The Dynamic SHA Hash Function

Xu ZiJie
xuzijiewz@gmail.com

Outline

1. Introduction
2. Design considerations / Responses
3. Dynamic SHA
4. Implementations
5. Security Analysis
6. Improvements to resist known attack

1. Introduction

Why design Dynamic SHA?

Biham and Shamir discovere Differential cryptanalysis (1990). Considers step-by-step ``difference'' (XOR) between two computations…

Wang used Differential cryptanalysis to break MD5；

NIST SHA-3 competition

Input: 0 to 2^{64}-1 bits, size not known in advance

Output sizes 224, 256, 384, 512 bits

Preimage resistance.

Second preimage resistance.

Collision-resistance.

Pseudorandomness, simplicity, flexibility, speedy, …

2. Design Considerations / Responses

How to resist Differential analysis?

If the difference of working variables between two computations is very complex, it will be hard to analyse difference. There are two ways that we can use to resist differential analysis :

More rounds or steps. After more rounds, the difference will be more complex. It is harder to control the difference of working variables.

Use the functions that need huge ANFs to describe. And it need huge ANFs to describe Data-Depend functions. So Dynamic SHA use Data-Depend functions to resist differential analysis.

What is new in Dynamic SHA?

Dynamic SHA has data-depend function R. When input different message, different rotate right operation maybe done. After 48 steps, the (one block) message value space is divided into 2^{235} (resp. 2^{282}) parts. In different part, the rotate operations are different.

The ANFs that describe function R has up to 2^{229}(resp. 2^{454}) items. This will make the difference of working variables very complex.

3. Dynamic SHA

Operations

Dynamic SHA-224/256 operations on 32-bit words.
Dynamic SHA-384/512 operations on 64-bit words.
The following operations:

- $+$ modulo 2^{32} or modulo 2^{64} .
- $\wedge$ AND.
- $\vee$ OR.
- $\oplus$ XOR.
- $\neg$ Negation.

$SHR^n(x)$ shift right operation.
$SHL^n(x)$ shift left operation.
$ROTR^n(x)$ rotate left (circular left shift) operation.

3.1 Functions and Constants

3.1.1. Initial Hash Value and Constants

Dynamic SHA-224	Dynamic SHA-256
$H_0^{(0)}$=0xc1059ed8, $H_1^{(0)}$=0x367cd507, $H_2^{(0)}$=0x3070dd17, $H_3^{(0)}$=0xf70e5939, $H_4^{(0)}$=0xffc00b31, $H_5^{(0)}$=0x68581511, $H_6^{(0)}$=0x64f98fa7, $H_7^{(0)}$=0xbefa4fa4,	$H_0^{(0)}$=0x6a09e667, $H_1^{(0)}$=0xbb67ae85, $H_2^{(0)}$=0x3c6ef372, $H_3^{(0)}$=0xa54ff53a, $H_4^{(0)}$=0x510e527f, $H_5^{(0)}$=0x9b05688c, $H_6^{(0)}$=0x1f83d9ab, $H_7^{(0)}$=0x5be0cd19,

Table 1. Initial Hash Value of Dynamic SHA-224/256

Dynamic SHA-384	Dynamic SHA-512
$H_0^{(0)}$=0xcbbb9d5dc1059ed8, $H_1^{(0)}$=0x629a292a367cd507, $H_2^{(0)}$=0x9159015a3070dd17, $H_3^{(0)}$=0x152fecd8f70e5939, $H_4^{(0)}$=0x67332667ffc00b31, $H_5^{(0)}$=0x8eb44a8768581511, $H_6^{(0)}$=0xdb0c2e0d64f98fa7, $H_7^{(0)}$=0x47b5481dbefa4fa4,	$H_0^{(0)}$=0x6a09e667f3bcc908, $H_1^{(0)}$=0xbb67ae8584caa73b, $H_2^{(0)}$=0x3c6ef372fe94f82b, $H_3^{(0)}$=0xa54ff53a5f1d36f1, $H_4^{(0)}$=0x510e527fade682d1, $H_5^{(0)}$=0x9b05688c2b3e6c1f, $H_6^{(0)}$=0x1f83d9abfb41bd6b, $H_7^{(0)}$=0x5be0cd19137e2179,

