorithm–1 (SHA-1) [8] was developed by the National Institute of Standards and Technology (NIST) in 1995 as a revision of the original SHA. SHA-1 has a maximum input data length of 64-bit as opposed to the no restriction found in MD-5. The input restriction is said to be one of the strong points of the algorithm since it helps ensure a reduction in the number of collisions due to the input restrictions. SHA-1 produces a fixed output of 160-bit. SHA-1 like MD-5 processes inputs in 512-bit blocks, which are further divided into 16 sub-blocks each of size 32 bits. SHA-1 carries out 80 steps of computation to arrive at the final hash value.

3.3 SHA-224

Secure Hash Algorithm-224 (SHA-224) [9] was announced by NIST in 2004. SHA-224 also has a maximum input data length of 64-bit. A fixed hash value or digest of 224 bits is outputted from the algorithm. SHA-224 also processes inputs in 512-bit blocks which are further divided into sub-block divisions each of a length of 32 bits. SHA-224 performs 64 steps in the computation of the final hash value.

3.4 SHA-256

Secure Hash Algorithm-256 (SHA-256) [9] operates on 512-bit message blocks which are further divided into 32-bit words. It accepts 64-bit input and outputs a digest of fixed length of 256 bits. It also performs a total of 64 steps before the hash values are generated.

3.5 SHA-384

SHA-384 [9] accepts a maximum input of 128 bits and produces a fixed output digest size of 384 bits. It operates on

1024-bit blocks which are further sub-divided into 64-bit word sizes. It goes through a total of 80 steps to produce the final hash value.

3.6 SHA-512

SHA-512 [9] like SHA-384 accepts a maximum of 128-bit input message. It however, produces an output message digest of 512 bits. It operates on inputs in 1024-bit blocks each of which is further sub-divided into 64-bit words.

Table 1. Candidate hash function summarization

	MD-5	SHA -1	SHA -224	SHA -256	SHA -384	SHA -512
Message Digest Size (bit)	128	160	224	256	384	512
Message Size	Varying	$<2^{64}$	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block Size (bit)	512	512	512	512	1024	1024
Word Size (bit)	32	32	32	32	64	64
Number of Steps	64	80	64	64	80	80

4. METHODOLOGY

Analyses carried out in this paper include a combination of consideration from literature and simulation of short-listed hash functions. Initial comparisons were done based on the knowledge from previous works carried out concerning the current state of the hash functions and their suitability for typical sensor networks based on the TinySec security scheme [10]. Next sensor network link layer Protocol Data Unit (PDU) was used to carry out specific analysis. The link layer PDU was chosen because it is the last point before the data accesses the transmission medium and thus it is the best option to focus on. Finally, simulations were carried out on the short-list of hash functions. The hash functions were implemented on a virtual computer using optimized codes of the hash functions. The implementations were fed with the input "wireless sensor networks". The time taken by each hash function to perform the hashing operation was noted. The assumption was that the runtimes are measures of the hash algorithm's time complexity, which affects the compute times of the sensor nodes. Table 2 below shows the parameters of the virtual computer.

Table 2. Parameters of virtual computer running the candidate hash examples

Parameter	Value
Operating System	Ubuntu 12.04 LTS
Hard Disk Space	8GB
Memory	512MB
Compiler	GNU GCC Compiler

The next section details the analysis carried out using existing literature and also displays the CPU execution times of each short-listed candidate.

5. RESULT AND ANALYSIS

MD-5 hash function is ruled out from the list of potential hash functions because even though it is faster in its operation, the unlimited input length makes it very prone to collisions [7]. This is a very critical concern when it comes to services such as authentication.

Shown in fig. 3 is an illustration of the packet format of the TinySec security scheme.

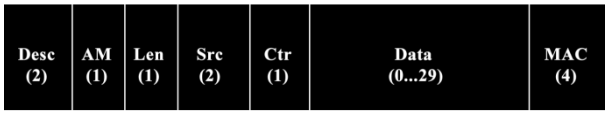

Fig. 3: Packet format for TinySec-AE

In fig. 3 it is clear that the data field has a maximum size of 29 bytes. For authentication purposes of sensor nodes, therefore, a hash function that produces a message digest that is not more than 29 bytes needs to be used. From table 1 digest sizes of 160, 224, 256, 384 and 512 bits are the outputs for SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 respectively. These output sizes translate to 20, 28, 32, 48, 64 bytes respectively. Based on these output data byte values the hash functions, SHA-256, SHA-384 and SHA-512, are also ruled out of the possible candidates since their digest sizes are each beyond the 29-byte maximum acceptable data size per packet. The remaining hash functions which meet the data size criterion are SHA-1 and SHA-224.

When codes of both SHA-1 and SHA-224 were compiled and executed on a virtual computer up to ten times. Figs. 4 - 6 show graphs displaying the execution times recorded. These graphs show that in seven out of the ten cases SHA-224 performed better than SHA-1. The average execution times recorded are shown in table 3.

