Lappeenranta University of Technology

Ti5318800 - SECURED COMMUNICATIONS

SEMINAR WORK

Whirlpool hashing function

Arsi Hartikainen	0226485
Timo Toivanen	0221846
Herkko Kiljunen	0222405

1 Introduction

Whirlpool is a one-way hashing function designed by Paulo S.L.M. Barreto and Vincent Rijmen (also of AES fame). It was originally submitted to NESSIE (New European Schemes for Signatures, Integrity and Encryption) project and is the only hash function alongside SHA-256, SHA-384 and SHA-512 in the NESSIE portfolio.

Whirlpool is based on 512-bit block cipher, which structure is similar to Rijndael (AES). It uses 512-bit keys. The block cipher is dedicated only to be used for hashing, which is very exceptional in cryptography i.e. Whirlpool block cipher will most likely never be used for standalone encryption. It is designed for both software and hardware implementations, with compactness and performance in mind.

2 Goals

The security goals for Whirlpool are:

Assume we take as hash result the value of any n-bit substring of the full Whirlpool output.

- The expected workload of generating a collision is of the order of $2^{n/2}$ executions of Whirlpool.
- Given an n-bit value, the expected workload of finding a message that hashes to that value is of the order of 2^n executions of Whirlpool.
- Given a message and its n-bit hash result, the expected workload of finding a second message that hashes to the same value is of the order of 2^n executions of Whirlpool.
- Moreover, it is infeasible to detect systematic correlations between any linear combination of input bits and any linear combination of bits of the hash result. It is also infeasible to predict what bits of the hash result will change value when certain input bits are flipped, i.e. Whirlpool is resistant against differential attacks.

[1]

3 How it Works

3.1 Function

The Whirlpool hash function can be expressed as follows:

H_0 = initial value

$H_i = W(H_{i-1}, m_i) \oplus H_{i-1} \oplus m_i$ = intermediate value

where m_1, m_2, m_3, …, m_t are the message blocks and thus H_t is the hash code value. W is the Whirlpool block cipher.

This is a one iteration of Whirlpool.

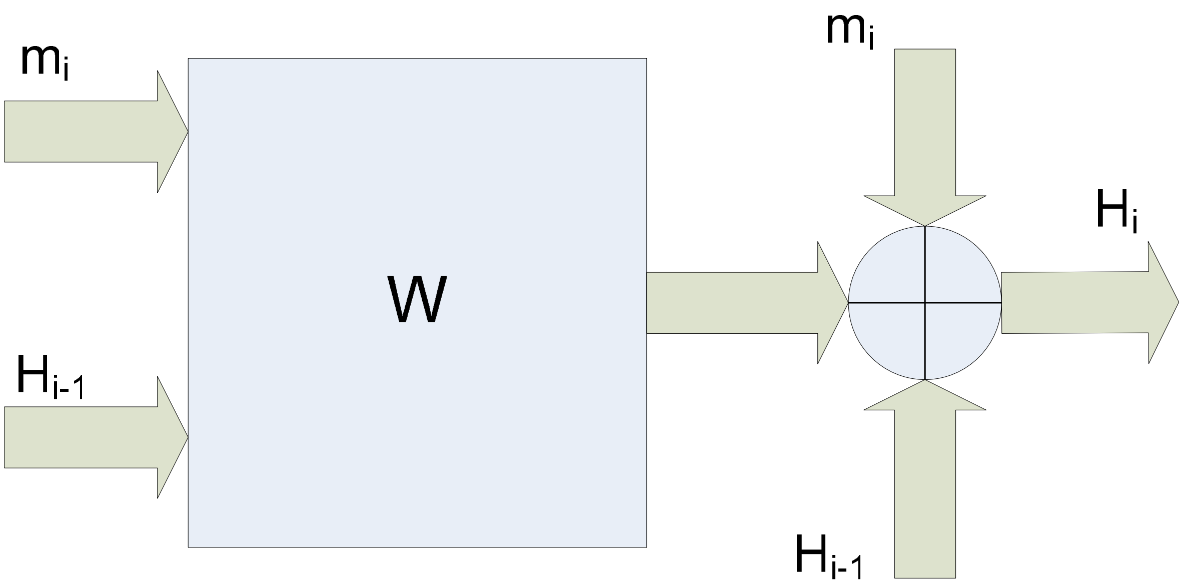

Figure 1Single iteration of Whirlpool

The complete digest is produced in four steps.

