Dynamic SHA-2

Zijie Xu
E-mail: xuzijiewz@gmail.com

Abstract. In this paper I describe the construction of Dynamic SHA-2 family of cryptographic hash functions. They are built with design components from the SHA-2 family, but I use the bits in message as parameters of function G, R and ROTR operation in the new hash functionh. It enabled us to achieve a novel design principle: *When message is changed, the calculation will be different.* It make the system can resistant against all extant attacks.

Key words: Cryptographic hash function, SHA, Dynamic SHA-2

1 Introduction

The SHA-2 family of hash functions was designed by NSA and adopted by NIST in 2000 as a standard that is intended to replace SHA-1 in 2010 [6]. Since MD5, SHA-0 and SHA-1 was broght out, people has not stop attacking them, and they succeed. Such as: den Boer and Bosselaers [2,3] in 1991 and 1993, Vaudenay [8] in 1995, Dobbertin [5] in 1996 and 1998, Chabaud and Joux [4] in 1998, Biham and Chen [1] in 2004, and Wang et al. [9–12] in 2005. Most well known cryptographic hash functions such as: MD4, MD5, HAVAL, RIPEMD, SHA-0 and SHA-1, have succumbed to those attacks.

Since the developments in the field of cryptographic hash functions, NIST decided to run a 4 year hash competition for selection of a new cryptographic hash standard [7]. And the new cryptographic hash standard will provide message digests of 224, 256, 384 and 512-bits.

In those attack, we can find that when different message inputed, the operation in the hash function is no change. If message space is divide many parts, in different part, the calculation is different, the attacker will not know the relationship between message and hash value. The hash function will be secure. To achieve the purpose, Dynamic SHA-2 use bits in message as parameter of function G, R and ROTR operation to realize the principle.

My Work: By introducing a novel design principle in the design of hash functions, and by using components from the SHA-2 family, I describe the design of a new family of cryptographic hash functions called Dynamic SHA-2. The principles is:

When message is changed, the calculation will be different.

The principle combined with the already robust design principles present in SHA-2 enabled us to build a compression function of Dynamic SHA-2 that has the following properties:

1. There is not message expansion part.
2. The iterative part include 8 rounds.

2 Preliminaries and notation

In this paper I will use the same notation as that of NIST: FIPS 180-2 description of SHA-2 [6].

The following operations are applied to 32-bit or 64-bit words in Dynamic SHA-2:

1. Bitwise logical word operations:‘$\wedge$’–AND ,‘$\vee$’–OR,‘$\oplus$’–XOR and ‘$\neg$’–Negation.
2. Addition ‘+’ modulo 2^{32} or modulo 2^{64}.
3. The shift right operation, $SHR^n(x)$, where x is a 32-bit or 64-bit word and n is an integer with 0≤n<32 (resp. 0≤n<64).
4.The shift left operation, $SHL^n(x)$, where x is a 32-bit or 64-bit word and n is an

integer with 0≤n<32 (resp. 0≤n<64).
5. The rotate right (circular right shift) operation, $ROTR^n(x)$, where x is a 32-bit or 64-bit word and n is an integer with 0 ≤ n < 32 (resp. 0 ≤ n < 64).
6. The rotate left (circular left shift) operation, $ROTL^n(x)$, where x is a 32-bit or 64-bit word and n is an integer with 0 ≤ n < 32 (resp. 0 ≤ n < 64).

Depending on the context I will sometimes refer to the hash function as Dynamic SHA-2, and sometimes as Dynamic SHA-224/256 or Dynamic SHA-384/512.

2.1 Fuctions

Dynamic SHA-2 include three functions. The functions are used in compression function.

2.1.1 Function G(x1,x2, x3,t)

Function G operates on three words x1,x2, x3 and an integer t, produces a word y as output. And funcion G as follow:

$$y = G_t(x1, x2, x3) = \begin{cases} x1 \oplus x2 \oplus x3 & t = 0 \\ (x1 \wedge x2) \oplus x3 & t = 1 \\ (\neg(x1 \vee x3)) \vee (x1 \wedge (x2 \oplus x3)) & t = 2 \\ (\neg(x1 \vee (x2 \oplus x3))) \vee (x1 \wedge \neg x3) & t = 3 \end{cases}$$

Table 1.1. functions G for Dynamic SHA-2

x1	x2	x3	f1	f2	f3	f4
0	0	0	0	0	1	1
0	0	1	1	1	0	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	1
1	1	1	1	0	0	0

Table 1.2. truth table for logical functions

2.1.2 Function R(x1,x2,x3,x4,x5,x6,x7,x8,t)

Function R operates on eight words x1,x2, x3,x4, x5,x6, x7,x8 and an integer t. Produces one word y as output. Function R as follow:

$$y = ROTR^t((((((x1 \oplus x2) + x3) \oplus x4) + x5) \oplus x6) + x7) \oplus x8$$

2.1.3 Function COMP(hv1,hv2, …,hv8,w(1),w(2),…,w(8),t)

Function ME1 operates on sixteen words hv1,hv2, …,hv8,w(1),w(2),…,w(8) and an integer t. Function COMP is defined as table 2.

2.2 Dynamic SHA-2 Constants

Dynamic SHA-2 does not use any constants.

