LAB Form for Iterated Hash Functions

Xigen Yao
Wuxi Hengqi Electromechanical Device Co.Ltd.China
email : dihuo377@163.com

Abstract.This paper propose a sufficient and efficient scheme for iterative hash functions to prevent and defeat those generic attacks , such as Multicollisions Attack,Second Preimage Attack and Herding Attack.The structure of this new framework is different from HAIFA or any other proposals,it contains a new method "Locking Abutting Blocks"(LAB)with checksum ,it makes a larger size of connotative chaining value without requirements of intricate computing and larger memory and it allows for an online computation in one pass with a fixed memory independently .It's also easy to avoid the generic attacks of Praveen Gauravaram and John Kelsey's.

.

keywords: hash function ,iterating ,abutting blocks,checksum

1 Introduction:

1.1

Hash functions play an increasingly important role in cryptography. The popular and dedicated hash functions stem from Rabin [6]. (Rabin used successive message blocks as keys to an iterated encryption operation using a block cipher.).The most hash functions used a"Merkle-Damgard" structure of iterating "compression function"[5],which transform a compression function $C : \{0,1\}^{m_c} \times \{0,1\}^n \to \{0,1\}^{m_c}$ into a hash function $h_C(\cdot)$.(where,for consistency with HAIFA,throughout this paper m_c denotes the size of the chaining value,and n denotes the size of the block inputted of in a iteration.)

The iterating hash functions try to maintain the following three properties:

* **pre-image resistance:**For any given code h,it is computationally infeasible to find x such that $h(x) = h$.

* **second pre-image resistance:**For any given block x,it is computationally infeasible to find $y \neq x$ with $h(y) = h(x)$.

* **collision resistance:**It is computationally infeasible to find any pair (x, y) such that $h(x) = h(y)$.

In fact,it is also required that the hash function can be computable in linear time (i.e. time $O(L)$) with limited storage,and furthermore generally can be computable in an on-line manner.

In the past few years many attacks have applied to the compression function of the hash function.

1. Generic attacks apply to the M-D construction.
2. Cryptanalytic attacks apply to the compression function of the hash function.

In this paper, we propose a new scheme for iterative hash functions to prevent those generic attacks.

In 1999,Dean [2] presented a second preimage attack on the M-D structure, he showed that fix-points of the compression function can be used for a second preimage attacks against long messages using $O(m2^{m/2})$ time and $O(m2^{m/2})$ memory (where m is the digest size). Later,the M-D structure was strengthened with addition of the length of the message,and in 2005 Ronald L. Rivest presented a way of "dithering"[7] the operation of an iterated hash function,to overcome some "message expansion" attacks,but the "dithering" was defeated by Kelsey and Schneier[8]. In 2004,Joux gave another attack , the multicollision attack [13] on cascade hash functions.In 2006, John Kelsey and Tadayoshi Kohno presented a generic preimage attack named Herding Attack[9].in 2008,Elena Andreeva , Charles Bouillaguet et al. developed a new generic long-message second preimage attack[14], based on combining the techniques in the second preimage attacks of Dean and Kelsey and Schneier with the herding attack of Kelsey and Kohno .

Recently,several hashing schemes were proposed ,such as the randomized hashing scheme proposed by Krawczyk and Halevi[10], the enveloped Damgaard-Merkle construction [11] proposed by Bellare and Ristenpart and HAIFA Scheme proposed by Eli Biham and Orr Dunkelman [12].

This paper presents a concise proposal for iterated hash function to prevent those generic attacks ,such as Multicollisions Attack,Second Preimage Attack and Herding Attack ,of course, we need a stronger compression function to avert the differential attacks on collision resistance .