Step 1: Padding

Message is padded to odd multiple of 256 bits. In the case where the unpadded message is already of that length it is padded with 512 bits (2x256), which is the maximum padding length. Minimum is naturally 1 bit.

The first padding bit is always 1 and the rest are zeros.

Step 2: Message length

The length of the unpadded message is appended to the message. The length is expressed as a 256 bit unsigned integer, with the most significant byte being the leftmost.

After this step the message length is n x 512 bits (n=1, 2, …).

Step 3: Hash matrix initialization

The results of the hash function (both intermediate and final) and stored in an 8x8 matrix. Each element of the matrix is 8 bits (a byte) of the message, thus the hash matrix holds 512 bits in total.

The first matrix H_0 is initialized with zeros (each byte is 0000 0000)

Step 4: Block cipher

The block cipher processes the message in 512-bit blocks.

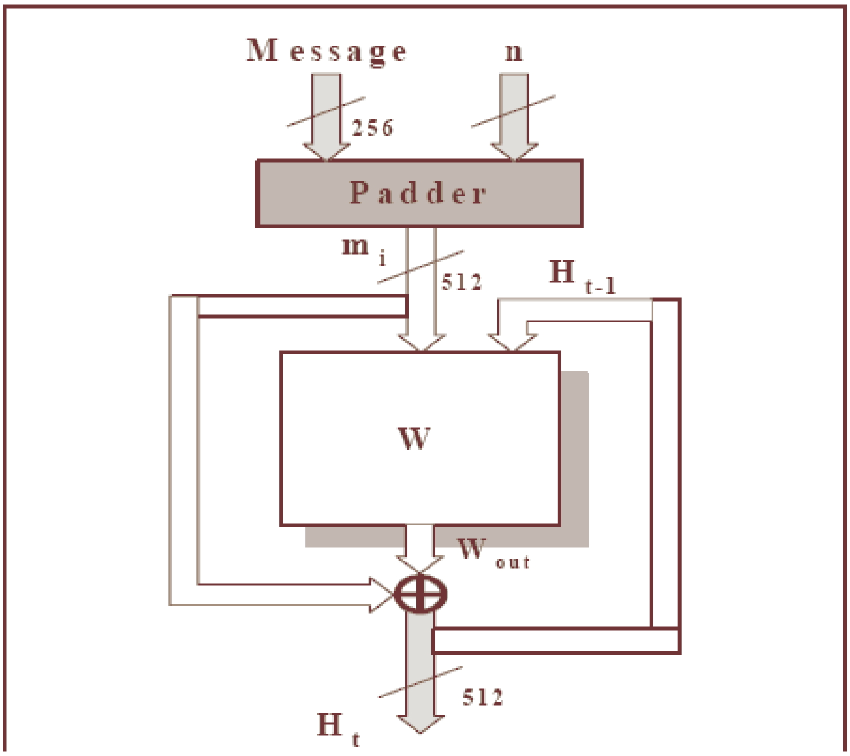

Figure 2 The generation of the Whirlpool digest

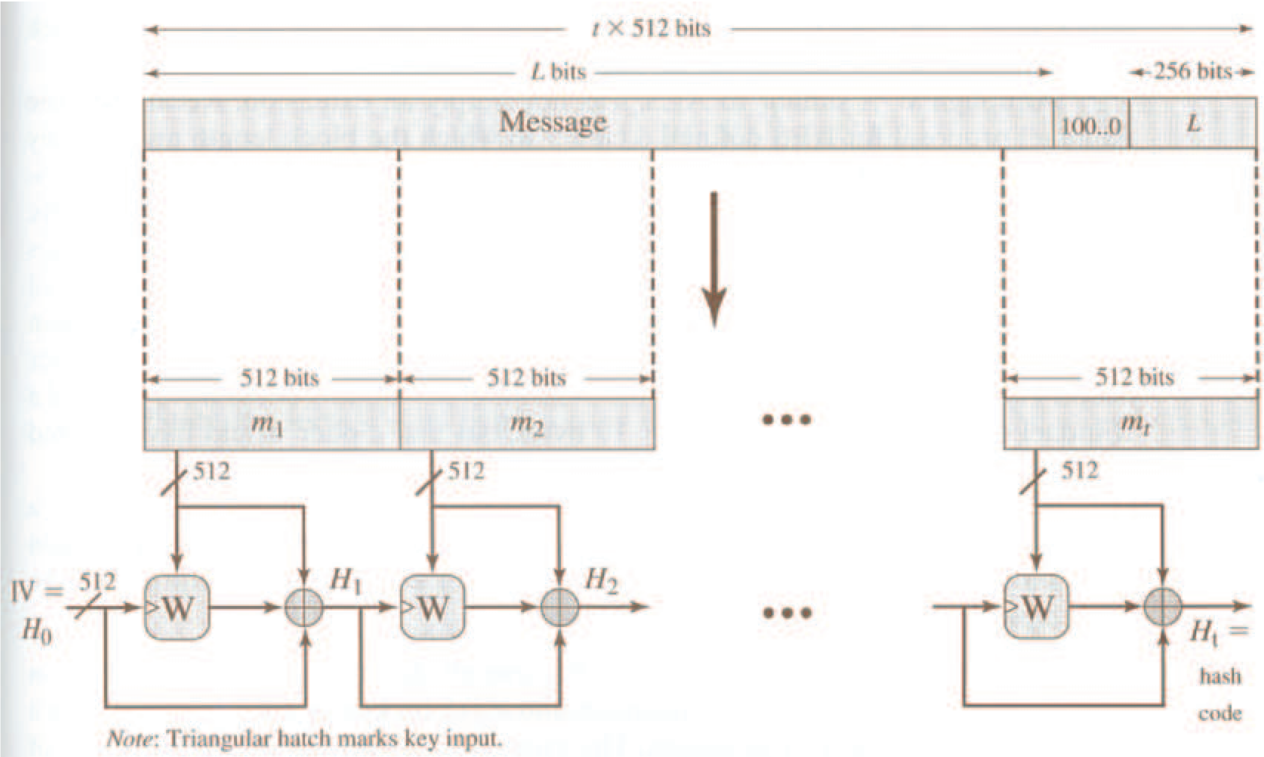

Figure 3 Structure of Whirlpool

3.1.1 Block cipher W

The block cipher W has similar structure and uses same elementary functions as AES. W uses 512-bit keys and 512-bit blocks while block length of AES is 128 and key length is 128, 192 or 256. W operates with 8x8 byte matrixes because it's faster than using for example 4x16 matrixes. 4x16 byte matrix requires more rounds than 8x8 byte matrix.