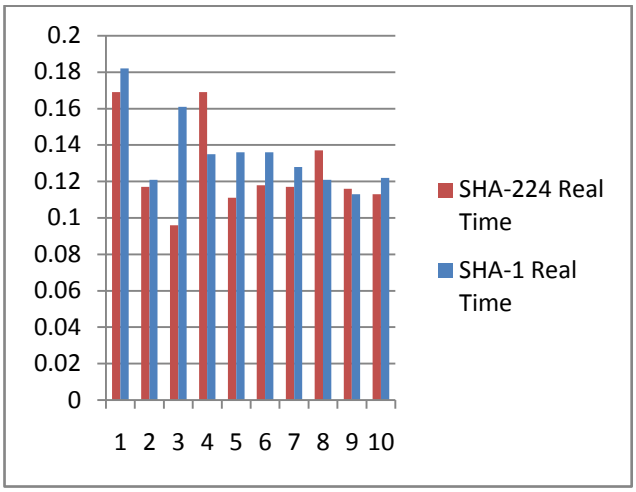

Fig. 4: Execution time (Real-time)

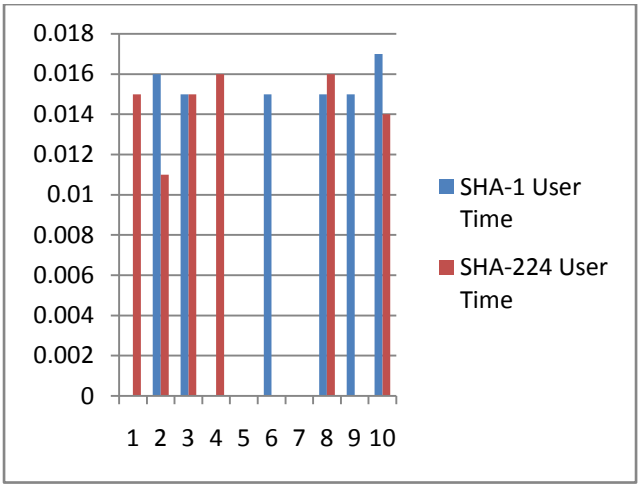

Figure 5: Execution time (User-time)

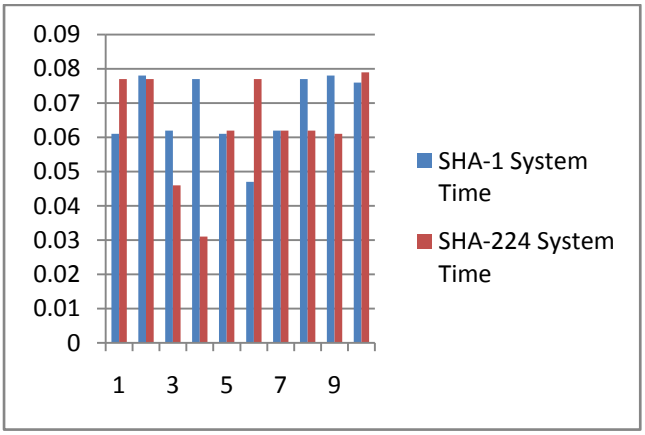

Figure 6: Execution time (System-Time)

Table 3. Average execution time of the compiled hash functions

Hash	Real time	User time	System time
SHA-1	0.1355s	0.0093s	0.0679s
SHA-224	0.1263s	0.0087s	0.0634s

6. CONCLUSION

In this paper, analyses was performed based on findings in literature to compile a shortlist from a set of popular hash functions with the view to determining the most energy-efficient algorithm to recommend for hash-based security schemes for WSNs. Finally, the algorithms in the short list were implemented from their optimized codes and run on a virtual computer and observed their average runtimes as measures of their time complexities. From the final analysis, it is concluded that SHA-224 is the best SHA algorithm for implementing authentication of sensor nodes in a wireless sensor network. This is because SHA-224 met the data field requirement of 29 bytes and also produced a shorter average execution time as compared to SHA-1.

7. REFERENCES

[1] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," IEEE Computer, vol. 35, pp. 53–57, 2002.

[2] X. Du and H. Chen, "Security in Wireless Sensor Networks," IEEE Wireless Communications, pp. 60-66, August 2008.

[3] A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography. CRC Press Inc, 1997.

[4] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-tolerant routing in wireless sensor networks," Technical Report CU-CS-939- 02, Department of Computer Science, University of Colorado at Boulder, November 2002.

[5] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," In Proceedings of the 6th ACM International Symposium on Mobile Ad hoc Networking and Computing, New York, ACM Press, 2005, pp. 58-67.

[6] O. Delgado-Mohatar, A. Fúster-Sabater, J. M. Sierra, "A light-weight authentication scheme for wireless sensor networks," Ad Hoc Networks, Vol. 9, No. 5, Jul. 2011, pp. 727-735

[7] X. Wang, D. Feng, X. Lai and H. Yu, "Collisions for Hash Functions," in Crypto, 2004

[8] A. A. Putri Ratna, P. D. Purnamasari, A. Shaugi and M. Salman, "Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system," Int. Conf. on QiR (Quality in Research), pp.99-104, Jun. 2013

[9] Description of SHA-256, SHA-384, SHA-512, Available at:http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf

[10] C. Karlof, N. Sastry and D. Wagner, "TinySec: A link layer Security Architecture for Wireless Sensor Networks," 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys) '04, pp. 162– 175, Nov. 2004