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 maybe 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, I bring in data depend function R 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 maybe 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. The message expansion part has 16 new variables.
2. The iterative part has just 16(resp.27) rounds.
3. The iterative part has three different functions G , R , MRN.
4. The iterative part has five ROTR operations.

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 six functions. Five functions are used in compression function, one functions is used in message expansion part.

2.1.1 Function MRN(x1,……,x17)

Function MRN operates on seventeen words x1,…, x17, produces a word y as output. And funcion MRN as table 1:

Dynamic SHA-224/256	$\mathrm{t0} = (((((((\mathrm{x1} \oplus x2) + x3) \oplus x4) + x5) \oplus x6) + x7) \oplus x8) + x9$ $\mathrm{t0} = ((((((\mathrm{t0} \oplus x10) + x11) \oplus x12) + x13) \oplus x14) + x15) \oplus x16$ $\mathrm{t1} = (SHR^{17}(\mathrm{t0}) \oplus \mathrm{t0}) \wedge (2^{18} - 1)$ $\mathrm{t2} = (SHR^{10}(\mathrm{t1}) \oplus \mathrm{t1}) \wedge (2^{11} - 1)$ $\mathrm{t} = (SHR^{5}(\mathrm{t2}) \oplus \mathrm{t2}) \wedge 31$ $y = ROTR^{t}(x17)$
Dynamic SHA-384/512	$\mathrm{t0} = (((((((\mathrm{x1} \oplus x2) + x3) \oplus x4) + x5) \oplus x6) + x7) \oplus x8) + x9$ $\mathrm{t0} = ((((((\mathrm{t0} \oplus x10) + x11) \oplus x12) + x13) \oplus x14) + x15) \oplus x16$ $\mathrm{t1} = (SHR^{34}(\mathrm{t0}) \oplus \mathrm{t0}) \wedge (2^{35} - 1)$ $\mathrm{t2} = (SHR^{18}(\mathrm{t1}) \oplus \mathrm{t1}) \wedge (2^{18} - 1)$ $\mathrm{t3} = (SHR^{12}(\mathrm{t2}) \oplus \mathrm{t2}) \wedge (2^{12} - 1)$ $\mathrm{t} = (SHR^{6}(\mathrm{t3}) \oplus \mathrm{t3}) \wedge 63$ $y = ROTR^{t}(x17)$

Table 1. functions MRN for Dynamic SHA-2

2.1.2 Function GRT(x(1),……, x(16),t1)

Function GRT operates on sixteen w-bit words x1,…, x16 and an integer t1, produces an integer t2 as output. And funcion GRT as table 2:

$$\mathrm{t10} = t1 \wedge (w-1)$$

$$\mathrm{t11} = SHR^{\log_2^w}(t1)$$

$$t2 = \begin{cases} (SHR^{t10} X(t11)) \wedge (w-1) & w - t10 \geq \log_2^w \quad t1 < 966 \\ SHL^{\log_2^w - w + t10}(X(t11+1) \wedge (2^{w-t10} - 1)) + SHR^{t10} X(t11) & w - t10 < \log_2^w \quad t1 < 966 \\ SHR^{6}(X(16)) \wedge 15 & t1 = 966 \end{cases}$$

Table 2. functions GRT for Dynamic SHA-2

2.1.3 Function GGT(x1,t1)

Function GGT operates on one word x1 and an integer t1, produces an integer t2 as output. And funcion GGT as follow:

$$t2 = SHR^{t1}(x1) \wedge 3$$

2.1.4 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 3.1. functions GL 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 3.2. truth table for logical functions

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

Function ME1 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:

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

2.1.6 Function ME1(x1,x2, x3,x4)

Function ME1 operates on four words x1,x2, x3,x4. Produces one word as output. Function ME1 is same as SHA-2.

$$ME1(x1, x2, x3, x4) = ROTR^{1}(x1 \oplus x2 \oplus x3 \oplus x4)$$

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 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 $M_0^{(i)}, M_1^{(i)}, \ldots\ldots, M_{15}^{(i)}$.

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. 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-2 is the same as that of SHA-2 (given in Table 4).