The proposal used a new mode called "Locking Abutting Blocks" (LAB Mode),it means that for a r-rounds compression function C and the chaining values h_i:

$h_i = C(h_{i-1}, M_i)$,the block input M_i will be in reality replaced by the two abutting blocks M_{i-1} and M_i , such that:

$$h_i = C(h_{i-1}, M_{i-1}, M_i).$$

This method seems a bit similar from Concatenate-Permute-Truncate design strategy,which used in several hash functions e.g. Hash Function Hamsi (Hamsi is a Second Round Candidate of the SHA-3 competition),but they are different at all,further more, we improve the LAB Mode by replacing the abutting blocks with two checksums $\sum M_{i-1}$ and $\sum M_i$,e.g.,such that:

$h_i = C(h_{i-1}, \sum M_{i-1}, \sum M_i)$, and the related work contains some hash functions which using checksums. Synchronously,we'll avoid the generic attack of Praveen Gauravaram and John Kelsey's.

1.2 Related work

Hamsi

The Hash Function Hamsi is one of the candidates in second round SHA-3 competition.Hamsi is based on the Concatenate-Permute-Truncate design strategy used in several hash functions like Snefru[15] and Grindhal[16]. In addition to this approach,Hamsi uses a message expansion and a feedforward of the chaining value in each iteration. For each iteration of Hamsi-512 , the input includes a 512-bit chaining value and a 64-bit message, the concatenation of the 64-bit message is expanded to 512-bit , then connected with the 512-bit initial chaining value to be 1024 bit. Finally ,truncate the 1024-bit to produce 512-bit output value.

Hash Function with Linear-xor/additive Checksum

We quote *Cryptanalysis of a class of cryptographic hash functions* of Praveen Gauravaram and John Kelsey[17]: A family of variants of Damgaard-Merkle, in which a linear-XOR/additive checksum is computed over the message blocks, intermediate states of the hash function, or both. In a Damgaard-Merkle hash with a linear-XOR checksum, each bit of the checksum is a XOR function of the bits of the message, intermediate states or both; the checksum is processed as a final block after the padding and length encoding of the original message have been processed.E.g., the 3C construction[18] follow this pattern. F-Hash[19] uses a XOR checksum of the outputs of the compression function alongside a normal Damgaard-Merkle construction. Similarly, in a Damgaard-Merkle hash with linear-additive checksums, an additive checksum computed using message blocks, intermediate states or both is processed as a final block. The 256-bit GOST hash function specified in the Russian standard GOST [20] uses an additive checksum mod 2^{256} computed using 256-bit message blocks.

HAIFA

HAIFA is a framework proposed by Eli Biham and Orr Dunkelman, it can fix many flaws of Damgaard-Merkle construction .We quote Eli Biham and Orr Dunkelman below:

The main ideas behind HAIFA are the introduction of number of bits that were hashed so far and a salt value into the compression functions. Formally, a compression function $C : (0,1)^{m_c} \times (0,1)^n \times (0,1)^b \times (0,1)^s \rightarrow (0,1)^{m_c}$is proposed to substitute the form of M-D Structure $C_{MD} : (0,1)^{m_c} \times (0,1)^n \rightarrow (0,1)^{m_c}$, i.e., in HAIFA the chaining value h_i is computed as

$$hi = C(h_{i-1}, M_i, \#bits, salt),$$

where $\#bits$ is the number of bits hashed so far and *salt* is a salt value.

The following operations are performed:

1. Pad M according to the padding scheme .

2. Compute IV_m the initial value for a digest of size m using the prescribed way .

3. Iteratively digest the padded message using $C(\cdot)$, starting from the initial value IV_m and using the salt. In case an additional block is padded to the message, the compression function is called on this block with $\#bits = 0$.

4.Truncate the final chaining value if needed.

The rest of this paper is summarized as: In Section2 we explain the symbols.In Section3 we describe the Merkle- Damgaard construction simply.In Section4 We explain the new mode of LAB,and give a detailed analysis . In Section5 we summary this paper.

For brevity, we assume the size n of each block is 1024 bits in this paper,ie.,each the subblock of word is 64 bit.

The first is the explanation of symbols.

2 The Symbols And Setting

M denotes the message has been padded and Appended length as M-D structure,M is divided to be L blocks: $M_1M_2...M_{L-1}M_L$.

M_i denotes the i-th block,$1 \le i \le L$,and n denotes the size of each block M_i.

$+$ denotes addition modulo 2^{64} .

h_i denotes the i-th output of chaining value,m_c denotes the size of the chaining value.

