

An Architecture for a SHA-1 Applied for DSA

S. Pongyupinpanich

Faculty of Electrical Engineering
Naresuan University, Payao Campus, 56000 and
Research Center for Communications and Information
Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang
(KMITL), Bangkok 10520, Thailand
Email: p_surapong2000@yahoo.com

S. Choomchuay

Department of Electronics
Faculty of Engineering, and
Research Center for Communications and Information
Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang
(KMITL), Bangkok 10520, Thailand
Email: kchsomsa@kmitl.ac.th

ABSTRACT

This paper discusses the implementation of the Secure Hash Algorithm (SHA-1) required in the Digital Signature Algorithm (DSA) and many other applications. A single computing module with 80-rounds operation is ideal for a compact size hardware implementation. The round-word calculation module is also simple as a 16x32 bit register with few XOR gates. For the higher throughput those designs can be pipelined to any order, but at the cost of hardware size.

Category and Subject Descriptors

B.7.1 [Integrated Circuit]: Types and Design Styles – Algorithms implemented in hardware

General Terms

Algorithm, Design, Security

Keywords

SHA-1, Digital Signature, Secure Hashing Algorithm, Data Security, IPsec., VLSI

1. INTRODUCTION

In the era of information technology and electronics commerce, data transmission using electronic means becomes more and more important. The National Institute of Standard and Technology (NIST) has issued the Digital Signature Standard (DSS) as its Federal Information Processing Standards Publications, FIP PUB 186-2 [1]. This standard specifies algorithms appropriate for applications requiring a digital, rather than written, signature. An algorithm provides the capability to generate and verify signatures. A private key is used in the signature generation whereas a public key is used in the verification process. Unlike the private key, the public key is assumed to be known in public. As such, anyone can verify the signature but the signature generation can be performed only by the possessor of the user's private key.

Hash functions are common and important cryptographic primitives. Beside their uses in digital signature scheme, they are also basic building block of secret key Message Authentication Codes (MAC), including American federal standard HMAC [2] which is used in security protocols such as SSL and IPsec [3,4]. Other applications of hash functions are

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, May 26-28, 2004, Bangkok, Thailand.
Copyright 2004 ACM x-xxxxx-xxx-x/xx/0004...\$5.00.

password storage and verification, computer virus detection and etc. An n -bit cryptographic hash is an n -bit hash which is one-way and collision-resistant. Current popular hashes produce hash values of length $n = 128$ (MD4 and MD5) [5] and $n = 160$ (SHA-1), and therefore can provide no more than 64 or 80 bits of security, respectively, against collision attacks. The newly NIST approved Advanced Encryption Standard (AES) [6] has offered three cryptovisible sizes, 128, 192, and 256 bits of security. There is a need for companion hash algorithms which provide similar levels of enhanced security. The SHA 256, 384 and 512 has been proposed for such need. They are much detailed in [7]. The comparative analysis of SHA-1 and SHA-512 is given by Grembowski *et al.* [8].

A digital signature algorithm can also be used in proving to a third party that data was actually signed by the generator of the signature. It is intended for use in electronic mail, electronic data interchange, and other applications that require data integrity assurance and data origin authentication. The wireless protocols, such as HiperLAN/2 [9], and WAP [10], have specified security layers and the DSA have been applied for the authentication purposes.

Many applications that include DSA are generally implemented with software approach. Those can be too slow for many modern communications. In the hardware approach, there are also number of SHA-1 encryption design cores available commercially in both ASIC [11] and FPGA [11,12]. The maximum achieved throughput is claimed to about 600 Mbit/s [12]. However in a particular case when the DSA or SHA is intended for small size application and/or dedicate firmware, unreasonable overhead cost can be further reduced by using only necessary and appropriate building blocks. With these regards and according to its implementability and variety of uses (i.e. both generation and verification of signature), we are focusing on SHA-1 in this paper.

The rest of this section will be devoted to a brief revision of DSA. The Secure Hash Algorithm or SHA-1 will be given in details in section 2. The message padding that required for 512 bits data block completion is given in section 3. In section 4 we will describe functions and variable constants. Architectures of a single module operation as well as its corresponding pipeline version are given in section 5. On the fly-round-word computation scheme is elaborated in section 6 before the conclusion.