2.3 Preprocessing

Preprocessing in Dynamic SHA-2 is exactly the same as that of SHA-2. That means that these three steps: padding the message M, parsing the padded message

Dynamic SHA-224/256	$T = R(hv1, hv2, hv3, hv4, hv5, hv6, hv7, hv8, w(t) \wedge 31)$ $hv8 = hv7$ $hv7 = ROTR^{(SHR^5(w(t)))\wedge 31}(hv6)$ $hv6 = hv5 + w((t+3) \wedge 7)$ $hv5 = ROTR^{(SHR^{10}(w(t)))\wedge 31}(hv4)$ $hv4 = G(hv1, hv2, hv3, SHR^{30}(w(t))) + w((t+2) \wedge 7)$ $hv3 = hv2$ $hv2 = hv1$ $hv1 = T + w((t+1) \wedge 7)$ $T = R(hv1, hv2, hv3, hv4, hv5, hv6, hv7, hv8, (SHR^{15}(w(t))) \wedge 31)$ $hv8 = hv7 + w((t+7) \wedge 7)$ $hv7 = ROTR^{(SHR^{20}w(t))\wedge 31}(hv6)$ $hv6 = hv5 + w((t+6) \wedge 7)$ $hv5 = ROTR^{(SHR^{25}w(t))\wedge 31}(hv4)$ $hv4 = G(hv1, hv2, hv3, t \wedge 3) + w((t+5) \wedge 7)$ $hv3 = hv2 + w(t)$ $hv2 = hv1$ $hv1 = T + w((t+4) \wedge 7)$
Dynamic SHA-384/512	$T = R(hv1, hv2, hv3, hv4, hv5, hv6, hv7, hv8, w(t) \wedge 63)$ $hv8 = hv7$ $hv7 = ROTR^{(SHR^6(w(t)))\wedge 63}(hv6)$ $hv6 = ROTR^{(SHR^{12}(w(t)))\wedge 63}(hv5) + w((t+3) \wedge 7)$ $hv5 = ROTR^{(SHR^{18}(w(t)))\wedge 63}(hv4)$ $hv4 = G(hv1, hv2, hv3, SHR^{62}(w(t))) + w((t+2) \wedge 7)$ $hv3 = ROTR^{(SHR^{24}(w(t)))\wedge 63}(hv2)$ $hv2 = hv1$ $hv1 = T + w((t+1) \wedge 7)$ $T = R(hv1, hv2, hv3, hv4, hv5, hv6, hv7, hv8, (SHR^{30}(w(t))) \wedge 63)$ $hv8 = hv7 + w((t+7) \wedge 7)$ $hv7 = ROTR^{(SHR^{36}(w(t)))\wedge 63}(hv6)$ $hv6 = ROTR^{(SHR^{42}(w(t)))\wedge 63}(hv5) + w((t+6) \wedge 7)$ $hv5 = ROTR^{(SHR^{48}(w(t)))\wedge 63}(hv4)$ $hv4 = G(hv1, hv2, hv3, (SHR^{60}(w(t))) \wedge 3) + w((t+5) \wedge 7)$ $hv3 = hv2 + w(t)$ $hv2 = ROTR^{(SHR^{54}(w(t)))\wedge 63}(hv1)$ $hv1 = T + w((t+4) \wedge 7)$

Table 2. functions COMP for Dynamic SHA-2

intomessage blocks, and setting the initial hash value, H^0 are the same as in SHA-2. Thus in the parsing step the message is parsed into N blocks of 512 bits (resp. 1024 bits), and the i-th block of 512 bits (resp. 1024 bits) is a concatenation of sixteen 32-bit (resp. 64-bit) words denoted as $M_0^{(i)}, M_1^{(i)}, ..., M_{15}^{(i)}$.

Dynamic SHA-2 may be used to hash a message, M, having a length of l bits, where $0 \le l < 2^{64}$.

2.3.1 padding

Suppose that the length of the message M is L bits. Append the bit "1" to the end of themessage, followed by k zero bits, where k is the smallest, non-negative solution to the equation L+1+k ≡ 448 mod 512 (resp. L+1+k ≡ 960 mod 1024). Then append the 64-bit block that is equal to the number L expressed using a binary representation.

2.4 Initial Hash Value H^0

The initial hash value, H^0 for Dynamic SHA is the same as that of SHA-2 (given in Table 3.1).

Dynamic SHA-224	Dynamic SHA-256	Dynamic SHA-384	Dynamic SHA-512
$H_0^{(0)} = c1059ed8,$	$H_0^{(0)} = 6a09e667,$	$H_0^{(0)} = cbbb9d5dc1059ed8,$	$H_0^{(0)} = 6a09e667f3bcc908,$
$H_1^{(0)} = 367cd507,$	$H_1^{(0)} = bb67ae85,$	$H_1^{(0)} = 629a292a367cd507,$	$H_1^{(0)} = bb67ae8584caa73b,$
$H_2^{(0)} = 3070dd17,$	$H_2^{(0)} = 3c6ef372,$	$H_2^{(0)} = 9159015a3070dd17,$	$H_2^{(0)} = 3c6ef372fe94f82b,$
$H_3^{(0)} = f70e5939,$	$H_3^{(0)} = a54ff53a,$	$H_3^{(0)} = 152fecd8f70e5939,$	$H_3^{(0)} = a54ff53a5f1d36f1,$
$H_4^{(0)} = ffc00b31,$	$H_4^{(0)} = 510e527f,$	$H_4^{(0)} = 67332667ffc00b31,$	$H_4^{(0)} = 510e527fade682d1f,$
$H_5^{(0)} = 68581511,$	$H_5^{(0)} = 9b05688c,$	$H_5^{(0)} = 8eb44a8768581511,$	$H_5^{(0)} = 9b05688c2b3e6c1f,$
$H_6^{(0)} = 64f98fa7,$	$H_6^{(0)} = 1f83d9ab,$	$H_6^{(0)} = db0c2e0d64f98fa7,$	$H_6^{(0)} = 1f83d9abfb41bd6b,$
$H_7^{(0)} = befa4fa4,$	$H_7^{(0)} = 5be0cd19,$	$H_7^{(0)} = 47b5481dbefa4fa4,$	$H_7^{(0)} = 5be0cd19137e2179,$