$=$ denotes equal sign or assignment.

$\leftarrow$ denotes assignment or simultaneous assignment.

Each a block M_i contains 16 words: $m_{i,j}$, $0 \le j \le 15$ i.e:

$$m_{i,0}, m_{i,1}, ..., m_{i,15},$$

and similar, a block $\sum M_i$ contains 16 words:

$$\sum m_{i,0}, \sum m_{i,1}, ..., \sum m_{i,15} \ ,$$

and then,:

$$\sum M_i = \sum M_{i-1} + M_i \text{ ,i.e,}$$
$$\sum m_{i,j} = \sum m_{i-1,j} + m_{i,j} \ .$$

3 The Normal Mode of M-D Structure

The Merkle-Damgard construction is the most common way to transform a compression function $C : \{0,1\}^{m_c} \times \{0,1\}^n \rightarrow \{0,1\}^{m_c}$ into a hash function $h_C(.)$,the Message Digest is m_c-bit value.

C denotes the compression function. M denotes the padded and Appended message , it is formatted as $16L$ $n/16$-bit words :$m_0, m_1, ..., m_i, ..., m_{16L-1}$ ie.,the message is made up of L n-bit blocks and each the block contains 16 $n/16$-bit words,for h_C:

h_i = Chaining variable ,$h_0 = IV$(given Initial Value)

$M[i]$ = the i-th block

$h_i = C(h_{i-1}, M[i])$

$h_C(M) = h_L$

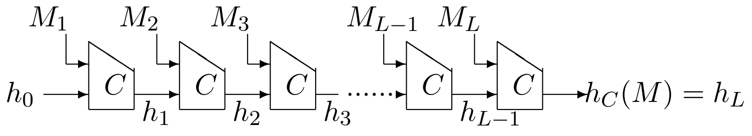


Fig.1. The Damgard-Merkle construction

4 New Structure

4.1 The New Mode of LAB

For message$M$:$M_1M_2...M_{L-1}M_L$, We give the compression function f :

$$h_i = f(h_{i-1}, M_{i-1}, M_i).$$

M_{i-1} and M_i are two successive blocks,we call them "LAB"Blocks,the i-th message block input M_i of M-D structure is replaced by M_{i-1} and M_i.

For a r-round compression function f,the i-th iteration is:

Copy the i-th block M_i of 16 64-bit words into Buffer:

$A[j] \leftarrow B[j]$,$B[j] \leftarrow m_{16i+j}$, $0 \leq j \leq 15$.

First, compute with the 16 words of $B[j]$ in the first 2 rounds, then compute with the 16 words of $A[j]$ in the rest $(r-2)$ rounds.(Of course, it's better that the 16 words of $B[j]$ are used in inconsecutive rounds. e.g.:Round 1 and Round 3 ;and the best is rearranging the sequences of the 32 words together from $A[j]$ and $B[j]$.)

We give the terse LAB mode below :

Define an additive block M_0.

For i from 1 to L,

$h_0 = IV$(given Initial Value)

$h_1 = f(h_0, M_0, M_1)$

$h_i = f(h_{i-1}, M_{i-1}, M_i)$

$h_L = f(h_{L-1}, M_{L-1}, M_L)$

$h_f(M) = h_L$

It is clear that the input of the i-th iteration increases a third part,the last block M_{i-1}.In reality this makes as M_{i-1} a connotative chaining variable for the i-th iteration (M_{i-1} can be also regarded as a concatenation as in Hamsi.),ie.,the (i-1)th iteration output is tantamount to a $(m_c + n)$-bit chaining variable,where n is the size of the block,in a general way,$n > m_c$,it means the size of the chaining variable has increased to more than $2m_c$,e.g., for a 512- hash code hash function,the size of the chaining variable in each iteration is tantamount to $(512 + 1024)$ bits,thus, LAB Mode is different from the others (such as the chaining variable output in Hamsi,$(h_i = T \cdot P \cdot C(E(M_i), h_i)) \bigoplus h_{i-1}$,the size of chaining variable was kept as n-bit, and the concatenation is expanded from 32-bit to 256-bit or 64-bit to 512-bit ,for Hamsi-512,the chaining variable outputed of each iteration is tantamount to $(512 + 64)$bits.).