The signature generation is defined with the following equations:

$$r = \left\langle \left\langle g^k \right\rangle_p \right\rangle_q \quad (1)$$

$$s = \left\langle k^{-1} (M^u + xr) \right\rangle_q \quad (2)$$

where

- a) $g = \left\langle h^{(p-1)/q} \right\rangle_p$, $1 < h < p-1$, and $0 < k < q$ for a randomly selected k ,
- b) $2^{L-1} < p < 2^L$ is a prime for $512 < L < 1024$ and $L = 64m$ for any integer m , (p limited to 1024 in the latest changes),
- c) q is a prime; $2^{159} < q < 2^{160}$,
- d) x is a private key randomly generated, $0 < x < q$; y is a public key, $y = \left\langle g^x \right\rangle_p$,
- e) $M'' = \text{SHA-1}(M)$, for the message M .

The verification process operates on r', s' and M' which are the received versions of r, s and M'' respectively.

$$v = \left\langle \left\langle g^{u1} y^{u2} \right\rangle_p \right\rangle_q \quad (3)$$

If $v = r'$ then the message is verified, otherwise the message may be modified or incorrectly signed. Here; $w = \left\langle s^{-1} \right\rangle_q$, $u1 = \left\langle R w \right\rangle_q$ and $u2 = \left\langle r' w \right\rangle_q$, where $R = \text{SHA-1}(M')$. Be noted that y, g, p and q are transparent to both the signatory and the receiver. The proof for $v = r'$ can be found in [13].

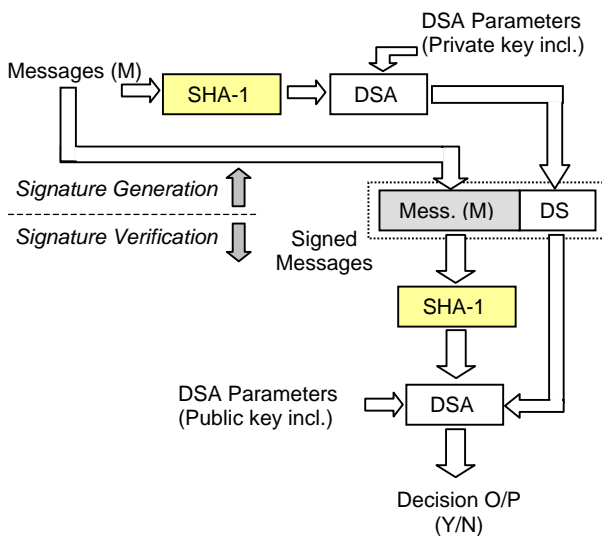


Figure 1: Digital Signature Generation and Verification Process

Although SHA-1 specifies the output size of either 256 or 384 or 512 bits (as SHA-1/256, SHA-1/384 and SHA-1/512 respectively), we are aiming on the most complicate one. SHA-1/512 has the complexity of exhaustive key search of 2^{256} . This figure is similar to that holds by the AES with 256 bit key length. The input messages are thus divided into blocks of 512 bits. Message padding is applied to the last block to complete the 512 bit block size. The SHA-1 operates on those 512 bits data and condense them into a 160 bit data that so called digest. The SHA-1 is called secure because it is computationally

infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. SHA-1 is a technical revision of SHA (FIP 180).

The message digest together with other parameters are computed by the Digital Signature Algorithm (DSA) and the signature is obtained. The mentioned parameters are those required by the signature generation process denoted by (1) and (2) above. p and g are 512 bits data while others (such as private key x) are 160 bits. As discussed above one may notice that SHA-1 is used in both the signature generation and verification process. An efficient SHA-1 computation can have a major impact on the DSA performance.

2. SECURE HASH ALGORITHM (SHA-1)

Illustrated in Fig. 2, SHA-1 requires the operation of 80 rounds which can be grouped into 4 groups, 20 rounds each. Each round operates on five 32 bits hashing words (H_0 to H_4) which have A to E as their temporary versions. Functions and constants are basic operation of each round. Those constants are round constants (K_t) and message word constants (W_t). In summary, SHA-1 is written as:

- a) First block initialization:

$$\begin{aligned} A &= H_0 = 67452301, & B &= H_1 = \text{EFCDAB89} \\ C &= H_2 = 98BACDFE, & D &= H_3 = 10325476 \\ E &= H_4 = \text{C3D2E1F0} \end{aligned}$$