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 4. The initial hash value, H^0 for Dynamic SHA-2

For i = 1 to N:
{
1.Message expansion part for obtaining additional thirty two 32-bit(resp.64-bit) words:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ME1(W_{t-3}, W_{t-8}, W_{t-14}, W_{t-16}) & 16 \le t \le 31 \end{cases}$$

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

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

3. For t=0 to 15(resp.27)
{

$$T = R(a,b,c,d,e,f,g,h,GRT(6\times t\times \log_2^w)) + W_{(16+t)\bmod 32}$$
$$h = ROTR^{GRT((1+6\times t)\times \log_2^W)}(g)$$
$$g = ROTR^{(GRT(2+6\times t)\times \log_2^W)}(f) + W_{31-t}$$
$$f = ROTR^{(GRT(3+6\times t)\times \log_2^W)}(e)$$
$$e = ROTR^{(GRT(4+6\times t)\times \log_2^W)}(d) + W_{(47-t)\bmod 32}$$
$$d = G(a,b,c,GGT(w-2-2\times t)) + W_t$$
$$c = ROTR^{(GRT(5+6\times t)\times \log_2^W)}(b)$$
$$b = MRN(W_0,...,W_{15},a)$$
$$a = T$$

}
4.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 5. Algorithmic description of Dynamic SHA-2 hash function.

2.6 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, $M^{(0)}, M^{(1)}, \ldots\ldots, M^{(N)}$, is processed in order, using the steps described algorithmically in Table 5. 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 $W_0, W_1, \ldots, W_{47}$. 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 $H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}$, 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.

3.1 Properties of message expansion part

In message expansion part $W_{16}, \ldots, W_{31}$ are produced by function ME1, It is easy to prove the following Theorem:

Theorem 1. *The message expansion of Dynamic SHA-224/256 is a bijectionξ:* $\{0,1\}^{512} \rightarrow \{0,1\}^{512}$ *and of Dynamic SHA-384/512 is a bijection ξ :* $\{0,1\}^{1024} \rightarrow \{0,1\}^{1024}$

Proof. It is enough to show that the message expansion part one is surjection, i.e. for every 16-tuple $W = (W_{16}, W_{17}, \ldots, W_{31})$ there exist a 16-tuple preimage $M = (M_0, M_1, \ldots, M_{15})$, such that ξ(M) = W.

By rearranging the recurrent equation that describes the message expansion part one for a given 4-tuple $W = (W_{17}, W_{23}, W_{28}, W_{31})$ we have the relation: $W_{31} = ROTR^{1}(W_{28} \oplus W_{23} \oplus W_{17} \oplus W_{15})$. From there it is straightforward to compute the unique value for W_{15}. Now, having the new 4-tuple $W = (W_{16}, W_{22}, W_{27}, W_{30})$, we can proceed further to compute the unique value for W_{14}, and so on until we compute the unique value for W_0. □

3.2 Properties of iterative part

In the iterative part, there are functions G, MRN, R and five ROTR operations, in one round, four message words will be mixed.

3.3 Design rationale

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

From the definition of function G, R and five ROTR operations, it is easy to know when the variable is different, the parameter of function G, R and five ROTR operations is different. So message value space is divided into $(4 \times 32^6)^{16} = 2^{512}$ (resp. 2^{1024}) parts. In different part the calculation is different.

When one bit different in message, the message will be in different part, the calculation will be different.

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.

Dynamic SHA-2 is even function, it means the number of collisions of every hash value is same. The workload for birthday attack is of O(2^{112}) (resp. O(2^{128}) O(2^{192}) O(2^{256})).

To analyse the relation between message value and hash value, it need know the unchangeable formulas that represent Dynamic SHA-2. Someone can use Algebraic Normal Form (ANF) to represent Dynamic SHA-2, but the ANFs that represent function R, MRN has up to 2^{256} , 2^{512} (resp. 2^{512} , 2^{1024}) 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 two 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. in Dynamic SHA-2. So attacker had to gusee the parameter that will be used in Dynamic SHA-2. All the 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, 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, MRN has up to 2^{256} , 2^{512} (resp. 2^{512} , 2^{1024}) 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, MRN has up to 2^{256}, 2^{512} (resp. 2^{512}, 2^{1024}) 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})).

4 Improvement

Although the ANFs for function MRN has up to 2^{512} (resp. 2^{1024}) monomials, attacker can use a series functions to replace it. for example $y_i = ROTR^{i}(x17)$ $0 \le i \le 31(resp.63)$. If use $b = MRN(W_t,...,W_{15+t},a)$ replace $b = MRN(W_0,...,W_{15},a)$ in table 5, attacker had to gusee 16(resp.27) times, there are $32^{16} = 2^{80}$ (resp. $64^{27} = 2^{162}$) combinations. And this will increase system calculation.