3.1.1.1 Overall Structure

The encryption algorithm takes 512-bit plaintext block and 512-bit key as input and produces 512-bit cipher text as output. The encryption algorithm uses four different functions or transformations: add key (AK), substitute bytes (SB), shift columns (SC), and mix rows (MR). Overall structure of W block cipher is shown in Figure 4. Before first round W, consists single application of AK, that's followed by ten rounds that involve use of all four operations. One round can be expressed as round function RF.

$$RF(K_r) = AK[K_r] \circ MR \circ SC \circ SB$$

where K_r is the round key matrix for round r.

The overall algorithm can be defined as follows:

$$W(K) = \left(\bigcirc_{r=1}^{10} RF(K_r)\right) \circ AK(K_0)$$

Large circle indicates iteration of composition function, with index r running from 1 to 10. Plaintext input to W is single 512-bit block. Block is 8x8 byte matrix labelled CState. Figure 5 illustrates how matrix is filled with 512-bit block. First eight bytes of 512 bit plaintext input are put in first row of the matrix. Second eight bytes to second row and so on.

Whirlpool uses 512-bit key, called KState. Like CState, KState is also a 8x8 matrix. Key is used as input to initial AK function. On rounds 2 to 10 previous hash value is used as a key. So, output of the first round is the key for the second round (Figure 3). AK function is described in more detail later.