- b) Initialize: $A = H_0, B = H_1, C = H_2, D = H_3$ and $E = H_4$

- c) Perform 80 rounds; for $t = 0; t < 80; t++$;

$$\begin{aligned} T &= W_t + K_t + S^5(A) + E + f(B, C, D) \\ E_{t+1} &= D_t \\ D_{t+1} &= C_t \\ C_{t+1} &= S^{30}(B_t) \\ B_{t+1} &= A_t \\ A_{t+1} &= T \end{aligned} \quad (4)$$

for $t < 16; \dots W_t = W_t$ and for $t \geq 16$

$$W_t = S^1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}). \quad (5)$$

K_t are variable constants given below.

$$\begin{aligned} K_{1t} &= 5A827999, \text{ for } 0 \leq t \leq 19, \\ K_{2t} &= 6ED9EBA1, \text{ for } 20 \leq t \leq 39, \\ K_{3t} &= 8F1BBCDC, \text{ for } 40 \leq t \leq 59, \\ K_{4t} &= CA62C1D6, \text{ for } 60 \leq t \leq 79. \end{aligned}$$

And $f(B, C, D)$ will be detailed in section 4.

- d) Final Adds:

$$H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C$$

$$H_3 = H_3 + D, \text{ and } H_4 = H_4 + E. \quad (6)$$

For (4) – (6) above, + denotes addition with modulation reduction (i.e. mod x^{32} , for any 32 bits word $z = \sum_{j=0}^{31} a_j x^j$,

$a_j \in \{0,1\}$) and S^n denotes n places circular left shift. W_i can be computed for word-divided padded message.

For the next data block, the process loops to step b). To compute the round word (W_i), data is segmented into the block of 512 bits. The last block is padded with “1” and several “0” and the data length to complete 512 bit block size. Detail of message padding is given in section 3 below. After the last block has been processed, message digest (M^n , 160 bits) is obtained.

$$M^n = SHA-1(M) = \{H_0, H_1, H_2, H_3, H_4\} \quad (7)$$

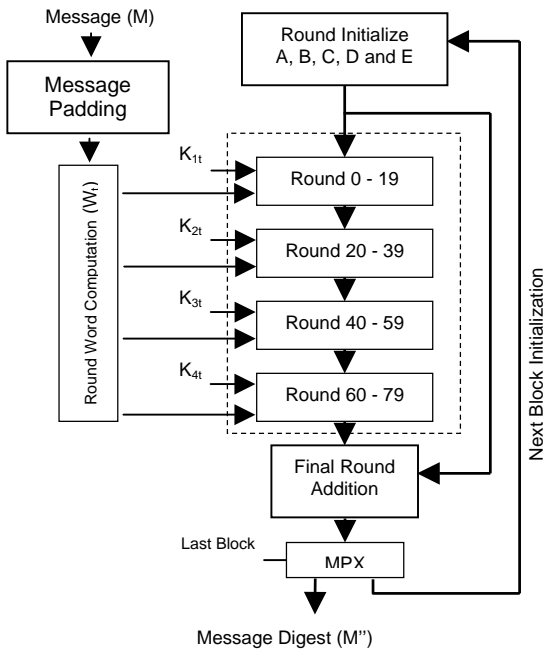


Figure 2: Secure Hash Algorithm (SHA-1)

3. MESSAGE PADDING

The message padding is applied to the last data block such that SHA-1 can process the data of $n \times 512$ bits. The last two words of padded message are reserved of the original message length (in bits). In summary, message padding operates as follows: 1) “1” is padded next to the message data bit (right side), 2) m consecutive “0” are padded to complete the length of 448 bits, 3) last 64 bits are for the original length. For example, message of the last block is given as:

```
01100001 01100010 01100011 01100100
01100101.
```

The message length is 40. After ‘1’ is appended, 407 ‘0’ are required to complete 448 bits. In Hex, this can be written as:

```
61626364 65800000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000.
```

The rest two words are preserved for the original message length. In this case $40 = \text{“00000000 00000028”}$. As a result, the passed message is

```
61626364 65800000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000028.
```

4. FUNCTIONS AND CONSTANTS

Secure Hash Algorithm defines function $f(B,C,D)$ for each round as follows:

$$f(B,C,D) = B \bullet C \wedge \bar{B} \bullet D, \text{ for } 0 \leq t \leq 19,$$

$$f(B,C,D) = B \oplus C \oplus D, \text{ for } 20 \leq t \leq 39,$$

$$f(B,C,D) = B \bullet C \wedge B \bullet D \wedge C \bullet D, \text{ for } 40 \leq t \leq 59,$$

$$f(B,C,D) = B \oplus C \oplus D, \text{ for } 60 \leq t \leq 79.$$

here \bullet denotes logical AND, \wedge denotes logical OR, and \oplus denotes logical XOR. Variable constants used in every round are K_t and W_t given above in section 3.