Table 2. Initial Hash Value of Dynamic SHA-384/512

Dynamic SHA-224/256	Dynamic SHA-384/512
C_0=0x5a827999, C_1=0x6ed9eba1, C_2=0x8f1bbcdc,	C_0=0x5a82799950a28be6, C_1=0x6ed9eba15c4dd124, C_2=0x8f1bbcdc6d703ef3,

Table 3. Constants of Dynamic SHA

3.1.2. Function G

Logical functions G_0,G_1,G_2,G_3 operate on three words produces a word y as output, these logical functions will be repeat used in order:

$$y = \begin{cases} G_0(a,b,c) = a \oplus b \oplus c \\ G_1(a,b,c) = (a \wedge b) \oplus c \\ G_2(a,b,c) = (\neg(a \vee c)) \vee (a \wedge (b \oplus c)) \\ G_3(a,b,c) = (a \vee \neg c) \vee (\neg(a \vee (b \oplus c))) \end{cases}$$

It can describe logical functions with Algebraic Normal Form(ANF) and Numerical Normal Form (NNF) as follow:

$$y_i = \begin{cases} G_0(a_i, b_i, c_i) = a_i \oplus b_i \oplus c_i \\ G_1(a_i, b_i, c_i) = a_i b_i \oplus c_i \\ G_2(a_i, b_i, c) = 1 \oplus a_i \oplus c_i \oplus a_i b_i \\ G_3(a_i, b_i, c_i) = 1 \oplus b_i \oplus c_i \oplus a_i c_i \end{cases}$$

$$y_i = \begin{cases} G_0(a_i, b_i, c_i) = a_i + b_i + c_i - 2a_i b_i - 2a_i c_i - 2c_i b_i + 4a_i b_i c_i \\ G_1(a_i, b_i, c_i) = c_i + a_i b_i - 2a_i b_i c_i \\ G_2(a_i, b_i, c_i) = 1 - a_i - c_i + 2a_i c_i + a_i b_i - 2a_i b_i c_i \\ G_3(a_i, b_i, c_i) = 1 - b_i - c_i + 2b_i c_i + a_i c_i - 2a_i b_i c_i \end{cases}$$

y_i,a_i,b_i,c_i is i-th bit of y,a,b,c.

- Ps: Claude Carlet had recounted NNF in “Boolean Functions for Cryptography and Error Correcting Codes”[2008]. NNF and ANF is equivalent, NNF and ANF can be transformed each other.

3.1.3. Function R

Function R operates on eight words a,b,c,d,e,f,g,h, produces a word y as output.

Dynamic SHA-224/256	t=(((((a+b) ⊕ c)+d) ⊕ e)+f) ⊕ g t=(SHR^{17}(t) ⊕ t) ∧ (2^{17}-1) t=(SHR^{10}(t) ⊕ t) ∧ (2^{10}-1) t=(SHR^{5}(t) ⊕ t) ∧ 31 y=ROTR^{t}(h)
Dynamic SHA-384/512	t=(((((a+b) ⊕ c)+d) ⊕ e)+f) ⊕ g t=(SHR^{36}(t) ⊕ t) ∧ (2^{36}-1) t=(SHR^{18}(t) ⊕ t) ∧ (2^{18}-1) t=(SHR^{12}(t) ⊕ t) ∧ (2^{12}-1) t=(SHR^{6}(t) ⊕ t) ∧ 63 y=ROTR^{t}(h)

The ANFs that describe function R have up to 2^{229}(resp. 2^{454}) items. This will make the difference of output very complex.

The profit/cost of function R is very high. 

3.2 Preprocessing

At first, the message will be paded and divided into some blocks. These blocks will be inputed compress function H_{DSHA} in order. The output of the last compution will be truncated as 224/256/384/512 bits.