In reality,the compression function f of LAB Mode increases a lager size of connotative chain variable without requirements of intricate compute and larger memory.It's suit to defeat Multicollisions Attack and Herding Attack since the increasing come from the last block itself.We will expound in next subsubsection.

4.1.1 Multicollisions Attack and Herding Attack

Multicollisions Attack

For Multicollisions[21],We quote *Multicollisions in Iterated Hash Functions Application to Cascaded Constructions* :In a normal iterating function with M-D construction ,recall that a collision is a pair of different messages M and M' such that $H(M) = H(M')$. Due to the birthday paradox, there is a generic attack that find collisions after about $2^{n/2}$ evaluations of the hash function, where n is the size in bits of the hash values.If there are t collisions of t pairs of different messages,then ,there can construct multicollisions of 2^t collisions.

First is how 4-collisions can be obtained,assume that two different blocks, A and A' that yield a collision, i.e. $f(h_0, A) = f(h_0, A')$. Let z denotes this common value and find two other blocks B and B' such that $f(z, B) = f(z, B')$. Put these two steps together to obtain the following 4-collision:
$f(f(h_0, A), B) = f(f(h_0, A), B') = f(f(h_0, A'), B) = f(f(h_0, A'), B')$

And 2^t-collision can obtain by analogy.

We assume f is an ideal compression function,in LAB Mode,

$$f : h_i = f(h_{i-1}, M_{i-1}, M_i)$$

Besides h_{i-1},each input contains the two abutting blocks of a iteration. Even the different blocks A and A' yield a collision,such that

$$f(h_0, M_0, A) = f(h_0, M_0, A') = z,$$

for the connected 2-block messages A-B and A'-B,we can't get a collisionin of the connected 2-block messages by substituting each other .

E.g., 2-block-messages A-B and A'-B

The first step of Block A or A' is:$f(h_0, M_0, A) = f(h_0, M_0, A') = z$

For the next iteration of Block B ,the input contains the last block,such that $f(z, A, B)$, $f(z, A^{'}, B)$,since the last blocks $A \neq A^{'}$, exactly the input of the second iteration $f(z, A, B)$ and $f(z, A^{'}, B)$ are different ,and obviously $f(z, A, B) \neq f(z, A^{'}, B)$. So, even A and $A^{'}$ yield a collision,we can't get a collision of a pair of connected message A-B and$A^{'}$-B.

It means if the corresponding collision blocks substitute each other in a connected message,the results are different.

Obviously,in LAB Mode,by analogy,the 2^t-collision ,i.e, Multicollisions Attack is not tenable.

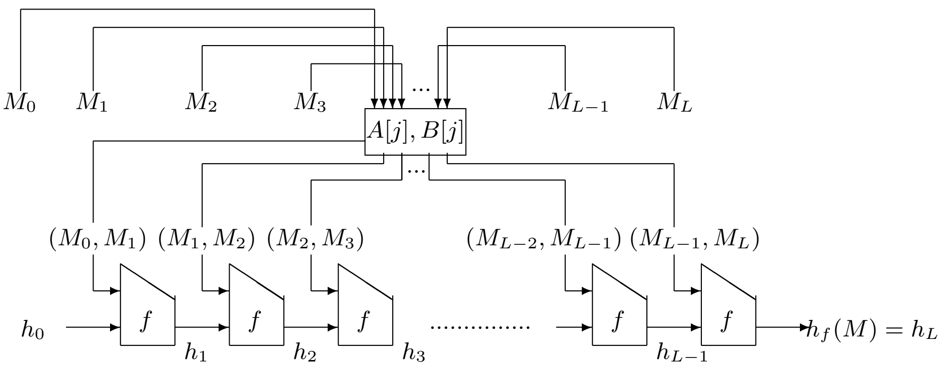
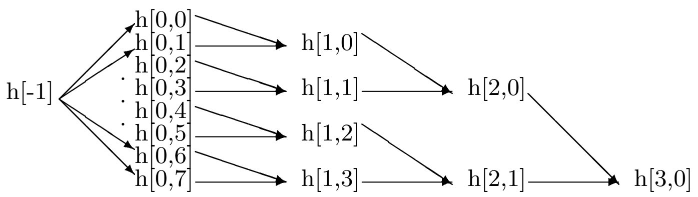


Fig.2. LAB construction

Herding Attack

Kelsey and Kohno give a diamond structure with width 2^k costs about $2^{k/2+n/2+3/2}$ work.