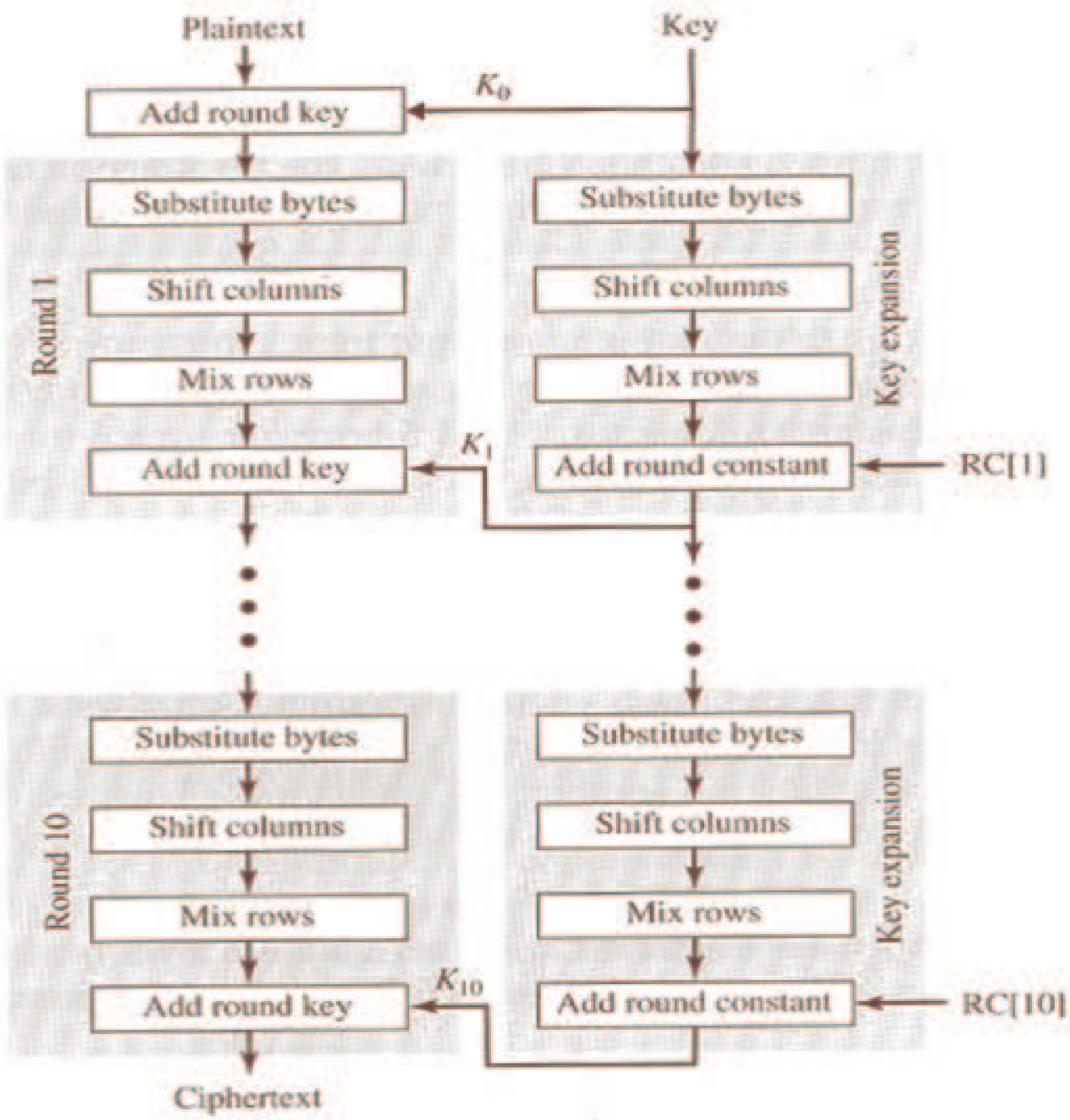

Figure 4 Whirlpool cipher

[4]