Table 3.1. The initial hash value, H^0 for Dynamic SHA

For i = 1 to N:

{

1.Initialize eight working variables a, b, c, d, e, f, g and h with the $(i-1)^{th}$ hash value:

$$a = H_0^{(i-1)}, \quad b = H_1^{(i-1)}, \quad c = H_2^{(i-1)}, \quad d = H_3^{(i-1)},$$
$$e = H_4^{(i-1)}, \quad f = H_5^{(i-1)}, \quad g = H_6^{(i-1)}, \quad h = H_7^{(i-1)}$$

2. Iterative part

For t=0 to 7

{

$$COMP(a,b,c,d,e,f,g,h,w_0,w_1,w_2,w_3,w_4,w_5,w_6,w_7,t)$$
$$COMP(a,b,c,d,e,f,g,h,w_8,w_9,w_{10},w_{11},w_{12},w_{13},w_{14},w_{15},t)$$

}

3.Compute the i^{th} intermediate hash value $H^{(i)}$:

$$H_0^{(i)} = a + H_0^{(i-1)}, \quad H_1^{(i)} = b + H_1^{(i-1)}, \quad H_2^{(i)} = c + H_2^{(i-1)}, \quad H_3^{(i)} = d + H_3^{(i-1)},$$
$$H_4^{(i)} = e + H_4^{(i-1)}, \quad H_5^{(i)} = f + H_5^{(i-1)}, \quad H_6^{(i)} = g + H_6^{(i-1)}, \quad H_7^{(i)} = h + H_7^{(i-1)}$$

}

Table 3.2. Algorithmic description of Dynamic SHA-2 hash function.

2.5 Dynamic SHA-2 Hash Computation

The Dynamic SHA-2 hash computation uses functions and initial values defined in previous subsections. So, after the preprocessing is completed, each message block, $M^{(0)}, M^{(1)}, ..., M^{(N)}$, is processed in order, using the steps described algorithmically in Table 3.2.

The algorithm uses 1) a message schedule of forty-eight 32-bit (resp. 64-bit) words, 2) eight working variables of 32 bits (resp. 64 bits) , and 3) a hash value of eight 32-bit (resp. 64-bit) words. The final result of Dynamic SHA-256 is a 256-bit message digest and of Dynamic SHA-512 is a 512-bit message digest. The final result of Dynamic SHA-224 and Dynamic SHA-384 are also 256 and 512 bits, but the output is then truncated as in SHA-2 to 224 (resp. 384 bits). The words of the message schedule are labeled $W_0, W_1, ..., W_{47}$. The eight working variables are labeled a, b, c, d, e, f, g and h and sometimes they are called "state register". The words of the hash value are labeled $H_0^{(i)}, H_1^{(i)}, ..., H_7^{(i)}$, which will hold the initial hash value, $H^{(0)}$, replaced by each successive intermediate hash value (after each message block is processed), $H^{(i)}$, and ending with the final hash value, $H^{(N)}$.

Dynamic SHA-2 also uses one temporary words T.

3 Security of Dynamic SHA-2

In this section I will make an initial analysis of how strongly collision resistant, preimage resistant and second preimage resistant Dynamic SHA-2 is. I will start by describing our design rationale, then I will analyze the properties of the message expansion part and finally I will discuss the strength of the function against known attacks for finding different types of collisions.

3.2 Properties of iterative part

In the iterative part, there are functions G, MRN, R and some ROTR operations, in one round, sixteen message words will be used.

3.3 Design rationale

The reasons for principle: *When message is changed, the calculation will be different.*

From the definition of function G, R and ROTR operations, it is easy to know all bits in message has been used as parameters of function G, R and ROTR operation. One bit different in message, different logical function or different ROTR operation will be done, it will make the calculation different. Different message will lead to different calculation, these different calculations divide message space into 2^{512} (resp. 2^{1024}) parts. In a part there is $2^{512-512}=1$ (resp. $2^{1024-1024}=1$) message value.

Why Dynamic SHA-2 does not have constants?

The reasons why I decided not to use any constants is that Dynamic SHA-2 is secure enough.

Controlling the differentials is hard in Dynamic SHA-2:

In Dynamic SHA-2, the message space is divided into 2^{512} (resp. 2^{1024}) parts. In different part, the calculation is different. In a part, there is only one message value. It can not find collisions in the same part.

It is hard to use linera function describe function G, R and data-depend ORTR operation. It can use Algebraic Normal Form (ANF) to represent Dynamic SHA-2, but the ANFs that represent function R has up to, 2^{261} (resp. 2^{518}) monomials.

3.4Finding Preimages of Dynamic SHA-2

To a hash function f(·), it need satisfy:

Given hash value H=f(M), it is hard to find message M that meet H=f(M).

There are three ways to find preimages of a hash function:

1,From the definition of Dynamic SHA-2 (similarly as with SHA-2) it follows that

from a given hash digest it is possible to perform backward iterative steps by guessing values that represent some relations between working variables of the extension part.

To do this, it need the parameter of the ROTR operation and function G, R in Dynamic SHA-2. But in Dynamic SHA-2, when message changed, the parameter of the ROTR operation and function G, R will changed. So attacker had to gusee the parameter that will be used in Dynamic SHA-2. From the definition of Dynamic SHA-2, it is know that all bits in message are used as the parameter of the ROTR operation and function G, R. When attacker complete guessing parameters, he has guessed all bits in message.

2, The probability of random guess of finding preimages is 2^{-224} (resp. 2^{-256}, 2^{-384}, 2^{-512}).

3.5Finding Second Preimages of Dynamic SHA-2

To a hash function f(·), it need satisfy:

Given M, it is hard to find M' s.t. f(M) = f(M').