Diamond structure has total of $2^{k+1} - 2$ intermediate hash values.Usually use only 2^k in widest level.



An attacker who can find many collisions on the hash function by brute force can first provide the hash of a message. The attacker first does a large precomputation, and then commits to a hash value h. Later, upon being challenged with a prefix P, the attacker constructs a suffix S such that hash $(P \parallel S) = h$. Kelsey and Kohno ,Their paper introduced the "diamond structure"[5], which is reminiscent of a complete binary tree. It is a 2^l multi-collision in which each message in the multi-collision has a different initial chaining value, and which is constructed in the pre-computation step of the attack. The herding attack on an m_c-bit hash function requires approximately $2^{2m_c/3+1}$ work[4].where m_c is the size of the hash value.

We assume the attacker first does a large pre-computation and find many collisions on the m_c-bit hash function by brute force.By the previous analysis ,in LAB Mode,we can defeat this type of generic attack .

The first reason is :We can't get a collision of the connected message by substituting the corresponding collision blocks each other,the multicollisions can't be used in building a diamond tree.

The second reason is :Any a chaining value output of a 2-block connected message doesn't only depend on chaining value input, it is also strictly related to the last block itself, this has broken the connected paths of the diamond tree.

The third reason is :Since each block outputs a large connotative chaining variable(the connotative size of n is much greater than the size m_c of CV.)and the effectual size of CV outputted is tantamount to $(m_c + n)$-bit, it make the size of chaining variable so large that the connection can't depend on the m_c-bit chaining variable .

Obviously,in LAB Mode,Herding Attack is not tenable.

4.2 LAB's Family

By using checksum $\sum M_i$,we can build the other forms of LAB:

a $h_i = f(h_{i-1}, M_{i-1}, \sum M_i)$;

b $h_i = f(h_{i-1}, \sum M_{i-1}, \sum M_i)$;

c $h_i = f(h_{i-1}, \sum M_{i-1}, M_i)$;

where $\sum M_i$ is the sum (modulo addition) of the i blocks ,it can also be the xor operation of the i blocks.

For each unit $m_{i,j}$ of $M_i(m_{i,0}, m_{i,1}, ..., m_{i,15})$ and each unit $\sum m_{i,j}$ of $\sum M_i(\sum m_{i,o}, \sum m_{i,1}, ..., \sum m_{i,15})$ $(1 \leq i \leq L$,$0 \leq j \leq 15)$,

Define:

$\sum M_i = \sum M_{i-1} + M_i$ i.e.:

$\sum m_{i,j} = \sum m_{i-1,j} + m_{i,j}$

We provide a detailed form Type C below(for 64-bit words and 512-bit hash value).

1.Append padding bits and append length just as M-D Structure:

The message is padded with single 1-bit followed by the necessary number of 0-bits,so that its length l congruent to 896 modulo1024 $[l \equiv 896(mod 1024)]$,append a block of 128 bits as an unsigned 128-bit integer(most significant byte first)and contains the length of the original message.M denotes the message after padding bits and appending length. message. M is split to be L blocks:$M_1M_2...M_L$

2.Difine a additive block M_0,encode the size of hash value m_c into M_0,just like

HAIFA.

3.Define an initial value IV,Set buffer $A[j]$ and $B[j]$,for a r-round compression function f ,for i from 1 to L,do the following operations of each iteration,and get the hash value $h_f(M)$:

$h_0 = IV, \sum M_0 = M_0$

$h_i = f(h_{i-1}, \sum M_{i-1}, M_i)$

$h_L = f(h_{L-1}, \sum M_{L-1}, M_i)$

$h_f(M) = h_L$

4.Truncate the final chaining value if needed.