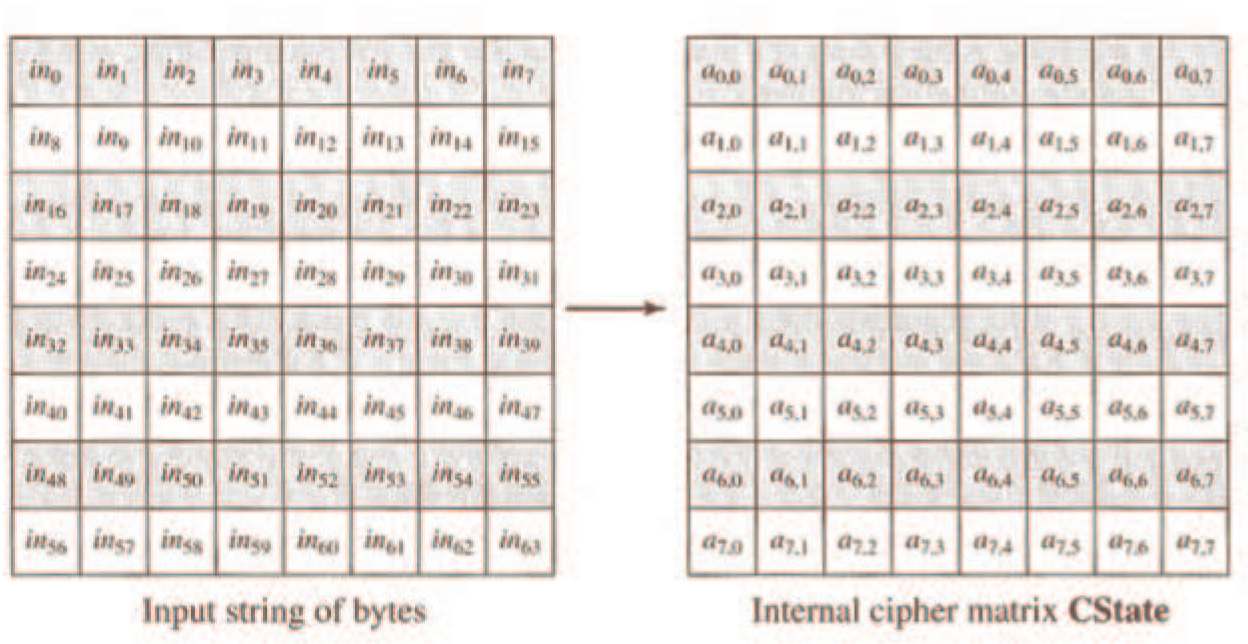

Figure 5 Whirlpool matrix structure

3.1.1.2 Substitute byte (SB)

In Whirlpool, the substitution box (S-box) (Figure 6) is a 16x16 table which contains all possible 8-bit values, i.e. 256 permutations. S-box is used for nonlinear mapping. Here is how:

Take four leftmost bits from a CState byte and use them as a row indicator for S-Box and take four rightmost bits and use them for a column index. Look up the proper 8-bit value from S-box using these indices and you have the output value.

Mathematically the function can be expressed as follows:

$$B = SB(A)$$

$$b_{i,j} = S[a_{i,j}]$$

where B is the output, A is the CState and $b_{i,j}$ is the value of S-box and $a_{i,j}$ represents the individual byte of CState. Indices i and j range from zero to seven (CState is 8 by 8 matrix). S here represents the process of S-box mapping.

3.1.1.2.1 The S-box

Below is a complete Whirlpool S-box.

	00_x	01_x	02_x	03_x	04_x	05_x	06_x	07_x	08_x	09_x	$0A_x$	$0B_x$	$0c_x$	$0d_x$	$0E_x$	$0F_x$
00_x	18_x	23_x	$c6_x$	$E8_x$	87_x	$B8_x$	01_x	$4F_x$	36_x	$A6_x$	$d2_x$	$F5_x$	79_x	$6F_x$	91_x	52_x
10_x	60_x	Bc_x	$9B_x$	$8E_x$	$A3_x$	$0c_x$	$7B_x$	35_x	$1d_x$	$E0_x$	$d7_x$	$c2_x$	$2E_x$	$4B_x$	FE_x	57_x
20_x	15_x	77_x	37_x	$E5_x$	$9F_x$	$F0_x$	$4A_x$	dA_x	58_x	$c9_x$	29_x	$0A_x$	$B1_x$	$A0_x$	$6B_x$	85_x
30_x	Bd_x	$5d_x$	10_x	$F4_x$	cB_x	$3E_x$	05_x	67_x	$E4_x$	27_x	41_x	$8B_x$	$A7_x$	$7d_x$	95_x	$d8_x$
40_x	FB_x	EE_x	$7c_x$	66_x	dd_x	17_x	47_x	$9E_x$	cA_x	$2d_x$	BF_x	07_x	Ad_x	$5A_x$	83_x	33_x
50_x	63_x	02_x	AA_x	71_x	$c8_x$	19_x	49_x	$d9_x$	$F2_x$	$E3_x$	$5B_x$	88_x	$9A_x$	26_x	32_x	$B0_x$
60_x	$E9_x$	$0F_x$	$d5_x$	80_x	BE_x	cd_x	34_x	48_x	FF_x	$7A_x$	90_x	$5F_x$	20_x	68_x	$1A_x$	AE_x
70_x	$B4_x$	54_x	93_x	22_x	64_x	$F1_x$	73_x	12_x	40_x	08_x	$c3_x$	Ec_x	dB_x	$A1_x$	$8d_x$	$3d_x$
80_x	97_x	00_x	cF_x	$2B_x$	76_x	82_x	$d6_x$	$1B_x$	$B5_x$	AF_x	$6A_x$	50_x	45_x	$F3_x$	30_x	EF_x
90_x	$3F_x$	55_x	$A2_x$	EA_x	65_x	BA_x	$2F_x$	$c0_x$	dE_x	$1c_x$	Fd_x	$4d_x$	92_x	75_x	06_x	$8A_x$
$A0_x$	$B2_x$	$E6_x$	$0E_x$	$1F_x$	62_x	$d4_x$	$A8_x$	96_x	$F9_x$	$c5_x$	25_x	59_x	84_x	72_x	39_x	$4c_x$
$B0_x$	$5E_x$	78_x	38_x	$8c_x$	$d1_x$	$A5_x$	$E2_x$	61_x	$B3_x$	21_x	$9c_x$	$1E_x$	43_x	$c7_x$	Fc_x	04_x
$c0_x$	51_x	99_x	$6d_x$	$0d_x$	FA_x	dF_x	$7E_x$	24_x	$3B_x$	AB_x	cE_x	11_x	$8F_x$	$4E_x$	$B7_x$	EB_x
$d0_x$	$3c_x$	81_x	94_x	$F7_x$	$B9_x$	13_x	$2c_x$	$d3_x$	$E7_x$	$6E_x$	$c4_x$	03_x	56_x	44_x	$7F_x$	$A9_x$
$E0_x$	$2A_x$	BB_x	$c1_x$	53_x	dc_x	$0B_x$	$9d_x$	$6c_x$	31_x	74_x	$F6_x$	46_x	Ac_x	89_x	14_x	$E1_x$
$F0_x$	16_x	$3A_x$	69_x	09_x	70_x	$B6_x$	$d0_x$	Ed_x	cc_x	42_x	98_x	$A4_x$	28_x	$5c_x$	$F8_x$	86_x