There are three ways to find Second Preimages of a hash function:

1, Get hash value H of message M, and find different message M' that has hash value H. then the problem become find Preimages of the hash function.

2, Given M, and find out the relationship between the difference △M=(M1-M) and the difference △H=f(M1)-f(M). And find out △M≠0 that make △H=0. To do this, someone will set up some system of equations obtained from the definition of the hash function, then trace forward and backward some initial bit differences that will result in fine tuning and annulling of those differences and finally obtain Second Preimages. It need know the unchangeable formulas that represent hash function f. In Dynamic SHA-2, when message is changed, the calculation is different. To get unchangeable formulas that represent hash function f, it need get ANFs for Dynamic SHA-2. And the ANFs that represent function R has up to 2^{261} (resp. 2^{518}) monomials

3. The probability of random guess of finding preimages is 2^{-224} (resp. 2^{-256}, 2^{-384}, 2^{-512}).

3.6 Finding Collisions in Dynamic SHA-2

To a hash function f(·), it need satisfy:

It is hard to find different M and M' s.t. f(M) = f (M').

There are three ways to find collisions of a hash function:

1, Fix message M, and find different message M' that has hash value H=f(M). then the problem become find Second Preimages of the hash function.

2. Find out the relationship between the (M, M') and the difference △H=f(M)-f(M'). And find out (M,M') that make △H=0. To do this, someone will set up some system of equations obtained from the definition of the hash function, then trace forward and backward some initial bit differences that will result in fine tuning and annulling of those differences and finally obtain collisions. It need know the unchangeable formulas that represent hash function f. In Dynamic SHA-2, when message is changed, the calculation is different. To get unchangeable formulas that represent hash function f, it need get ANFs for Dynamic SHA-2. And the ANFs that represent function R has up to 2^{261} (resp. 2^{518}) monomials

3. The attack base on the birthday paradox. the workload for birthday attack is of O(2^{112}) (resp. O(2^{128}) O(2^{192}) O(2^{256})).

3.7 Finding collisions in the reduced compression function of Dynamic SHA-2

If there are 1 round, the bits in message are mixed one times, the system will be weak, someone can backward Dynamic SHA-2 as table 5 show.

If there are more than 2 rounds, the bits in message are mixed at least twice, if attacker backward Dynamic SHA-2 as table 5 show, he will have a system of 32 equation with 16 unknow variables, The probability of there is solution for the system is 2^{-512} (resp. 2^{-1024}). And the message space is divided into 2^{128} (resp. 2^{256}) parts, in a part, there are 2^{384} (resp. 2^{768}) message values. The average number of collisions is 2^{160} (resp. 2^{128}, 2^{384}, 2^{256}). If an algorithm is developed to find collision for a calculation, then the probability of find the collision is 2^{-128} (resp. $2^{-128}, 2^{-256}, 2^{-256}$).

4 Improvement

There are some improvements for Dynamic SHA-2:

1.There is no any constants in Dynamic SHA-2. Use constants will increase system security.

2. In HMAC, the initial hash value is random variable. If Dynamic SHA-2 is used in HMAC, by theorem 2, it is easy know that the probability of hash value is 2^{-224} (resp. 2^{-256} 2^{-384} 2^{-512}).

There are some ways that we can adopt, for example: $IV_i = IV_{i-1} + c$, IV_i is i-th initial hash value, c is constant and c is odd number. And this way is easy to be attacked by "Man In The Middle" attack.

5 Conclusion

In the paper[1-5] and [8-12]. People has successfully attack SHA-2 family, in these attacks, people had use the fact that they can analyse what will happen at some bits in hash value when some bits in message changed.

If we can design a hash algorithm that when message change the calculation will be different. It will be hard to analyse the relationship between message and hash value. the system will be secure.

And based on components from the family SHA-2. I have introduced the principle in the design of Dynamic SHA-2: *When message is changed, the calculation will be different.* And I bring in data depend function G, R, and use bits in message as parameters of function G, R and ROTR operations. These steps realize the principle.

Function G, R and ROTR operations divided the message space into manys parts, in different part, the calculation is different. At the same time, the ANFs for function R have huge number monomials. So attacker has two choices: deal with a big formula or deal with divided message space. If attacker select big formula, he had to deal with formulaS that ANFs has up to 2^{261} (resp. 2^{518}) monomials. If attacker select divided message space, the message space is divided into 2^{512} (resp. 2^{1024}) parts, in a part, there is only one message value.

The principle enabled us to build a compression function of Dynamic SHA-2 that has not new variable, the iterative part has 8 rounds, it is more robust and resistant against generic multi-block collision attacks, it is resistant against generic length extension attacks.

References

1. E. Biham and R. Chen, "Near-collisions of SHA-0," Cryptology ePrint Archive, Report 2004/146, 2004. http://eprint.iacr.org/2004/146
2. B. den Boer, and A. Bosselaers: "An attack on the last two rounds of MD4", CRYPTO 1991, LNCS, 576, pp. 194-203, 1992.
3. B. den Boer, and A. Bosselaers: "Collisions for the compression function of MD5",