5. ARCHITECTURES FOR SHA-1

Two main functions of SHA-1 are round function and round word computation. The implementation of SHA-1 in both iterative mode and pipeline modes are given in this section while the corresponding round-word computation is detailed in section 6.

An architecture of the SHA-1 round operation detailed in section 2 is shown in Fig. 3 below. $E + W_i + K_t$ is computed in parallel with $f(B,C,D)$ and $S^5(A)$. The circular n -place shift operation is fast and easy with data line hard-wiring. We applied the same technique to $S^{30}(B)$ operation. This shifting operation operates in parallel with the second summation (BCLA_2).

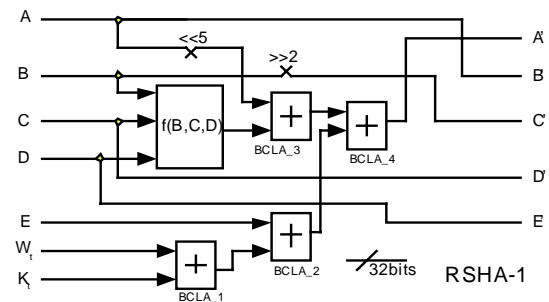


Figure 3: Round Operation of the SHA-1 (RSHA-1)

For the round function $f(B,C,D)$ given in section 4 above, logic block comprises of AND gates, XOR gate, Inverters and round selection multiplexers. Each data path is 32 bit wide. The implementation is given in Fig. 4 below.

In the implementation of a modulo reduction adder required in the implementation of round function given in Fig. 3, we have investigated such SHA-1 with three type of 32 bit parallel adders. These are Carry Propagate Adder (CPA), Carry Look Ahead Adder (CLA), and Block Carry Look Ahead Adder (BCLA) with different sub-block sizes of 4, 8 and 16 bits CLA. Performances are shown in Table 1 below.

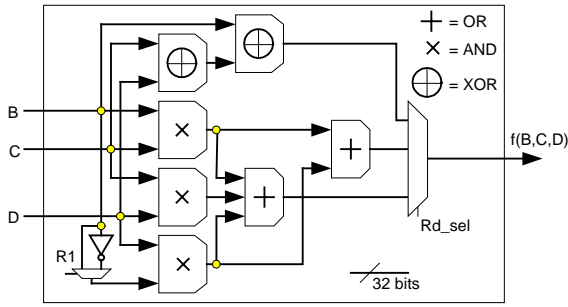


Figure 4: Round Function $f(B,C,D)$

Table 1: Performance Evaluation of SHA-1 with Different Type of Adders (HDL: Xilinx 2V1000FF896)

	CLA	BCLA			CPA
		16 b	8 b	4 b	
Function Generators	2092	1161	952	580	472
CLB Slices	1046	581	476	358	236
Max. Path Delays (ns)	26.9	30.5	24.6	23.8	46.6

Considering the above table one can notice that the Block Look Ahead Adder with 4 bit sub-block offers very good performance (in term of resource utilization and delay times) compared to others. We thus will use this type of adder in the followed implementations. For each 512 bit data block operation, the scheme given in Fig. 3 above is used iteratively for 80 rounds. For minimize hardware requirement only a single block can be designed. As shown in Fig. 5, the thick data paths are 160-bit wide. K_t are stored in four 32-bit registers, each is for 20 rounds.