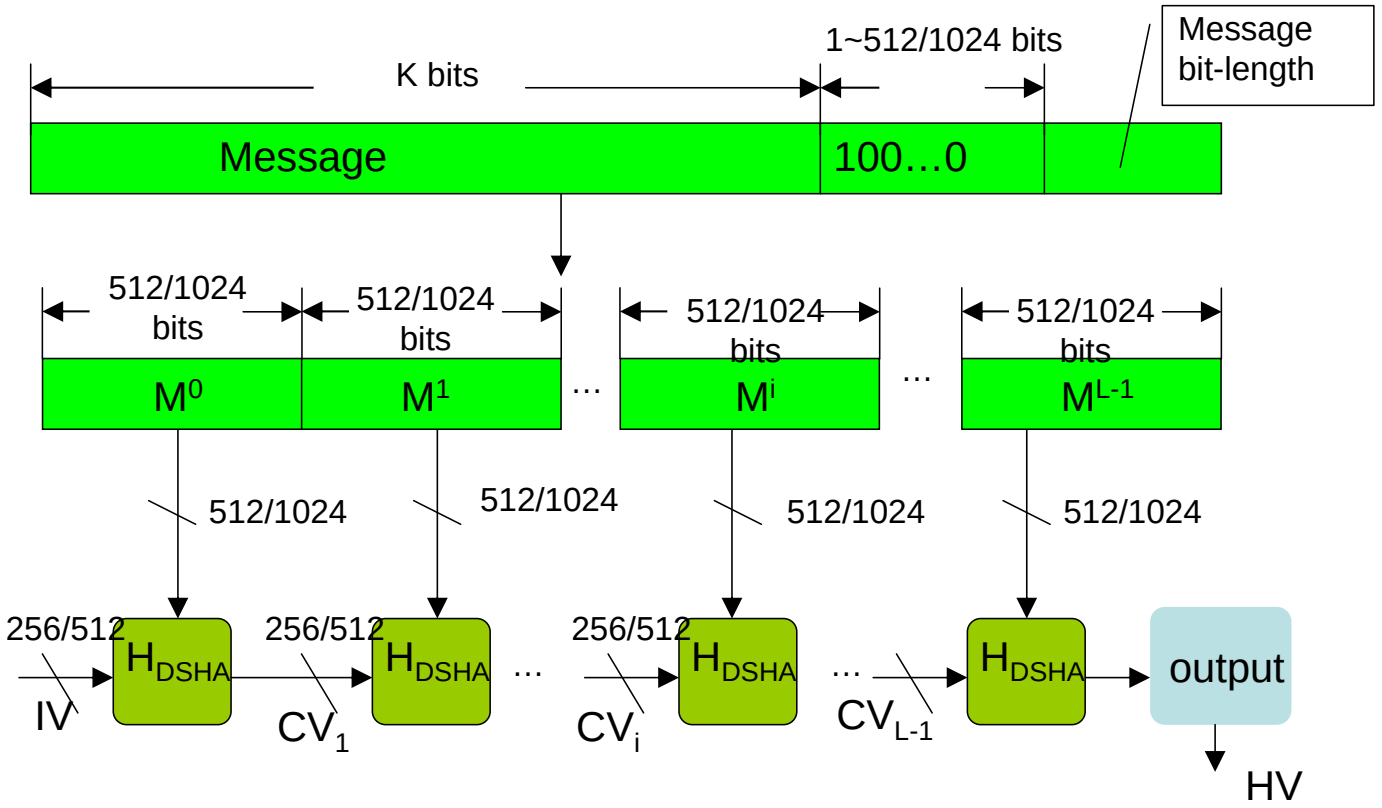

Fig 1. Preprocessing of Dynamic SHA

3.2.1 Compression function

Compression function inputs

16 words (512/1024 bits) data block
8 words (256/512 bits) chaining values