EUROCRYPT 1993, LNCS 765, pp. 293-304, 1994.
4. F. Chabaud and A. Joux, “Differential collisions in SHA-0,” Advances in Cryptology, Crypto98, LNCS, vol.1462, pp.56-71, 1998.
5. H. Dobbertin: “Cryptanalysis of MD4”, J. Cryptology 11, pp. 253-271, 1998.
6. NIST, Secure Hash Signature Standard (SHS) (FIPS PUB 180-2), United States of American, Federal Information Processing Standard (FIPS) 180-2, 2002 August 1.
7. NIST Tentative Timeline for the Development of New Hash Functions, http://csrc.nist.gov/groups/ST/hash/timeline.html
8. S. Vaudenay, “On the need for multipermutations: Cryptanalysis of MD4 and SAFER”, Fast Software Encryption- FSE95, LNCS 1008, pp. 286–297, 1995.
9. X. Wang, X. Lai, D. Feng, H. Chen and X. Yu, “Cryptanalysis of the Hash Functions MD4 and RIPEMD”, EUROCRYPT 2005, LNCS 3494, pp. 1–18, 2005.
10. X. Wang and H. Yu , “How to Break MD5 and Other Hash Functions”, EUROCRYPT 2005, LNCS 3494, pp. 19–35, 2005.
11. X. Wang, H. Yu, Y. L. Yin “Effcient Collision Search Attacks on SHA-0”, CRYPTO 2005, LNCS 3621, pp. 1–16, 2005.
12. X. Wang, Y. L. Yin, H. Yu, “Collision Search Attacks on SHA-1”, CRYPTO 2005, LNCS 3621, pp. 17–36, 2005.
13. Gupta and Sarkar “Computing Walsh Transform from the Algebraic Normal Form of a Boolean Function” http://citeseer.ist.psu.edu/574240.html

Appendix 1: Constitute Boolean functions to represent function.

We can use Algebraic Normal Form (ANF) to represent function. Gupta and Sarkar[13] have studied it.

Let n≥r≥1 be integers and let $F:\{0,1\}^n \to \{0,1\}^r$ be a vector valued Boolean function. The vector valued function F can be represented as an r-tuple of Boolean functions $F=(F^{(1)},F^{(2)},...,F^{(r)})$, where $F^{(s)}:\{0,1\}^n \to \{0,1\}(s=1,2,...,r)$, and the value of $F^{(s)}(x_1,x_2,...,x_n)$ equals the value of the s-th component of $F(x_1,x_2,...,x_n)$. The Boolean functions $F^{(s)}(x_1,x_2,...,x_n)$ can be expressed in the Algebraic Normal Form (ANF) as polynomials with n variables $x_1,x_2,...,x_n$ of kind $a_0 \oplus a_1x_1 \oplus ... \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus ... \oplus a_{n-1,n}x_{n-1}x_n \oplus ... \oplus a_{1,2,...,n}x_1,x_2,...,x_n$,where $a_\lambda \in \{0,1\}$. Each ANF have up to 2^n monomials, depending of the values of the coefficients a_λ.

Function R

Function R operates on six words x1,x2,x3,x4,x5,x6,x7,x8 and an integer t and produces a word y as output, where $0 \le t < w$. So we have $R:\{0,1\}^{8\times w+\log_2^w} \to \{0,1\}^w$, It is easy to know that one-bit different in words x1,x2,x3,x4,x5,x6,x7,x8. Because the parameter of the rotate right operation is depend on message, with differen message different rotate right operation will be done. So the bit in output maybe changed.

So the ANFs to represent function R have up to $2^{8\times w} \times w$ monomials, where w is bit length of the word.

Appendix 2: Function G and Function R

Let $p(x)$ is probability of x.

1, **Function G:**
Function y=G(x1,x2,x3,t) operates on tree words x1,x2,x3 and an integer t, $0 \le t \le 3$. Function G use the integer t select a logical funcion from f_0, f_1, f_2, f_3. And y, x1, x2, x3 are w-bit word. So the bit-length of (x1,x2,x3,t) is $3 \times w+2$, the bit-length of y is w.

To a given value y'=G(x1,x2,x3,t), there is $2^{2 \times w+2}$ 4-tuple (y',x1,x2,t). To a given 4-tuple (y',x1',x2', t'). there is the relation:

$$x4' = \begin{cases} x1' \oplus x2' \oplus y' & t=0 \\ (x1' \wedge x2') \oplus y' & t=1 \\ (\neg(x1' \vee y')) \vee (x1' \wedge (x2' \oplus y')) & t=2 \\ (\neg(x1' \vee (x2' \oplus y'))) \vee (x1' \wedge \neg y') & t=3 \end{cases}$$

To given 4-tuple (y',x1',x2',t'), it can compute the value for x3'. So there are $2^{2 \times w+2}$ 4-tuple (x1,x2,x3,t) have the same value y'. x1, x2, x3, t are random and uncorrelated variable, there is:
$p(x1) = p(x2) = p(x3) = 2^{-w}$ and $p(t) = 2^{-2}$

$$p(y) = \sum_{i1=0}^{2^w-1} \sum_{i2=0}^{2^w-1} \sum_{i3=0}^{2^w-1} \sum_{i4=0}^{3} p(y \mid (x1_{i1}, x2_{i2}, x3_{i3}, t_{i4})) \times p(x1_{i1}) \times p(x2_{i2}) \times p(x3_{i3}) \times p(t_{i4})$$

$$p(y) = p(x1_{i1}) \times p(x2_{i2}) \times p(x3_{i3}) \times p(t_{i4}) \times \sum_{i1=0}^{2^w-1} \sum_{i2=0}^{2^w-1} \sum_{i3=0}^{2^w-1} \sum_{i4=0}^{3} p(y \mid (x1_{i1}, x2_{i2}, x3_{i3}, t_{i4}))$$

$$p(y) = 2^{-w} \times 2^{-w} \times 2^{-w} \times 2^{-2} \times 2^{2 \times w+2} = 2^{-w}$$