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, bits in message will affect different bits in 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 maybe different.* And I bring in data depend function R to realize the principle. And function R make attacker do not know what will happen, when message changed.

Function R 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^{256}, 2^{512} (resp. 2^{512}, 2^{1024}) 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 16 new variables, the iterative part has 16(resp 47) 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 $F:\{0,1\}^n \rightarrow \{0,1\}^r$ 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 $F^{(s)}:\{0,1\}^n \rightarrow \{0,1\}(s=1,2,...,r)$, 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 $0 \le t < w$. The integer t is constant.So we have $R:\{0,1\}^{8\times w} \rightarrow \{0,1\}^w$, 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 $2^{8\times w}$ monomials, where w is bit length of the word.

Function MRN

Function MRN operates on seventeen words $x1,...,x17$, produces a word as output. In Dynamic SHA-2 , the word x17 is produced with words $x1,...,x16$. So we have $R:\{0,1\}^{16\times w} \rightarrow \{0,1\}^w$, when one-bit different in $x1,...,x16$. different rotate right operation will be done on word x17, the bit in output maybe changed.

So the ANFs to represent function MRN has up $2^{16\times w}$ monomials, where w is bit length of the word.

Appendix 2: distribution of function.

Definition 1: *In function* $f = F(x)$*, to a value f, the number of variable value that has the output value f is s(F,f).*

Definition 2: *To a function* $F:\{0,1\}^{n1} \to \{0,1\}^{n2}$*, where n1, n2 is integer and n1,n2>0. f(i) is the i-th output value, where* $0 \le i \le 2^{n2} - 1$.

If n1≥n2, s(F,f(i))=2^{n1-n2} *, where* $0 \le i \le 2^{n2} - 1$.

If n1<n2, $s(F, f(i)) \in \{0,1\}$ *, where* $0 \le i \le 2^{n2} - 1$.

We call the function F is even function.

Theorem 2, *To a function* $F:\{0,1\}^{n1} \to \{0,1\}^{n2}$*, there is* $\sum_{i=0}^{2^{n2}-1} s(F, f(i)) = 2^{n1}$

Proof:

To a given variable value x(j), where $0 \le j \le 2^{n1} - 1$. there is a only output f(i)=F(x(j)). Where $0 \le i \le 2^{n2} - 1$. So we have $\sum_{i=0}^{2^{n2}-1} s(F, f(i)) \ge 2^{n1}$

To two different value f1, f2, there is not a variable value that has value f1, f2 at the same time. So we have $\sum_{i=0}^{2^{n2}-1} s(F, f(i)) \le 2^{n1}$

So *there is* $\sum_{i=0}^{2^{n2}-1} s(F, f(i)) = 2^{n1}$. □

Theorem 3, *function* $y = x1 + x2$ *and* $y = x1 \oplus x2$ *is even function. x1,x2 is w-bit word.*

Proof. To function $y = x1 + x2$, we have $Y:\{0,1\}^{2\times w} \to \{0,1\}^{w}$

There is the relation $x2 = x1 + y$. To a given y', there are 2^w 2-tuple (y',x1). To a given 2-tuple (y',x1'), it can compute the value for x2. So to the given y', there are 2^w 2-tuple (x1,x2) have the same value y'. So s(Y,y') =2^w=$2^{2\times w-w}$.

To every input value y, there is s(Y,y)=2^w=$2^{2\times w-w}$. S*o* the function $y = x1 + x2$ is even function.

Function $y = x1 \oplus x2$ is similarly. □

Theorem 4, *function* $y1 = x \wedge (2^n - 1)$ *,* $y2 = ROTR^n(x)$ *is even function. n is constant, x is nx-bit word, y1 is n-bit word, y2 is nx-bit word.*

Proof. To function $y1 = x \wedge (2^n - 1)$*, there is* $Y1:\{0,1\}^{nx} \to \{0,1\}^n$. $x = (x1, x2)$*, and x1 is the first nx-n bit of word x, x2 is the last n bit of word x.*

To a given value y1, there are 2^{nx-n} 2-tuple (x1,y1), there is the relation: x2=y1. it can compute the value x2. so to a given y1', there is 2^{nx-n} 2- tuple (x1,x2) have the value y1', So s(Y1,y1')= 2^{nx-n}.

To every input value y, there is s(Y1,y)=2^{nx-n}.So $y1 = x \wedge (2^n - 1)$ is even function.