Compression function outputs

8 words (256/512 bits) chaining values

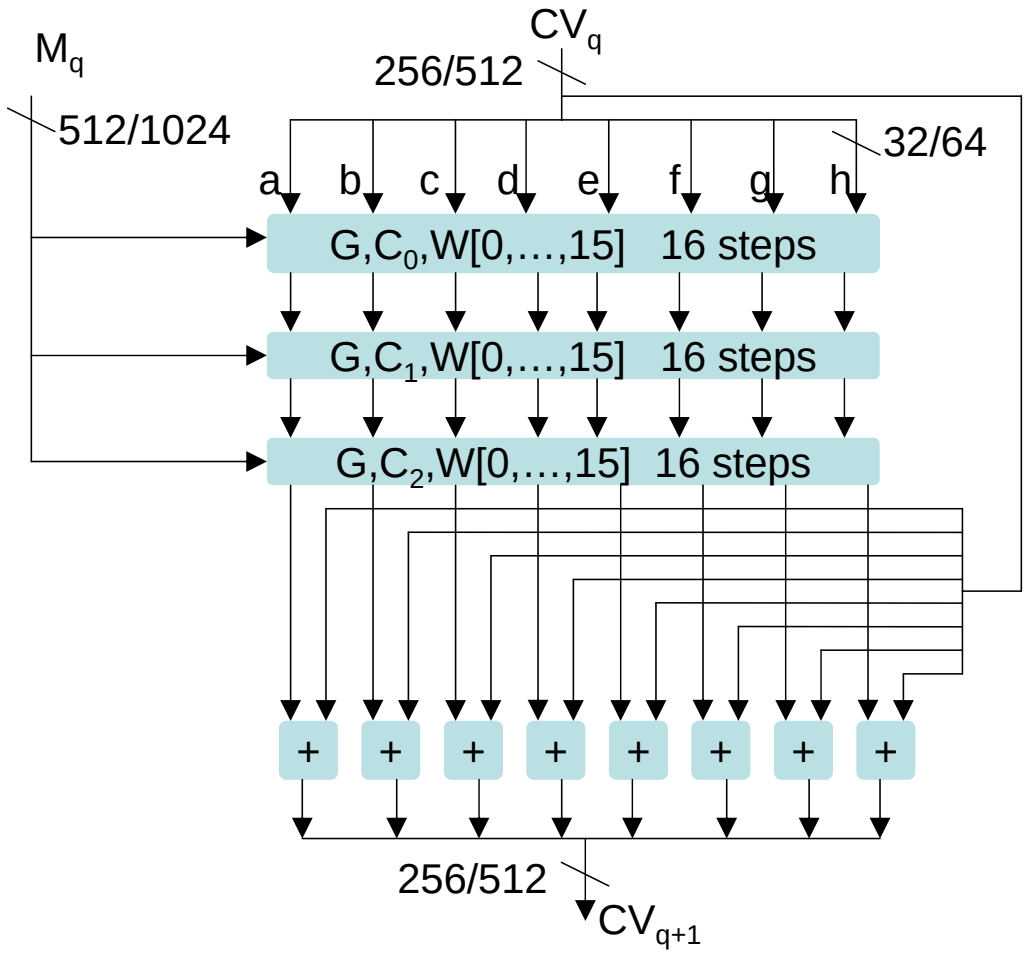

Fig 2. Compression function of Dynamic SHA

The t-th step compute:

