SHACAL

Helena Handschuh, David Naccache
Gemplus, 34 rue Guynemer
F-92447 Issy-les-Moulineaux
Contact: Helena.Handschuh@gemplus.com

Submission Statement

This submission presents the block cipher SHACAL in its two versions SHACAL-1 and SHACAL-2, as a submission to NESSIE. It is based on the hash standard SHA used in encryption mode. We believe the main strength of this block cipher family is its inheritance from the extensive analysis that has been made on the hash function itself. We state that no hidden weakness has been inserted in this block cipher, and we believe the design principles to be sound. To the best of our knowledge, SHACAL is not covered by any patents. We do not intend to apply for any patent covering SHACAL and undertake to update the NESSIE project whenever necessary.

The estimated computational efficiency of SHACAL-1 is 2480 cycles per 20 byte block encryption, 2320 cycles per 20 byte block decryption and 2280 cycles per 64 byte key setup. Timing measurements are given for a PC using a pentium III processor running at 800 Mhz. 20 million SHACAL-1 encryptions take about 62 seconds and 20 million decryptions take about 58 seconds. 20 million key setups take 57 seconds.

The estimated computational efficiency of SHACAL-2 is 3600 cycles per 32 byte block encryption, 3680 cycles per 32 byte block decryption and 2800 cycles per 64 byte key setup. Timing measurements are given for a PC using a pentium III processor running at 800 Mhz. 20 million SHACAL-2 encryptions take about 90 seconds and 20 million decryptions take about 92 seconds. 20 million key setups take 70 seconds.

The following report gives an overview of the tweak on submitted algorithm SHACAL. We define SHACAL as a block cipher family with variable key and block length. In its basic version, SHACAL-1 has a 160 bit block and in its extended version, SHACAL-2 has a 256 bit block. Both versions can accomodate a key of up to 512 bits, with a minimum of 128 bits.

The report on the security of SHACAL-1 has been omitted from this tweak submission, but remains available in the original submission package of SHACAL [4] .

Contents

SHACAL : a family of block ciphers

1 Introduction

In the following we give a brief introduction to the Secure Hash Algorithm (SHA).

Many of the popular hash functions today are based on MD4 [6]. MD4 was built for fast software implementations on 32-bit machines and has an output of 128 bits. Because of Dobbertin's work [3, 2] it is no longer recommended to use MD4 for secure hashing, as collisions have been found in about 2^{20} compression function computations.

In 1991 MD5 was introduced as a strengthened version of MD4. Other variants include RIPEMD-128, and RIPEMD-160. SHA was published as a FIPS standard in 1993. All these hash functions are based on the design principles of MD4. RIPEMD-128 produces hash values of 128 bits, RIPEMD-160 and SHA-1 produces hash values of 160 bits.

SHA was introduced by the American National Institute for Standards and Technology in 1993, and is known as SHA-0. In 1995 a minor change to SHA-0 was made, this variant known as SHA-1. The standard now includes only SHA-1 [8]. In 2000, NIST published a new algorithm commonly known as SHA-2 ; 3 new hash functions are described respectively generating a digest of 256, 384 and 512 bits. We have selected 2 primitives among these five for our SHACAL submission. These are SHA-1 and the 256 bit variant of SHA-2. We define SHACAL as a variable block and key length family of block ciphers. We describe and submit only two variants to NESSIE, but it is straightforward to derive SHACAL for 384 and 512 bit blocks from the description of the two other hash function variants described in the SHA-2 standard [9]. Descriptions of both algorithms follow in the next sections.

Notation:

- $+$. Addition modulo 2^{32} of 32 bit words.
- $ROTL_i(W)$. Rotate 32 bit word W to the left by i bit positions.
- $S_i(W)$. Rotate 32 bit word W to the right by i bit positions.

- $R_i(W)$. Shift 32 bit word W to the right by i bit positions.
- $\oplus$. Bitwise exclusive-or.
- &. Bitwise and.
- |. Bitwise or.

2 SHACAL-1

To hash a message with SHA-1 proceed as follows.

1. Pad the message, such that the length is a multiple which fits the size of the compression function, see [5].
2. Initialize the chaining variables AA, BB, CC, DD, EE, each of 32 bits, by
   (a) $AA = IV_1 = \texttt{67452301}_x$,
   (b) $BB = IV_2 = \texttt{EFCDAB89}_x$,
   (c) $CC = IV_3 = \texttt{98BADCFE}_x$,
   (d) $DD = IV_4 = \texttt{10325476}_x$,
   (e) $EE = IV_5 = \texttt{C3D2E1F0}_x$.
3. For each message block of 512 bits:
   (a) Set $A = AA, B = BB, C = CC, D = DD, E = EE$.
   (b) Expand the 512 bits to 2560 bits, cf. later.
   (c) Compress the 2560 bits in a total of 80 steps; each step updates in turn one of the working variables A, B, C, D, and E, see section on compression function.
   (d) Set $AA = AA + A, BB = BB + B, CC = CC + C, DD = DD + D$ and $EE = EE + E$.
4. Output the hash value $[AA \,||\, BB \,||\, CC \,||\, DD \,||\, EE]$.