We provide details :

♢ The first iteration $(i = 1)$:

$\sum M_0 = M_0$; $A[j] \leftarrow \sum M_0$; $B[j] \leftarrow M_1$;

For the first 2 rounds,compute with the input of $A[j]$,and the rest $(r-2)$ rounds, compute with the input of $B[j]$.At the end of the first iteration,operate:

$\sum M_1 = \sum M_0 + M_1$; $A[j] \leftarrow \sum M_1$.

♢ For i ($2 \leq i \leq L$) , $B[j] \leftarrow M_i$,

For the first 2 rounds,compute with the input of $A[j]$,and the rest $(r-2)$ rounds, compute with the input of $B[j]$,at the end of the iteration,operate:

$\sum M_i = \sum M_{i-1} + M_i$; $A[j] \leftarrow \sum M_i$

In reality,for a r-round compression function,the input of $A[j]$ can be computed in other round freely. it's better that the 16 words of $B[j]$ are used in inconsecutive rounds. e.g.:Round 1 and Round 3 ;and the best is rearranging the sequences of the 32 words from $A[j]$ and $B[j]$.

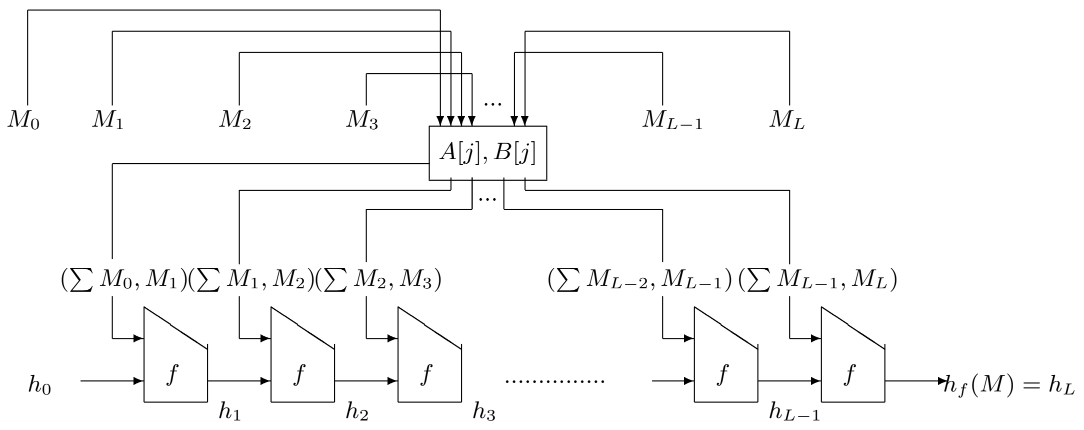


Fig.3. LAB Form C

4.2.1 Second Preimage Attacks And The Generic Attacks of Praveen Gauravaram and John Kelsey's

The second preimage attack of Dean means[7]is ,to insert a block (or blocks)at so called a fixed point i-th block,then make a preassigned output CV_i equal to the input CV_i,thus,the attacker can get a new message with a same hash code of the primary message.

Although M-D Structure strengthened by encoding the length into the last block to prevent the second preimage attack of Dean,the "message expansion" attack (proposed by Kelsey and Schneier) avoids the strengthening by repeating the inserted block to satisfy the length. And a way of "dithering" proposed by Ronald L. Rivest can't block the second preimage attacks proposed by Andreeva et al.

The hashes with linear-xor/additive checksum can easily defeat the second preimage attacks,for the length of message is restricted to checksum .However,the hashes with linear-xor/additive checksum are defeated by Praveen Gauravaram and John Kelsey [1],who use new techniques of Extending Joux 1-block multicollision attack and Checksum Control Sequences.

Checksum Control Sequences(CCS)is as a chunk of data which lets an attacker to control the checksum value in the hash functions with linear checksums. The attacker constructs the CCS by building a Joux multicollision of the correct size using a random choice of message blocks. He then uses the CCS to actually control the checksum using a checksum control algorithm without changing any intermediate hash value on the iterative chain.