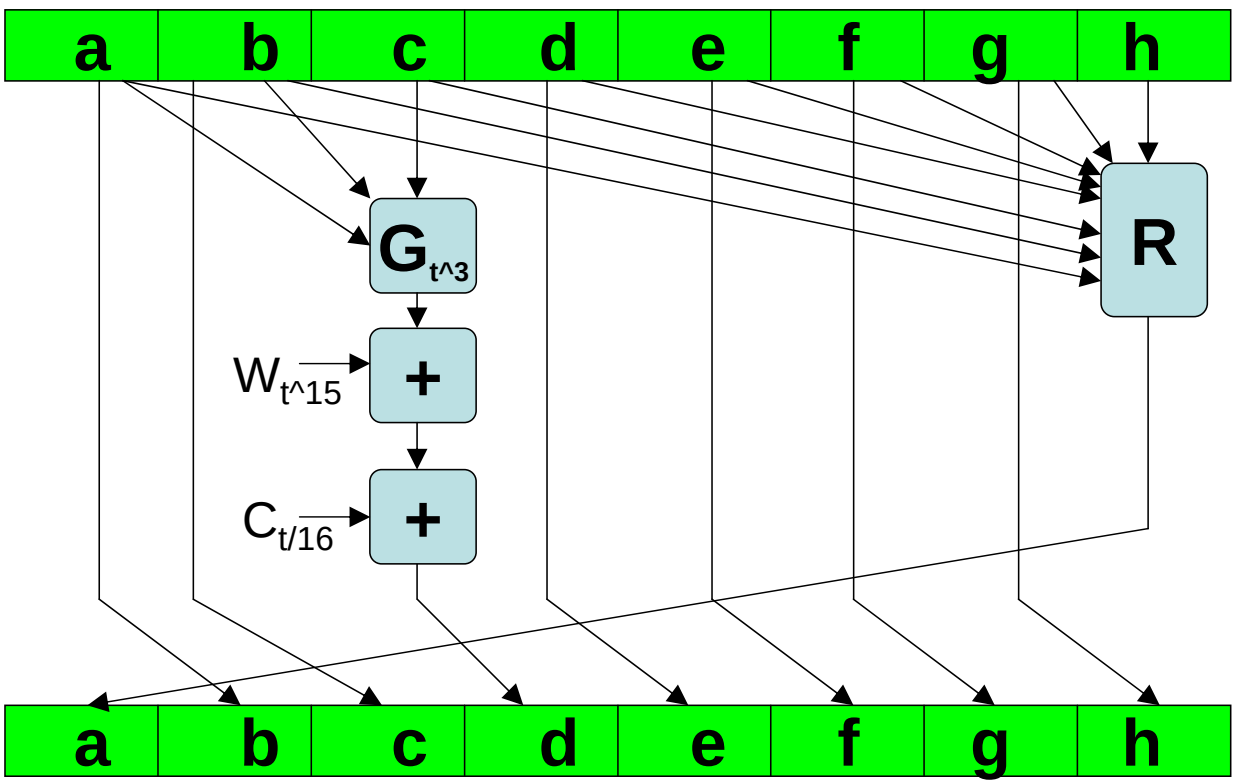

Fig 3. t-th step of compression function of Dynamic SHA

In t-th step, the (t^3)-th logical function is used.

Compression function code:

Input: $H_0^i, H_1^i, H_2^i, H_3^i, H_4^i, H_5^i, H_6^i, H_7^i, w_0, \ldots, w_{15}$

$a=H_0^i$; $b=H_1^i$; $c=H_2^i$; $d=H_3^i$;

$e=H_4^i$; $f=H_5^i$; $g=H_6^i$; $h=H_7^i$;

for i =0 **to 47**

{

T=R(a,b,c,d,e,f,g,h);

h=g; g=f; f=e; e=d;

$d=G_{t\hat{}3}(a,b,c)$;

c=b; b=a; a=T;

}

return $H_0^{i=}H_0^i+a$; $H_1^{i=}H_1^i+b$; $H_2^{i=}H_2^i+c$; $H_3^{i=}H_3^i+d$;

$H_4^{i=}H_4^i+e$; $H_5^{i=}H_5^i+f$; $H_6^{i=}H_6^i+g$; $H_7^{i=}H_7^i+h$;

4. Implementations

Dynamic SHA had been implemented on follow platform:

•Xilinx

•Keil uVision, the processor is Intel 80/87c58

•Wintel personal computer, with an Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 32-bit (x86) Edition.

The compiler is: The ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition.

•Wintel personal computer, with an Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 64-bit (x64) Edition.

The compiler is: The ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition.

5. Security Analysis

Birthday attack resistance

Birthday attack resistance of Dynamic SHA as follow:

Dynamic SHA-224	2^{-112}
Dynamic SHA-256	2^{-128}
Dynamic SHA-384	2^{-192}
Dynamic SHA-512	2^{-256}

When Dynamic SHA is analysed by differential analysis, what will happen?

The ANFs that describe function R have up to 2^{229} (resp. 2^{454}) items. The difference of working variables will be very complex.

If attacker guess the parameter in function R to reduce complexity, this will divide the message space to 2^{235} (resp. 2^{282}) parts. In different part, the rotate operations are different. The probability of get the collision is 2^{-235} (resp. 2^{-282}).