To function $y = ROTR^n(x)$, there is the relation: $x = ROTL^n(y)$. To given y', it can compute the value x'. so s(ROTR,y)=1=2^{nx-nx}. $y = ROTR^n(x)$ is even function. □

Theorem 5, *we can not know function* $f = F(x1, x2) = F1(x1, x2) + x1$ *is even function or not,* $f1 = F1(x1, x2)$ *is even function.* $f1 = F1(x1, x2)$*, x1, f is n1-bit word, x2 is n2-bit word.*

Proof. To convenience, let $f2 = x2$.

Function z=x+y is even function, so to a given f', there is 2^{n1} 2- tuple (f1,x1) that make f=f+x1. To given f1', there are 2^{n2} 2- tuple (x1,x2) that make f1'=F1(x1,x2).

So to a given f', there are:

$$s(F,f')=\sum_{i=0}^{2^{n1}-1} s(F2,F2(x2_j)\,|\,(f1_i=F1(x1_i,x2_j)\,|\,f'=f1_i+x1_i))$$

If to every f', there is

$$s(F,f')=\sum_{i=0}^{2^{n1}-1} s(F2,F2(x2_j)\,|\,(f1_i=F1(x1_i,x2_j)\,|\,f'=f1_i+x1_i))=2^{n2}$$

Function F is even function.

If there are some value f' make

$$s(F,f')=\sum_{i=0}^{2^{n1}-1} s(F2,F2(x2_j)\,|\,(f1_i=F1(x1_i,x2_j)\,|\,f'=f1_i+x1_i))\neq 2^{n2}$$

Function F is not even function. Theorem 9 is example.

So we can not know function $f=F(x1,x2)=F1(x1,x2)+x1$ is even function or not. □

By theorem 5, it is know that when function F is constituted with some even functions, if there are relevant variables in these functions, function F maybe not even function. So mixing messages word many times maybe make the collisions of some hash values more than other hash values.

Theorem 6, *If function f=F(x1), g=G(x3,x2) are even function, x1 is n1-bit word , x2 is n2-bit word, x3 is n-bit word, f is n-bit word, g is ng-bit word. n1≥n and n+n2≥ng. Then funfction h=H(x1,x2)=G(F(x1),x2) is even function, h is ng-bit word .*

Proof:

We have $F:\{0,1\}^{n1}\rightarrow\{0,1\}^{n}$. function F is even function, to a given value f'=F(x1), there is $s(F,f')=2^{n1-n}$.

We have $G:\{0,1\}^{n+n2}\rightarrow\{0,1\}^{ng}$. function G is even function, to a given value g'=G(x3,x2), there is $s(G,g')=2^{n+n2-ng}$.

To a given 2-tuple (x3',x2'), there is g'=G(x3',x2'), and there is $s(F,x3')$ different x1 that has the value x3'=F(x1), so there is $s(F,x3')$ different 2-tuple (x1,x2') that mak g' =G(x3',x2')= G(F(x1),x2')=H(x1,x2').

There are $s(G,g')$ different 2-tuple (x3,x2) has the same value g'. So there are differen $s(F,x3')\times s(G,g')$ 2-tuple (x1,x2) has the given value g'=G(x3',x2')= G(F(x1),x2')= H(x1,x2).

So to every value g=G(f(x1),x2)=H(x1,x2), there are $s(F,x3')\times s(G,g')=2^{n1-n}\times 2^{n+n2-ng}=2^{n1+n2-ng}$ 2-tuple (x1,x2) that has the value g.

Function H has two input x1 and x2, so the bit-length of (x1,x2) is n1+n2. Because to every value g=G(f(x1),x2)=h=H(x1,x2), there are $2^{n1+n2-ng}$ 2-tuple (x1,x2) that has the value h. So $s(H,h)=2^{n1+n2-ng}$, So funfction H(x1,x2)=G(F(x1),x2) is even function. □

Theorem 7, *If function g=G(x3,x2) is even function, f=F(x1) is not even function, x1 is n1-bit word , x2 is n2-bit word, x3 is n-bit word, f is n-bit word, g is ng-bit word. n1≥n and n+n2≥ng.*
Then we can not know funfction H(x1,x2)=G(f(x1),x2) is even function or not.

Proof:

We have $F:\{0,1\}^{n1}\rightarrow\{0,1\}^{n}$. function F is *not* even function.