If x1, x2, x3, t are random and uncorrelated, function G will produce random word and

$$p(y) = 2^{-w}$$

2, **Function R:**
Function y=R(x1,x2,x3,x4,x5,x6,x7,x8,t) operates on eight words x1,x2,x3,x4,x5,x6, x7, x8 and an ineger t. To a given value y'=R(x1,x2,x3,x4,x5,x6,x7,x8,t), there is $2^{7 \times w} \times w$ 9-tuple (y',x1',x2',x3',x4',x5',x6',x7',t'). To a given 9-tuple (y',x1',x2',x3',x4', x5', x6', x7',t'). there is the relation:

$$x8 = ROTR^{t'}((((((x1' \oplus x2') + x3') \oplus x4') + x5') \oplus x6') + x7') \oplus y'$$

To given 9-tuple (y',x1',x2',x3',x4',x5',x6',x7',t'), it can compute the value for x8, So there are $2^{7 \times w} \times w$ 9-tuple (y',x1',x2',x3',x4',x5',x6',x7',t') have the same value y'. x1,x2,x3,x4,x5,x6,x7,x8,t are random and uncorrelated variable, there is:

$$p(x1) = p(x2) = p(x3) = p(x4) = p(x5) = p(x6) = p(x7) = p(x8) = 2^{-w}$$

$$p(t) = w^{-1}$$

$$p(y) = \sum_{i1=0}^{2^w-1} \sum_{i2=0}^{2^w-1} \cdots \sum_{i8=0}^{2^w-1} \sum_{i9=0}^{w-1} p(y \mid (x1_{i1}, x2_{i2}, \ldots, x8_{i8}, t_{i9})) \times$$
$$\times p(x1) \times p(x2) \times p(x3) \times p(x4) \times p(x5) \times p(x6) \times p(x7) \times p(x8) \times p(t)$$

$$p(y) = 2^{-w} \times 2^{-w} \times 2^{-w} \times 2^{-w} \times 2^{-w} \times 2^{-w} \times 2^{-w} \times w^{-1} \times 2^{7 \times w} \times w = 2^{-w}$$

If x1,x2,x3,x4,x5,x6,x7,x8, t are random and uncorrelated, function R will produce random word and $p(y) = 2^{-w}$

It is hard to describe Dynamic SHA-2 with linear funcction

It is hard to describe data depend fuction $z = ROTR^{x}y$, with linear funcction, the ANF is a way, but the ANFs for function R has up to $2^{8\times w+\log_2^w}$ monomials. The ANFs that describe data depend fuction has more monomials than the ANFs that describe other function. It will be harder to analyse data depend fuction by differential analysis.

Avalanche of Dynamic SHA-2.

After first round, all bits in message has been mixed. From the definition of function R, it is easy to know that one bit different will lead to different ROTR operation been done. and after call function R 8 times, all working variables were affect. In one roundafter, function R will be called 4 times. There are 7 rounds after first round, function R will be called $4\times 7 = 28$ times after first round. When one bit changed in message, all bit in last hash value maybe changed.

Appendix 3: Some thing about Dynamic SHA-2

1.Function G and function R

As mentioned at appendix 2, if the variables are random word, function G and R will produced random word.

Function G operates on three words and an integer t, produces a word as output. What function G do is produce confusion word.

Function R operates on eight words and an integer t, produces a word as output. What function R do is confuse the place of bits.

2. Why Dynamic SHA-2 use function G and function R

As mentioned in Appendix 2, when the variable is random word, function G and function R will produce random word.

And function R is data-depend fuction, the ANFs that describe function R has has up to $2^{8\times w+\log_2^w}$ monomials, this mak it hard to analyse the relationship between message and hash value with differential analysis.

3. there is 8 rounds in Dynamic SHA-2

From the definition of Dynamic SHA-2, it is easy to know that after 8 rounds, all bits in message had been used as parameter of function G, R and ROTR operations. And the message value space had been divided into 2^{512} (resp. 2^{1024}) parts, in a part, there is only one message value.

From the definition of function COMP, it is easy to backward function COMP. If the message is mixed only one time, the system will be weak. In 8 rounds, the message bits are mixed 8 times, if attacker backward Dynamic SHA-2 as table 5 show, he will have a system of 128 equation with 16 unknow variables. The probability of there exist solution for the system is $2^{-7\times512}$ (resp. $2^{-7\times1024}$). Or attacker can use random guessing. The probability of random guess of finding preimages is 2^{-224} (resp. 2^{-256}, 2^{-384}, 2^{-512}).

Appendix 4: Spreading of Dynamic SHA

To simplification, Let:

1.MW1=(W(0),W(1),W(2),W(3),W(4),W(5),W(6),W(7)),
MW2=(W(8),W(9),W(10),W(11),W(12),W(13),W(14),W(15))
W(j) is the message word.

2. hv(i)=(a(i), b(i), c(i), d(i),e(i), f(i), g(i), h(i)). where a(i), b(i), c(i), d(i),e(i), f(i), g(i), h(i) are working variables at i-th function COMP called.

3. $H_i(\mathrm{hv}(-1), \mathrm{MW1}, \mathrm{MW2}) = \mathrm{hv}(i) \quad 1 \le i \le 15$

4. Message word and working variables are b-bit words.

From the definition of Dynamic SHA-2, it is easy know that function COMP had been called sixteen times, when function COMP is called, MW1 or MW2 will be mixed. So it can describe Dynamic SHA-2 as follow:

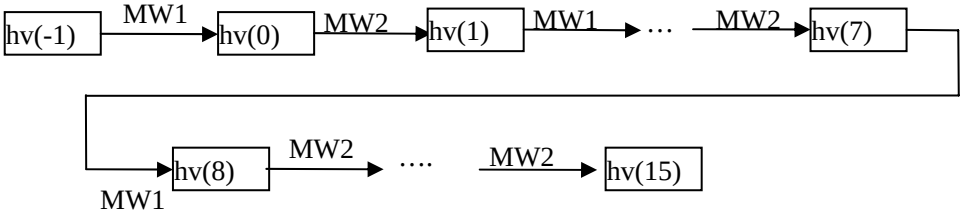

Table 4 data processing of Dynamic SHA-2

At first there are two theorems:

Theorem 1:

To function $COMP(a,b,c,d,e,f,g,h,w0,w1,w2,w3,w4,w5,w6,w7,t)$ *, there is:*
1.MW=(W0,W1,W2,W3,W4,W5,W6,W7), where W0,…,W7 are words that mixed.
2. hva=(a0, b0, 0c, d0, e0, 0f, g0, h0). Where a0, b0, c0, d0,e0, f0, g0, h0 are working variables that before call function COMP.
3. hvb=(a1, b1, c1, d1, e1, f1, g1, h1). Where a1, b1, c1, d1,e1, f1, g1, h1 are working variables that after call function COMP.
working variables are b-bit word. hva, MW are random and uncorrelated.