When Dynamic SHA is attacked by length extension attack and multicollision attack, what will happen?

In length extension attack and multicollision attack, it need find the collision. The probability of find collision of Dynamic SHA is $2^{-n/2}$, where n is output sizes in bits.

The hash value of Dynamic SHA-224/384 is truncated. In keyed hash, it can not get all bits of last compute. It is hard to be attacked by length extension attack.

There are some ways that resist length extension attack and multicollision attack. HMAC can resist length extension attack .

6. Improvements to resist known attack

There are some ways to attack hash algorithm. People are interested in length extension attack and multicollision attack. HMAC can resist length extension attack. There are some ways can stop these attack.

6.1. Improvement one

Let H(.) is hash function. M is message, and M^i are message block after padded, where $0 \leqslant i \leqslant L-1$.

$$hv = (H(M) \oplus H(C \oplus M^0 \oplus ... \oplus M^{L-1})) + H(M) \qquad (1)$$

Where C is constant . hv is hash value. (1) can stop length extension attack and multicollision attack.

If H is keyd hash function, it is hard to get H(M), then it is hard to attack (1) by length extension attack.

When the collision of H(M) is found, $H(C \oplus M^0 \oplus \ldots \oplus M^{L-1})$ maybe different. It is more harder to find multicollision of (1).

6.2. Improvement two

Let h(.) is compression function. M is message, and M^i are message block after padded, CV_i is chaining value, where $1 \leqslant i \leqslant L-1$. If the last chaining value is handled as fig 4 show:

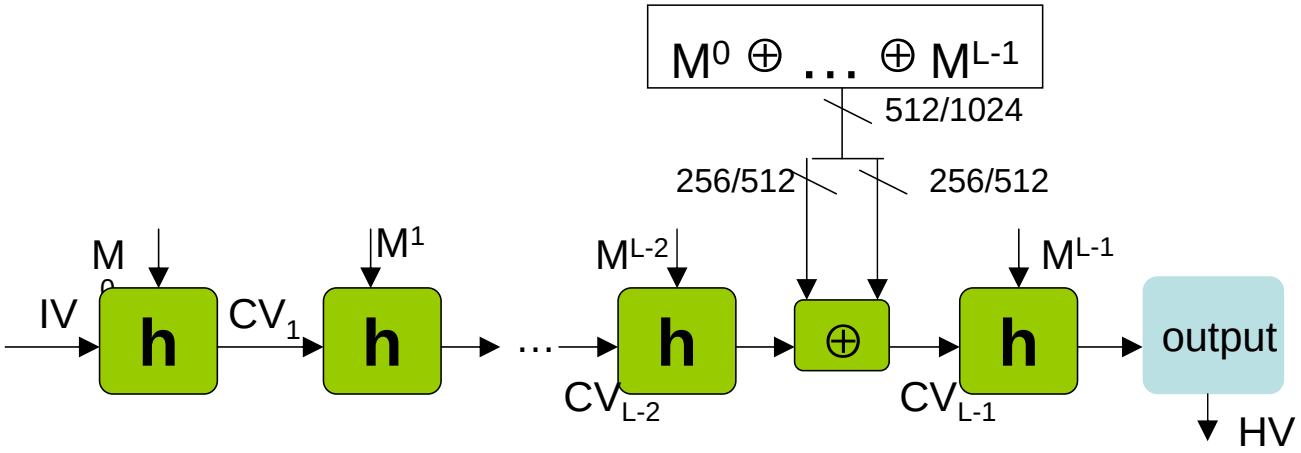

Fig 4. Improvement two

In improvement two as fig 4 show.

If H is keyd hash function, it is hard to get chaining value CV_{L-1} and CV_{L-2}. And when attacker try to pad some bits, chaining value CV_{L-1} maybe change. It is hard to attacked by length extension attack.

When it can find a different block data M^i that do not change chaining value CV_{L-2}, chaining value CV_{L-1} maybe different. It is hard to find multicollision by replace one block data M^i.

THANK YOU

END

January 12 . 2009