2.1 Compression function

Let the message blocks of 512 bits be denoted $M = [W^0 \,||\, W^1 \,||\, \ldots \,||\, W^{15}]$, where W_i are 32-bit words. For SHA-0 the expansion of 512 bits to 2560 bits is defined

$$W^i = W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16}, \quad 16 \leq i \leq 79. \tag{1}$$

In SHA-1 the expansion is defined

$$W^i = ROTL_1(W^{i-3} \oplus W^{i-8} \oplus W^{i-14} \oplus W^{i-16}), \quad 16 \leq i \leq 79. \tag{2}$$

These expansions are the only difference between SHA-0 and SHA-1.

Define the following functions.

$$\begin{aligned} f_{if}(X,Y,Z) &= (X\&Y)|(\neg X\&Z) && (3)\\ f_{xor}(X,Y,Z) &= (X \oplus Y \oplus Z) && (4)\\ f_{maj}(X,Y,Z) &= ((X\&Y)|(X\&Z)|(Y\&Z) && (5) \end{aligned}$$

The above 80 steps are then defined

$$\begin{aligned} A^{i+1} &= W^i + ROTL_5(A^i) + f^i(B^i,C^i,D^i) + E^i + K^i && (6)\\ B^{i+1} &= A^i && (7)\\ C^{i+1} &= ROTL_{30}(B^i) && (8)\\ D^{i+1} &= C^i && (9)\\ E^{i+1} &= D^i && (10) \end{aligned}$$

for $i = 0\ldots,79$, where

$$\begin{aligned} f^i &= f_{if}, \quad 0 \le i \le 19\\ f^i &= f_{xor}, \quad 20 \le i \le 39, 60 \le i \le 79\\ f^i &= f_{maj}, \quad 40 \le i \le 59. \end{aligned}$$

The constants K^i are defined

$$\begin{aligned} K^i &= \texttt{5A827999}_x, \quad 0 \le i \le 19\\ K^i &= \texttt{6ED9EBA1}_x, \quad 20 \le i \le 39\\ K^i &= \texttt{8F1BBCDC}_x, \quad 40 \le i \le 59\\ K^i &= \texttt{CA62C1D6}_x, \quad 60 \le i \le 79 \end{aligned}$$

The output after 80 steps, $A^{80}, B^{80}, C^{80}, D^{80}, E^{80}$ is then used to update the chaining variables AA, BB, CC, DD, EE.

The best attack known on SHA-0 when used as a hash function is by Chabaud and Joux [1]. They show that in about 2^{61} evaluations of the compression function it is possible to find two messages hashing to the same value. A brute-force attack exploiting the birthday paradox would require about 2^{80} evaluations. There are no attacks reported on SHA-1 in the open literature.

3 SHACAL-2

To hash a message with SHA-2 with 256-bit hash result proceed as follows.

1. Pad the message, such that the length is a multiple which fits the size of the compression function, see [5].
2. Initialize the chaining variables $AA, BB, CC, DD, EE, FF, GG, HH$, each of 32 bits, by
   (a) $AA = IV_1 = \texttt{6A09E667}_x$,

(b) $BB = IV_2 = \texttt{BB67AE85}_x$,

(c) $CC = IV_3 = \texttt{3C6EF372}_x$,

(d) $DD = IV_4 = \texttt{A54FF53A}_x$,

(e) $EE = IV_5 = \texttt{510E527F}_x$.

(f) $FF = IV_6 = \texttt{9B05688C}_x$,

(g) $GG = IV_7 = \texttt{1F83D9AB}_x$,

(h) $HH = IV_8 = \texttt{5BE0CD19}_x$.

3. For each message block of 512 bits:

   (a) Set $A = AA, B = BB, C = CC, D = DD, E = EE, F = FF, G = GG, H = HH$

   (b) Expand the 512 bits to 2048 bits, cf. later.

   (c) Compress the 2048 bits in a total of 64 steps; each step updates in turn two of the working variables A, B, C, D, E, F, G and H, see section on compression function.

   (d) Set $AA = AA + A, BB = BB + B, CC = CC + C, DD = DD + D, EE = EE + E, FF = FF + F, GG = GG + G$ and $HH = HH + H$.