Then there exist:

(1),p(hvb)= $2^{-8\times b}$
(2),p(hvb|MW)= $2^{-8\times b}$
(3),p(hvb|hva)= $2^{-8\times b}$

Proof.

The integer t in function COMP is decided by which round function COMP be. So The integer t can be look as constant. And we can use function $hvb = F(hva, MW)$ describe function COMP. And we have $F:\{0,1\}^{16\times b} \to \{0,1\}^{8\times b}$. hva, MW are random and uncorrelated. So there is p(hva)= $2^{-8\times b}$ and p(MW)= $2^{-8\times b}$

There are $2^{8\times b}$ MW. To a given MW', there exist:

To a given hva', frome the definition of F, there is only a hvb that make $hvb = F(hva', MW')$.

And to a given hvb', it can backward function F, and there is only a hva that make $hvb' = F(hva, MW')$. So there exist:

$$p(hvb) = \sum_{i1=0}^{2^b-1} \sum_{i2=0}^{2^b-1} p(hvb \mid (hva(i1), MW(i2))) \times p(hva(i1)) \times p(MW(i2))$$

$$p(hvb) = p(hva) \times p(MW) \times \sum_{i1=0}^{2^b-1} \sum_{i2=0}^{2^b-1} p(hvb \mid (hva(i1), MW(i2)))$$

$$p(hvb) = 2^{-8\times b} \times 2^{-8\times b} \times 2^{8\times b} = 2^{-8\times b}$$

$$p(hvb \mid MW) = \sum_{i1=0}^{2^b-1} p((hvb \mid MW) \mid hva(i1)) \times p(hva(i1))$$

$$p(hvb \mid MW) = p(hva) \times \sum_{i1=0}^{2^b-1} p((hvb \mid MW) \mid hva\ (i1))$$

$$p(hvb \mid MW) = 2^{-8\times b}$$

Dynamic SHA-224/256	$w0 = c1 - a0$ $w5 = d1 - G(b1, a0, b0, t \wedge 3)$ $w6 = f1 - ROTR^{32-(SHR^{10}w0)\wedge 31}(d0)$ $w7 = h1 - ROTR^{32-(SHR^{5}w0)\wedge 31}(f0)$ $w3 = ROTR^{32-(SHR^{20}w0)\wedge 31}(g1) - e0$ $w2 = ROTR^{32-(SHR^{25}w0)\wedge 31}(e1) - G(a0, b0, c0, SHR^{30}w0)$ $w1 = b1 - R(a0, b0, c0, d0, e0, f0, g0, h0, w0 \wedge 31)$ $d' = ROTR^{32-(SHR^{25}w0)\wedge 31}(e1)$ $e' = f1 - w6$ $f' = e0 + w3$ $g' = ROTR^{32-((SHR^{5}w0)\wedge 31)}(e)$ $w4 = a1 - R(b1, a0, b0, d', e', f', g', g0, (SHR^{15}(w0)) \wedge 31)$
Dynamic SHA-384/512	$w0 = c1 - a0$ $a' = ROTR^{64-((SHR^{54}w0)\wedge 63)}b1$ $c' = ROTR^{64-((SHR^{24}w0)\wedge 63)}b0$ $w5 = d1 - G(a', a0, c', t \wedge 3)$ $w6 = f1 - ROTR^{128-((SHR^{42}w0)\wedge 63)-((SHR^{18}w0)\wedge 63)}(d0)$ $w7 = h1 - ROTR^{64-(SHR^{6}w0)\wedge 63}(f0)$ $w3 = ROTR^{128-((SHR^{36}w0)\wedge 63)-((SHR^{12}w0)\wedge 63)}(g1) - e0$ $w2 = ROTR^{64-(SHR^{48}w0)\wedge 31}(e1) - G(a0, b0, c0, SHR^{62}w0)$ $w1 = ROTR^{64-(SHR^{54}w0)\wedge 31}(b1) - R(a0, b0, c0, d0, e0, f0, g0, h0, w0 \wedge 63)$ $d' = ROTR^{64-(SHR^{48}w0)\wedge 63}(e1)$ $e' = ROTR^{64-(SHR^{18}w0)\wedge 63}(d0)$ $f' = ROTR^{64-(SHR^{36}w0)\wedge 63}(g1)$ $g' = ROTR^{64-((SHR^{6}w0)\wedge 63)}(f0)$ $w4 = a1 - R(a', a0, c', d', e', f', g', g0, (SHR^{15}(w0)) \wedge 31)$

Table 5. relationship of hva, hvb

(3)

To given hva', there exist:

To a given hvb', there is the relationship as table 5, It is easy to compute the value for MW that make $hvb' = F(hva', MW)$. So there exist:

$$p(hvb \mid hva) = \sum_{i=0}^{2^b-1} p((hvb \mid hva) \mid MW(i)) \times p(MW(i)) = 2^{-8\times b} = 2^{-8\times b} \qquad \square$$

By theorem 1, To function COMP, it is easy know that:

To a given hva', mix different message words MW, the hvb will be different.