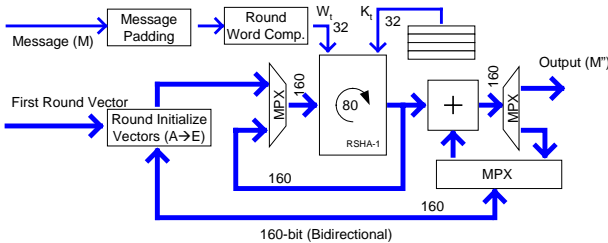


Figure 5: An Iterative Use of a Single RSHA-1 Module

To perform the pipeline computation, the single iterative block (RSHA-1) shown in Fig. 3 above can be cascaded. The throughput can be made higher upon the number of pipeline stage Q . However, the hardware size is increased with proportion to Q . The throughput (TP) can be easily computed from

$$TP = \frac{160}{80/Q} \text{ bits / clock cycle}$$

An illustrated example of the pipeline SHA-1 computation is shown in Fig. 6. Q is set to 4, such that the arrangement is similar to that given in Fig. 2. The throughput of 160 bits/20 clock cycles with the latency of 80 clock cycles could be obtained. With this calculation, the 80 stage fully-pipelined scheme can offer the throughput of 160 bits per clock cycle.

Although the throughput can be boosted by internal pipelining of the RSHA-1 module, what one has to pay for is the increasing of gate utilization. Upon our experiment, per one round computation cell, CLB slice increased by the factor of 1.5. The implementation of round-word calculation is fairly easy when $Q \leq 5$. For the new class of FPGA, the LUT can be employed for better results. However for the higher number of pipeline stage (i.e. $Q > 5$), the design will cumbersome and the gate count increases dramatically.

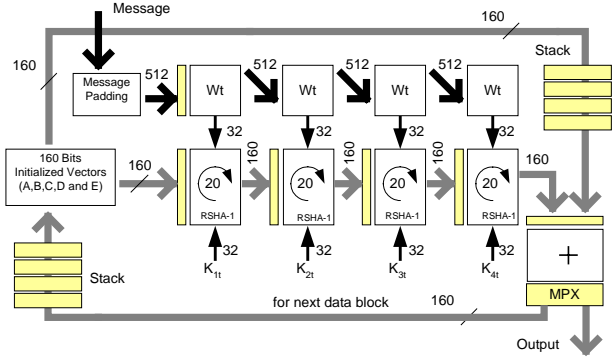


Figure 6: Pipeline Implementation of SHA-1 ($Q=4$)

6. ON THE FLY ROUND-WORD COMPUTATION

Based on equation (5) given in section 2, for $t \geq 16$ the round-word W_t can be computed from a single left-shifted version of the XOR sum of the prior 4 words; i.e. $W_{t-16}, W_{t-14}, W_{t-8},$ and W_{t-3} . We thus need 16 of 8 bits location to store 16 words and use 4 of them. The computed round-word is looped back to store in the previously used location. The same process repeats for other 3 loop sets.

The architecture that supports the above discussed data flow can be designed straight forwardly. As shown in Fig. 7, a 16×32 bit register, 2 multiplexers and 3 XORs are required. MPX_1 passes 16 first round directly to the SHA-1 computing blocks shown in Fig. 5. Data are shifted to the right every clock cycle. Data are cycled to complete 80 round when the feedback path is enable via MPX_2 (the 512 bits message is completed in the first 16 clock cycles). With this arrangement W_t is obtained in very clock cycle.

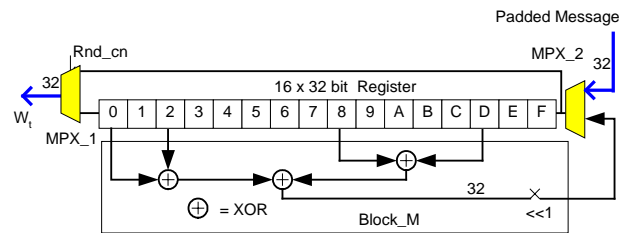


Figure 7: Round-Word Computation Arrangement

When the SHA-1 computation is arranged in the pipeline fashion as shown in Fig. 6. The round-word must be arranged in slight different manner. This is much depended on the number of pipeline stages. To visualize this, let's take an example of the 4-stage pipelining. Each SHA-1 must take care of 20 round processing. As such we need one round-word computation circuit for each pipeline stage (4 in this case). The first set (stage_0) is organized slightly different from others since it has to take care of the first 16 rounds. Data transferring

from a pipeline stage to another pipeline stage can be observed as shown in table 2.

Table 2: Data located in the contiguous set of registers allocated for the pipeline stages.