We have $G:\{0,1\}^{n+n2}\rightarrow\{0,1\}^{ng}$. function G is even function, to a given value g'=G(x3,x2), there is $s(G,g')=2^{n+n2-ng}$.

There are $s(G,g')$ different 2-tuple (x3,x2) has the make g'=*G(x3,x2)*.

To a given 2-tuple (x3',x2'), there is $s(F,x3')$ x1 that make F(x1)=x3', so there are $\sum_{i=0}^{s(G,g')} s(F,f_i'\,|\,G(f_i',x2_i)=g')$ 2- tuple (x1,x2) that make g' =G(x3',x2')= H(x1,x2').

If to every value h, there is $\sum_{i=0}^{s(G,h)} s(F, f_i \mid G(f_i, x2_i) = h) = 2^{n1+n2-ng}$, then function F is even function.

If there is a value h' that make $\sum_{i=0}^{s(G,h)} s(F, f_i \mid G(f_i, x2_i) = h') \neq 2^{n1+n2-ng}$, then function F is not even function. □

Theorem 8, *If function f=F(x1), g=G(x2) is even function, x1 is n1-bit word , x2 is n2-bit word, f is nf-bit word, g is ng-bit word. n1≥nf and n2≥ng.*
Then funfction H(x1,x2)=(F(x1),G(x2)) is even function, and $s(H, H(x1, x2)) = s(F, F(x1)) \times s(G, G(x2))$.

Proof. To given f',there are s(F,f') x1 that make f'=F(x1). To a given g',there are s(G,g') X2 that make g'=G(x2), So to given (f',g') there are $s(F, f') \times s(G, g') = 2^{n1-nf} \times 2^{n2-ng}$ $2^{n1+n2-nf-ng}$ 2-tuple (x1,x2) that make H(x1,x2)=(F(x1),G(x2)). So $s(H, H(x1, x2)) = 2^{n1+n2-nf-ng}$.
The bitlength of (x1,x2) is (n1+n2), the bitlength of (f(x1),g(x2)) is (nf+ng). And $s(H, H(x1, x2)) = 2^{n1+n2-nf-ng}$. So function H(x1,x2)=(F(x1),G(x2)) is even function. And $s(H, H(x1, x2)) = s(F, F(x1)) \times s(G, G(x2))$ □

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'. So s(G,y') $= 2^{2 \times w+2} = 2^{3 \times w+2-w}$. So funfction G(x1,x2,x3, t) is even function.

2, **Function R:**
Function y=R(x1,x2,x3,x4,x5) operates on five words x1,x2,x3,x4 and x5.
To a given value y'=R(x1,x2,x3,x4,x5), there is $2^{4 \times w}$ 4-tuple (y',x1',x2', x3',x4'). To a given 4-tuple (y',x1',x2',x3',x4'). there is the relation:

Dynamic SHA-224/256	$t0 = ((x1' \oplus x2') + x3') \oplus x4'$ $t1 = (SHR^{17}(t0) \oplus t0) \wedge (2^{18} - 1)$ $t2 = (SHR^{10}(t1) \oplus t1) \wedge (2^{11} - 1)$ $t = (SHR^{5}(t2) \oplus t2) \wedge 31$ $x5' = ROTR^{32-t}(y')$

Dynamic SHA-384/512	$t0 = ((x1' \oplus x2') + x3') \oplus x4'$ $t1 = (SHR^{34}(t0) \oplus t0) \wedge (2^{35} - 1)$ $t2 = (SHR^{18}(t1) \oplus t1) \wedge (2^{18} - 1)$ $t3 = (SHR^{12}(t2) \oplus t2) \wedge (2^{12} - 1)$ $t = (SHR^{6}(t3) \oplus t3) \wedge 63$ $x5' = ROTR^{64-t}(y')$

Table 8. functions R1 for Dynamic SHA-2

To given 5-tuple (y',x1',x2',x3',x4'), it can compute the value for x5, So there are $2^{4\times w}$ 5-tuple (y',x1',x2',x3',x4') have the same value y'. So s(R,y') $= 2^{4\times w} = 2^{5\times w - w}$. So funfction y=R(x1,x2,x3,x4,x5) is even function.

3, **There is only one message in a divided message value space part:**
From the definition of function GGT, GRT and iterative part, It is easy to know that the bits in message are used as parameter of function G, R and ROTR operation once. By theorem 8, it is easy know that function GGT, GRT divide the message space to $2^{16\times w}$ part, and in a part there is only one message value.