Figure 6 Whirlpool S-box

S-box may be generated in the following way:

Refer to the figure 7. There is two nonlinear layers, both having two 4*4 substitution boxes, labelled E and called E-boxes. In centre there is a 4x4 randomly generated box, labelled R. E-boxes and R takes a 4-bit input and outputs 4-bits.

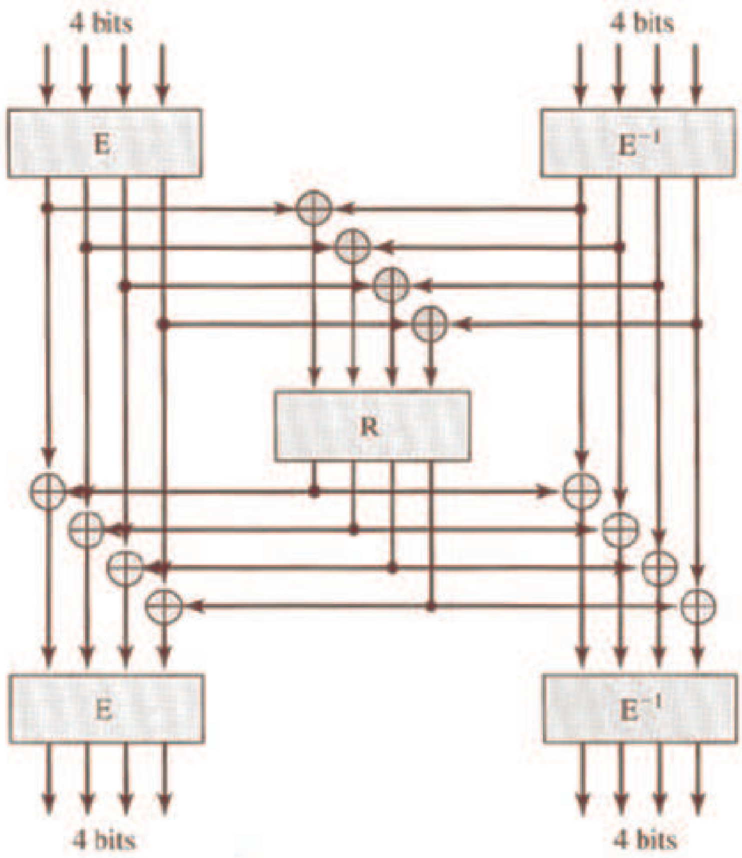

Figure 7 Generation of S-box

E-box is defined as $E(u) = \{B\}^u$ if $u \neq \{F\}$ and $E(\{F\}) = 0$

3.1.1.3 Shift Columns (SC)

The permutation layer makes each column of CState to shift downwards circularly, except the first column. To second column, a 1-byte shift is performed. For the third column, a 2-byte shift is performed. This is made to each column.