4. Output the hash value $[AA \,||\, BB \,||\, CC \,||\, DD \,||\, EE \,||\, FF \,||\, GG \,||\, HH]$.

3.1 Compression function

Let the message blocks of 512 bits be denoted $M = [W^0 \,||\, W^1 \,||\, \ldots \,||\, W^{15}]$, where W_i are 32-bit words. In SHA-2 with 256-bit hash values the expansion is defined by

$$W^i = \sigma_1(W^{i-2}) + W^{i-7} + \sigma_0(W^{i-15}) + W^{i-16}, \quad 16 \leq i \leq 63. \tag{11}$$

where σ_0 and σ_1 are defined as follows:

$$\sigma_0(x) = S_7(x) \oplus S_{18}(x) \oplus R_3(x) \tag{12}$$
$$\sigma_1(x) = S_{17}(x) \oplus S_{19}(x) \oplus R_{10}(x) \tag{13}$$

Define the following functions:

$$Ch(X, Y, Z) = (X \& Y) \oplus (\neg X \& Z) \tag{14}$$
$$Maj(X, Y, Z) = (X \& Y) \oplus (X \& Z) \oplus (Y \& Z) \tag{15}$$
$$\Sigma_0(X) = S_2(X) \oplus S_{13}(X) \oplus S_{22}(x) \tag{16}$$
$$\Sigma_1(X) = S_6(X) \oplus S_{11}(X) \oplus S_{25}(x) \tag{17}$$

The above 64 steps are then defined

$$
\begin{aligned}
T_1 &= H + \Sigma_1(E) + Ch(E,F,G) + K^i + W^i && (18)\\
T_2 &= \Sigma_0(A) + Maj(A,B,C) && (19)\\
H^{i+1} &= G^i && (20)\\
G^{i+1} &= F^i && (21)\\
F^{i+1} &= E^i && (22)\\
E^{i+1} &= D^i + T_1 && (23)\\
D^{i+1} &= C^i && (24)\\
C^{i+1} &= B^i && (25)\\
B^{i+1} &= A^i && (26)\\
A^{i+1} &= T_1 + T_2 && (27)\\
& && (28)
\end{aligned}
$$

for $i = 0 \ldots, 63$.

The 32-bit constants K^i are different in each of the 64 steps. We refer to the description of SHA-2 [9] for their exact value. The output after 64 steps, $A^{64}, B^{64}, C^{64}, D^{64}, E^{64}, F^{64}, G^{64}, H^{64}$ is then used to update the chaining variables $AA, BB, CC, DD, EE, FF, GG, HH$.

There are no attacks reported on SHA-2 in the open literature even though the relative number of steps for each application of the compression function is much lower compared with SHA-1. On the other hand, two variables are updated at each step whereas only one is updated for SHA-1. We expect that SHA-2 is a secure as SHA-1 in terms of inverting the hash function or finding collisions, but no thorough security analysis has been reported in the open literature yet. We conjecture that SHA-2 used as a block cipher has the same security level as SHA-1.

4 SHACAL or using SHA in encryption mode

SHA was never defined to be used for encryption. However, the compression function can be used for encryption. Each of the round steps are invertible in the round variables. Therefore, if one inserts a secret key in the message and a plaintext as the initial value, one gets an invertible function from the compression function by ignoring the final addition with the initial values. This is the encryption mode of SHA considered for SHACAL. Thus SHACAL-1 is a 160-bit block cipher defined for a 512-bit secret key and SHACAL-2 is a 256-bit block cipher defined with a 512-bit secret key. Shorter keys may be used by padding the key with zeroes to a 512-bit string. However, SHACAL is not intended to be used with a key shorter than 128 bits.

5 Acknowledgements

We would like to thank Lars R. Knudsen and Matt J. Robshaw for their extensive security analysis on SHACAL-1; without their help this submission would not have been possible. We would also like to thank the NESSIE project for suggesting to extend our submission to 256-bit blocks using SHA-2. Finally, we would like to thank Julien Brouchier and Antoon Bosselaers for the tremendous help they provided on all implementation aspects of this submission.

References

[1] F. Chabaud and A. Joux. Differential collisions in SHA-0. In H. Krawczyk, editor, *Advances in Cryptology: CRYPTO'98, LNCS 1462*, pages 56–71. Springer Verlag, 1999.

[2] H. Dobbertin. Cryptanalysis of MD5 compress. Presented at the rump session of EUROCRYPT'96, May 1996.

[3] H. Dobbertin. Cryptanalysis of MD4. To appear in Journal of Cryptology, 1996.

[4] H. Handschuh, D. Naccache. SHACAL. In *Proceedings of the First Open NESSIE Workshop*, November 2000.

[5] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[6] R.L. Rivest. The MD4 message digest algorithm. In S. Vanstone, editor, *Advances in Cryptology - CRYPTO'90, LNCS 537*, pages 303–311. Springer Verlag, 1991.

[7] R.A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer Verlag, 1986.

[8] Secure Hash Algorithm FIPS 180-1, US Department of Commerce, N.I.S.T., 1995.

[9] New Secure Hash Algorithms http://csrc.nist.gov/cryptval/shs.html, US Department of Commerce, N.I.S.T., 2000.