LAB Mode with checksum e.g.LAB Form C can easy overcome this attack.

1.Joux's multicollision attack is not tenable in LAB Mode no matter the multicollisions are" 1-block" or "multi-block" ones .

2.If we change any a block of a message in LAB Mode, exactly the input of the next iteration has been changed, the checksum will change and the output of chaining value, ie.,the intermediate hash value will change,and the other blocks will change so on,...,The attacker can't control the intermediate hash value and checksum at all.

So,a random choice of message blocks can't achieve.

By putting the checksum and block together ,we make the the checksum and block as input of each iteration,if the intermediate checksum has any change,the intermediate chaining value output will change immediatly . So,LAB Forms with checksum are different from those of other hash functions with linear-xor/additive checksums.

5 Summary

In LAB form ,the abutting 2 blocks are inputted together in a iteration, we can regard the last block as the connotative chaining variable in each iteration (and we can also regard the last block as the feedforward of the chaining value in each iteration). Since the connotative chaining variable is the last block itself,it means the connotative chaining variable will different if any different block has substituted .Thus,the multicollisions attack is struck at the foundation,and the herding attack loses it's foundation also.

The LAB form with checksum is easy to defeat the 2nd Preimage attack.

We can compare LAB structure with HAIFA structure .

In HAIFA structure ,each iteration inputs the effective size of chaining variable is tantamount to $(n + \#bits, salt)$ bits,maybe this size is not the most sufficient.

In LAB structure,since the size of the connotative chaining variable is so larger,it means that each iteration inputs a size of $(m_c + n)$-bit chaining variable!

The form of the table below is from HAIFA .

Type of Attack	Idel Hash Function $=$	MD $\geq$	HAIFA fixed salt $\geq$	HAIFA with(distinct)salts $\geq$	LAB (Form C) $\geq$
Preimage	2^{m_c}	2^{m_c}	2^{m_c}	2^{m_c}	2^{m_c}
one-of-many pre-image ($k^{'} < 2^s$targets)	$2^{m_c}/k^{'}$ ($k^{'} < 2^s$targets)	$2^{m_c}/k^{'}$	$2^{m_c}/k^{'}$	2^{m_c}	2^{m_c}
Second-pre-image (k blocks)	2^{m_c}	$2^{m_c}/k$	2^{m_c}	2^{m_c}	2^{m_c}
one -of- many second pre-image(k blocks in total,$k^{'} < 2^s$messages	$2^{m_c}/k^{'}$	$2^{m_c}/k$	$2^{m_c}/k^{'}$	2^{m_c}	2^{m_c}
Collision	$2^{m_c/2}$	$2^{m_c/2}$	$2^{m_c/2}$	$2^{m_c/2}$	$2^{m_c/2}$
Multi-collision (k-collision)	$2^{m_c(k-1)/k}$	$\lceil \log 2^K \rceil 2^{m_c/2}$	$\lceil \log 2^K \rceil 2^{m_c/2}$	$\lceil \log 2^K \rceil 2^{m_c/2}$	2^{m_c}
Herding Offline:	–	$2^{m_c/2+t/2}$	$2^{m_c/2+t/2}$	$2^{m_c/2+t/2+s}$	–
Herding online:		2^{m_c-t}	2^{m_c-t}	2^{m_c-t}	–

Complexities of Attacks on Merkle-Damgaard and HAIFA Hash Functions with Comparison for an Ideal Hash Function and LAB Form C

Since LAB structure increases the size of the connotative chaining variable by the last block itself,this doesn't need requirements of intricate computing and larger memory ,we can develop it a multi-block LAB structure.E.g.,a 3-block LAB structure:

$$h_i = f(h_{i-1}, M_{i-2}, M_{i-1}, \sum M_i)$$

References

[1] Praveen Gauravaram.John Kelsey.*Cryptanalysis of a class of cryptographic hash functions.*Addison–wesley Co.,Inc.,reading,2009

[2] Richard D. Dean.*Formal Aspects of Mobile Code Security.*, Ph.D. dissertation, Princeton University, 1999.