SC Function, where A is input matrix and B is output matrix:

$$B = SC(A) \Leftrightarrow b_{i,j} = a_{(i-j)\bmod 8,\, j} \quad 0 < i,\, j < 7$$

The shift column transformation is very important because CState is an 8x8 matrix, where first 8 bytes of plaintext is copied to first row, and so on. The shift moves a single byte from one row to another. The transformation also makes certain, that bytes of one row are put in to eight different rows.

[4]

3.1.1.4 Mix rows (MR)

MR function is the linear diffusion layer of Whirlpool block cipher. For diffusion functions, each output bit is affected by several input bits. The MR transformation can be expressed with matrices as follows:

B=AC, A being the input and B the output and C the transformation matrix. Notice that in C (Figure 8) the nth row is (n-1)th row shifted one right so that the last element of a row becomes the first after the shift. C is a maximum distance separable (MDS) matrix.
[4]
According to a paper by Pascal Junod and Serge Vaudenay, MDS matrices provide for optimal diffusion. [5]

$$C = \begin{bmatrix} 01 & 01 & 04 & 01 & 08 & 05 & 02 & 09 \\ 09 & 01 & 01 & 04 & 01 & 08 & 05 & 02 \\ 02 & 09 & 01 & 01 & 04 & 01 & 08 & 05 \\ 05 & 02 & 09 & 01 & 01 & 04 & 01 & 08 \\ 08 & 05 & 02 & 09 & 01 & 01 & 04 & 01 \\ 01 & 08 & 05 & 02 & 09 & 01 & 01 & 04 \\ 04 & 01 & 08 & 05 & 02 & 09 & 01 & 01 \\ 01 & 04 & 01 & 08 & 05 & 02 & 09 & 01 \end{bmatrix}$$

Figure 8 Transformation matrix C

[4]

3.1.1.5 Add key (AK)

Add key layer simply XORs 512 bits of CState with 512 bits of round key bitwise. AK can be expressed by following function, where A is the input matrix, B is the output matrix and K_i is the round key.

$$B = AK[K_i](A) \Leftrightarrow b_{i,j} = a_{i,j} \oplus k_{i,j} \quad 0 \le i, j \le 7$$

3.1.1.6 Key expansion

The round key, K, used in the AK layer is generated using the very cipher itself.

For key expansion, the round constant acts as the round key for the add key layer. The round constant for row *r* can be defined as follows:

$$RC\left[r\right]_{0,j} = S\left[8(r-1)+j\right] \qquad 0 \le j \le 7 \;\; 1 \le r \le 10$$

$$RC\left[r\right]_{i,j} = 0 \qquad 1 \le i \le 7 \quad 0 \le j \le 7 \quad 1 \le r \le 10$$

All rows expect first one are zeros. S in the formula refers to the S-box (Figure 6).

The whole process for key expansion can be expressed with formula:

$$K_r = RF\left[RC\left[r\right]\right](K_{r-1}) \qquad (RF(K_r) = AK[K_r] \circ MR \circ SC \circ SB\ )$$

4 Security

There might be doubts that there is some hidden weakness in algorithm. That's why designers asked NESSIE effort to evaluate algorithm. No hidden weaknesses found in evaluation. So, designers declare that there are no hidden weaknesses inserted by them in the Whirlpool primitive.

[1]

The hash code length is 512-bits, same as the longest hash code with SHA. According to NESSIE evaluation of Whirlpool "The Statistical Evaluation of the NESSIE Submission Whirlpool" hash should be random. Evaluation stated that: "The statistical test results for the NESSIE submission Whirlpool do not indicate a deviation from random behaviour." [6] Compared to other block ciphers Whirlpool doesn't have same randomness related weakness as block ciphers usually tend to have. Overall structure of Whirlpool hash function hash been shown to be resistant to the usual attacks on block-cipher-based hash codes.
[4]

Yet there's no known successful attacks made against Whirlpool. Though reason for that might be that Whirlpool is rather new hash function (version 2 released 2003) and it hasn't been used very much.