Mix given message words MW', if the hva is different, the hvb will be different.

Theorem 2. *In Dynamic SHA-2, there exist:*

(1) p(hv(j))= $2^{-8\times b}$

(2),p(hv(j)|MW1)= $2^{-8\times b}$

(3),p(hv(j)|MW2)= $2^{-8\times b}$

$j=1,....,15$

Proof.

hv(-1), MW1 and MW2 are random and uncorrelated, so there exist:

p(hv(-1)) = $2^{-8\times b}$

p(MW1) = $2^{-8\times b}$

p(MW2) = $2^{-8\times b}$.

To simplification, Let F(hv(i-1),MW)=hv(i), MW is mixed words MW1 or MW2.

To a given hv(i)' $i=1,....,15$, there are $2^{16\times b}$ 2-tuple(MW1,MW2).

To a given 2-tuple(hv(i)',MW1'), there are $2^{8\times b}$ MW2. To a given 2-tuple (hv(i)',MW2'), there are $2^{8\times b}$ MW1.

To a given 3-tuple(hv(i)',MW1',MW2'), It is easy to backword iterative steps, and it is easy to compute the value for hv(-1), and the hv(-1) make $\mathrm{H_i(hv(-1),MW1',MW2')=hv(i)'}$.

So there exist:

$$p(hv(i))=\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}\sum_{i2=0}^{2^b-1}p(hv(i)\,|\,(hv(-1)_{i0},MW1_{i1},MW2_{i2}))\times p(hv(-1)_{i0},MW1_{i1},MW2_{i2})$$

$$p(hv(i))=p(hv(-1),MW1,MW2)\times\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}\sum_{i2=0}^{2^b-1}p(hv(i)\,|\,(hv(-1)_{i0},MW1_{i1},MW2_{i2}))$$

$$p(hv(i))=2^{-8\times b}\times 2^{-8\times b}\times 2^{-8\times b}\times 2^{8\times b}\times 2^{8\times b}=2^{-8\times b}$$

$$p(hv(i)\,|\,MW1)=\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}p((hv(i)\,|\,MW1)\,|\,(hv(-1)_{i0},MW2_{i1}))\times p(hv(-1)_{i0},MW2_{i1})$$

$$p(hv(i)\,|\,MW1)=p(hv(-1),MW2)\times\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}p((hv(i)\,|\,MW1)\,|\,(hv(-1)_{i0},MW2_{i1}))$$

$$p(hv(i)\,|\,MW1)=2^{-8\times b}\times 2^{-8\times b}\times 2^{8\times b}=2^{-8\times b}$$

$$p(hv(i)\,|\,MW2)=\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}p((hv(i)\,|\,MW2)\,|\,(hv(-1)_{i0},MW1_{i1}))\times p(hv(-1)_{i0},MW1_{i1})$$

$$p(hv(i)\,|\,MW2)=p(hv(-1),MW1)\times\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1}p((hv(i)\,|\,MW2)\,|\,(hv(-1)_{i0},MW1_{i1}))$$

$$p(hv(i)\,|\,MW2)=2^{-8\times b}\times 2^{-8\times b}\times 2^{8\times b}=2^{-8\times b}$$

□

Theorem 3. *In Dynamic SHA-2, to a given hv(-1), there exist:*

(1) p(hv(2)| hv(-1))= $2^{-8\times b}$

Proof. To simplification, Let F(hv(i-1),MW)=hv(i), MW is mixed words MW1 or MW2.

To a given 2-tuple (hv(2)',hv(-1)'), there are $2^{8\times b}$ MW1.

To a given 2-tuple (hv(2)', MW1'), by theorem 1, there is p(hv(2)| MW1) = $2^{-8\times b}$, so to a given 2-tuple(hv(2)', MW1'), there is $2^{-8\times b}\times 2^{8\times b}=1$ hv(1) that make F(hv(1),MW1')= hv(2)'.

To a given 2-tuple (hv(-1)', MW1'), from the definition of function COMP, it is enough to know that there is a hv(0) that make F(hv(-1)',MW1')= hv(0).

To a given 2-tuple (hv(0)', hv(1)'), by theorem 1, there is p(hv(1)| hv(0)) = $2^{-8\times b}$, so to a given 2-tuple(hv(0)', hv(1)'), there is $2^{-8\times b}\times 2^{8\times b}=1$ MW2 that make F(hv(0)', MW2) = hv(1)' .

So there exist:

$$p(hv(2)\,|\,hv(-1)) = \sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1} p((hv(2)\,|\,hv(-1))\,|\,(MW1_{i0}, MW2_{i1}))\times p(MW1_{i0}, MW2_{i1})$$

$$p(hv(2)\,|\,hv(-1)) = p(MW1, MW2)\times\sum_{i0=0}^{2^b-1}\sum_{i1=0}^{2^b-1} p((hv(2)\,|\,hv(-1))\,|\,(MW1_{i0}, MW2_{i1}))$$

$$p(hv(2)\,|\,hv(-1)) = 2^{-8\times b}\times 2^{-8\times b}\times 2^{8\times b} = 2^{-8\times b}$$

□

By theorem 2 and 3, it is know that:

1. When hv(-1) is random variable, the probability of hash value is $2^{-8\times b}$,
2. To a given hv(-1), the probability of different hash value maybe different.

After first round, the bits in message have been mixed, the mixed bits and working variables value are not uncorrelated, it is hard to analyse the probability of hash value. To get better property of spreading, Dynamic SHA-2 adopt ways as follow:

1. When the variable of function COMP is random value. function COMP will produce random value.
2. To reduce the times that message bits mixed, there is no message expansion part in Dynamic SHA-2.