[3] Ivan Damgaard, *A Design Principle for Hash Functions*, Advances in Cryptology, proceedings of CRYPTO 1989, Lecture Notes in Computer Science 435, pp. 416C 427, Springer-Verlag, 1990.

[4] Ralph C. Merkle, *Secrecy, Authentication, and Public Key Systems*, UMI Research press, 1982.

[5] Ralph C. Merkle, *One Way Hash Functions and DES*, Advances in Cryptology, proceedings of CRYPTO 1989, Lecture Notes in Computer Science 435, pp. 428C446, Springer-Verlag, 1990.

[6] M. O. Rabin. Digitalized signatures. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 155C168. Academic Press, 1978.

[7] Rivest, R.L. *Abelian Square-Free Dithering for Iterated Hash Functions.* Pre- sented at ECrypt Hash Function Workshop, June 21, 2005, Cracow, and at the Cryptographic Hash workshop, November 1, 2005, Gaithersburg, Maryland (Au- gust 2005)

[8] John Kelsey, Bruce Schneier,*Second Preimages on n-Bit Hash Functions for Much Less than* 2^n, Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 474C490, Springer-Verlag, 2005.

[9] John Kelsey, Tadayoshi Kohno, *Herding Hash Functions and the Nostradamus At- tack*, preproceedings of Cryptographic Hash Workshop, held in NIST, Gaithers- burg, Maryland, 2005.

[10] Shai Halevei, Hugo Krawczyk, *Strengthening Digital Signatures via Randomized Hashing*, Advances in Cryptology, proceedings of CRYPTO 2006, Lecture Notes in Computer Science 4117, pp. 41C59, Springer-Verlag, 2006.

[11] Mihir Bellare, Thomas Ristenpart, *Multi-Property-Preserving Hash Domain Ex- tension: The EMD Transform*, NIST 2nd hash function workshop, Santa Barbara, August 2006.

[12] Eli Biham, Orr Dunkelman, *A Framework for Iterative Hash Functions HAIFA*, NIST 2nd hash function workshop, Santa Barbara, August 2006.

[13] Antoine Joux, *Multicollisions in Iterated Hash Functions*, Advances in Cryptology, proceedings of CRYPTO 2004, Lecture Notes in Computer Science 3152, pp. 306C 316, Springer-Verlag, 2004.

[14] Elena Andreeva and Charles Bouillaguet and Pierre-Alain Fouque and Jonathan J. Hoch and John Kelsey and Adi Shamir and Sebastien Zimmer *Second Preimage Attacks on Dithered Hash Functions*at Eurocrypt 2008.

[15] Merkle R.C. *A Fast Software One-Way Hash Function.* Journal of Cryp- tology, 3(1):43-58, 1990.

[16] R. Knudsen, L., Rechberger, C., S. Thomsen.*Grindahl- a family of hash functions*. In Biryukov, A, ed.: Fast Software Encryption, FSE 2007. Vol- ume 4593 of Lecture Notes in Computer Science, Springer-Verlag (2007) 39-57

[17] Praveen Gauravaram ,John Kelsey*Cryptanalysis of a class of cryptographic hash functions*, CT-RSA 2008

[18] Praveen Gauravaram.*Cryptographic Hash Functions: Cryptanalysis, Design and Applications*. PhD thesis, Information Security Institute, Queensland University of Technogy, June 2007.

[19] Duo Lei. *New Integrated proof Method on Iterated Hash Structure and New Structures*. Cryptology ePrint Archive, Report 2006/147, 2006. The paper is available at http://eprint.iacr.org/2006/147.pdf. Last access date: 5th of November 2006.

[20] Government Committee of the Russia for Standards. GOST R 34.11-94, Gosudarstvennyi Standard of Russian Federation, Information Technology, Cryptographic Data Security, Hashing function, 1994.

[21] Antoine Joux *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions*,In Proceedings of CRYPTO, LNCS 3152, pp. 306-316, Springer, 2004