5 Implementation

It is possible to implement Whirlpool efficiently on different platforms; different optimizations and tradeoffs are possible. Whirlpool is not specifically oriented toward any platform, but it is efficient on many of them.

It does not require too much storage space (either for code or for tables), and can therefore implemented in quite restricted systems like smart cards. With large cache memory it is able to achieve higher performance. It does not use expensive or unusual instructions that must be built in the processor. Analysis is easier because the mathematical simplicity of the primitive resulting from the design strategy. It also has a very long hash length, which provides increased protection against birthday attack and offers a larger internal state for entropy containment, as is needed for certain classes of pseudo-random number generators.
[1]

There has been little implementation experience with Whirlpool. Study by P. Kitsos and O. Koufopavlou has been published. They compared Whirlpool with a number of other secure hash functions, including all of the versions of SHA. The authors developed multiple hardware implementations of each hash function.

Their conclusions were that Whirlpool requires more hardware resources compared with the other hash families implementations. Also Whirlpool's throughput was better. Finally, the Whirlpool has the smaller algorithm execution latency. It needs only 10 clock cycles in order to transform each block compared with the 64 block cycles of the MD5, and SHA-2(256), and 80 clock cycles of the RIPEMD-160, SHA-1, SHA-2(384, 512).
[3]

Whirlpool has been included in several cryptography libraries, such as gnu.crypto package and BouncyCastle for Java. There is also a C implementation available free from the Whirlpool authors' website and Crypto++ library for C++ has a Whirlpool implementation.

6 Comparison to alternatives

6.1 AES

Rijndael has shown good performance in both hardware and software and it is well suited to low memory requirements. Because Whirlpool is based on same algorithm as AES, we can expect similar performance and characteristics.

6.2 SHA-512

SHA-512 and Whirlpool hash functions are both 64-bit hash functions. That's why they are expected to be well suited for 64-bit processors. Matsui and Fukuda have compared these two hash functions on a Pentium III and Pentium 4 computers.

Whirlpool runs faster in Pentium III and Pentium 4 Prescott in single message hashing, due to a better instruction scheduling. But the effect of double message hashing is evident; the SHA-512 becomes more than 30 % faster than Whirlpool. (I) and (II) in results means straightforward single message hashing using 64-bit MMX instructions and double message hashing using 128-bit XMM instructions.

1 block = 128 bytes	Pentium III	Pentium 4 Northwood		Pentium 4 Prescott	
	(I)	(I)	(II)	(I)	(II)
μops/block	13924	8710	4363	8710	4363
cycles/block	5148	4666	2826	5294	3111
cycles/byte	40.2	36.5	22.1	41.4	24.3
μops/cycles	2.70	1.87	1.54	1.65	1.40

Figure 9 Results of SHA512

1 block = 64 bytes	Pentium III	Pentium 4 Northwood	Pentium 4 Prescott
μops/block	5206	5526	5526
cycles/block	2061	3024	2319
cycles/byte	32.2	47.3	36.2
μops/cycles	2.53	1.83	2.38

Figure 10 Results of Whirlpool

[2]

7 Conclusion and analysis

Whirlpool is a promising function. It can do only one thing (hashing) but it does it well, being fast on platforms with plentiful resources while being able to scale well to hardware level. So far it has been collision free. SHA-1 and MD5 have been broken (found not collision free) and will be replaced with better functions. NESSIE suggests Whirlpool and SHA-256, SHA-384 and SHA-512.

8 References

1. Barreto, Paulo S.L.M. & Rijmen, Vincent: The Whirlpool Hashing Function. Available: http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html

2. Mitsuru Matsui, Sayaka Fukuda: “How to maximize software performance of symmetric primitives”. EICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, v E89-A, n 1, January, 2006, p 2-10

3. P. Kitsos, O. Koufopavlou: “Efficient Architecture and Hardware Implementation of the Whirlpool Hash Function”. IEEE Transactions on Consumer Electronics, Vol. 50, No. 1, Feb. 2004

4. Stallings, William: “Cryptography and Network Security”, Prentice Hall 2005, 4th edition.

5. Junod, P. & Vaudenay S.: “Perfect Diffusion Primitives for Block Ciphers” Avalable: http://crypto.junod.info/sac04b.pdf

6. Marcus Schafheutle: “The Statistical Evaluation of the NESSIE Submission Whirlpool” Available: https://www.cosic.esat.kuleuven.be/nessie/reports/phase1/sagwp3-037_1.pdf