		Register & Contents																
RND		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0-15		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
16		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	X	
17		2	3	4	5	6	7	8	9	10	11	12	13	14	15	X	Y	
18		3	4	5	6	7	8	9	10	11	12	13	14	15	X	Y	Z	
19		4	5	6	7	8	9	10	11	12	13	14	15	X	Y	Z	W	Stg_0
20		5	6	7	8	9	10	11	12	13	14	15	X	Y	Z	W	R	Stg_1
21		6	7	8	9	10	11	12	13	14	15	X	Y	Z	W	R	T	
22		7	8	9	10	11	12	13	14	15	X	Y	Z	W	R	T	U	

Data transfer arrangement is fairly simple as shown in Fig. 6. However, the below scheme is recommended for the pipeline that the number of round per stage is equal or greater than 16. The wiring complexity increases if the number of pipeline stage is greater than 5.

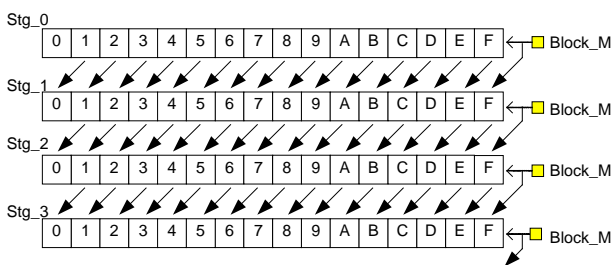


Figure 8: Data Transfer Arrangement for the Round-Word Computation for Pipeline Stages

The designed architecture of the round-word computation has been verified using high language programming and tested by VHDL. Results are given in Table 4 below.

Table 4: Performance Evaluation (Xilinx 2V10000fg896)

	Single Circuit (Iterative, Fig. 7)	4-Copy Circuit (Pipeline, Fig. 8)
Func. Generator	64	128
CLB Slices	128	736
DFF	544	1472

7. CONCLUSIONS

The importance of E-commerce, electronic data exchange, software distribution and wireless communication has resulted in the necessity of data integrity of which DSA is playing the main role. The SHA-1 architecture addressed in this paper is to speed up the DSA process by mean of hardware realization. The design of the SHA-1 computing module is fairly simple. For a compact size hardware such a module must work iteratively for 80 rounds. A round-word calculation module size is also small and works well with the SHA-1 main module. The pipeline architecture that may be needed for the higher throughput application has also been investigated. Despite the high throughput, such an architecture demands more gates.

8. REFERENCES

- [1] National Institute of Standards and Technology, Digital Signature Standard (DSS), FIPS PUB 186-2 (+ Change Notices), January 2000; <http://csrc.nist.gov/publications/>
- [2] FIPS 198, HMAC KeyedHash Message Authentication Code, available at <http://csrc.nist.gov/encryption/tkdisigs.html>
- [3] IP Security Protocol (IPSec) Charter Latest RFCs and Internet Drafts for IPSec, available at <http://ietf.org/html.charters/ipseccharter.html>
- [4] Stallings, W.: Cryptography and Network Security, 1999 Prentice Hall Inc., Upper Saddle River, New Jersey, 2nd Edition.
- [5] The MD5 Message-Digest Algorithm, available at <http://www.faqs.org/rfcs/rfc1321.html>
- [6] NIST, Advanced Encryption Standard (AES), (FIP PUB 197), November 26, 2001, <http://csrc.nist.gov/publications/>
- [7] National Institute of Standards and Technology, Secure Hash Standard, FIPS PUB 180-2, <http://csrc.nist.gov/publications/>.
- [8] T. Grembowski, R. Lien, K. Gaj, N. Nguyen, P. Bellows, J. Flidr, T. Lehman, B. Schott, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," *Proc. Information Security Conference*, Sao Paulo, Brazil, September 30-October 2, 2002.
- [9] Martin Johnson, "HiperLAN/2 – The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band," HiperLAN/2 Global Forum, 1999, Version 1.0 white-paper; <http://www.hiperlan2.com/technology.asp>
- [10] WAP Forum: "Wireless Application Protocol Architecture Specification" and "WAP White Paper", <http://www.wapforum.org>.
- [11] OL_SHA, SHA-1 Processor – High performance Encryption Core, available at; http://www.ocean-logic.com/pub/OL_SHA.pdf
- [12] SHA-1, Secure Hash Algorithm, Cryptoprocessor Core, available at; <http://www.cast-inc.com/cores/sha-1/index.shtml>.
- [13] National Institute of Standards and Technology, Secure Hash Standard (SHS), FIPS PUB 180-1, 1995; <http://csrc.nist.gov/publications/>