A New Hash Construction and a Specific Instance Dolly

Duo Lei[1], Da Lin[2], and Li Chao[1]

[1] Department of Science, National University of Defense Technology, Changsha, China
Duoduolei@163.com
[2] College of Mechanical and Electronic Control Engineering, Beijing Jiaotong University, BeiJing, China

Abstract. Most of block cipher based iterated hash functions are based on block ciphers with different input, output and key modes. We put forward a new type of iterated hash function, whose compression function is build from modified Feistel Structure and block cipher's round function, we call the compression function "FL-Cipher" and call the iterated hash function "FL-Construction hash function". The FL-Construction and FL-Cipher are designed to be secure and its security relies much on the security of known block cipher algorithm and Feistel structure. We also prove that FL-Cipher is a good function for build compression function and FL-Construction is a OWHF and CRHF, in black box model. We also describe an specific instance of FL-Construction named Dolly, which has same round function and key schedule algorithm as Rijndael and its security relies much on the security of Rijndael and the security of Feistel Structure.

1 Introduction

A hash function is a function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ for a fixed positive integer n and with the property that $H(x)$ is easy to compute for all $x \in \{0,1\}^*$ for any person. A cryptographic hash function is a hash function with collision resistant. A hash function uses a secret parameter (the key) then called a Message Authentication Code or MAC.

Almost all known hash functions are based on a compression function with fixed size input and called an "iterated" hash function. The iterated hash functions have been divided into four classes[2]: hash function based on a block cipher, hash functions based on modular arithmetic, hash functions based on a knapsack and dedicated hash functions.

The main idea of hash function based on block cipher construction is limited in changing the input, output and key modes of block cipher to build a hash function, it rarely considers the block cipher algorithm used in dedicated hash function construction.

In this paper, We present a new type of iterated hash function, whose compression function has modified Feistel Structure and block cipher's round function, we call the compression function "FL-Cipher" and call the iterated hash

function "FL-Construction hash function". We prove that FL-Cipher is a good function for build compression function and FL-Construction is a OWHF and CRHF, in black box model. We also describe an specific instance of FL-Construction named Dolly, which has same round function and key schedule algorithm as Rijndael.

The paper is organized as follows. The mathematical preliminaries and notation employed are described in section2. Specification of FL-Cipher and FL-Construction are given in section3. The hash function Dolly is presented in section4 and section5 is our conclusion.

2 Definition

2.1 The Feistel Like Structure

A Feistel structure is a general way of constructing block ciphers from simple functions. The original idea was used in the block cipher, invented by Horst Feistel. Let Feistel structure be adopted in a block cipher with round function f. Let $x^L_{(r)}, x^R_{(r)}$ be the left and the right halves of the r round inputs, The Feistel structure of block cipher is written as:

$$x^L_{(r+1)} = x^L_{(r)} \oplus f(x^R_{(r)}, k_{(r)}) \tag{1}$$

$$x^R_{(r+1)} = x^L_{(r)} \tag{2}$$

The security of the Feistel structure is not obvious, but analysis of DES[3] has shown that it is a good way to construct ciphers. And some new ciphers based on Feistel structure of SPN function have been discussed recently and no weakness is found in Feistel structure itself. In this section we give a modified structure of Feistel named Feistel like structure and call FL-structure.

Definition 1. *Let f be round function, $x_{(r)}$ be the rth round inputs, $x_{(1)}$ be the input sequence, then the FL-Structure is defined as Eq.(3), Eq.(4).*

$$x_{(2)} = f(x_{(1)}, k_{(1)}) \tag{3}$$

$$x_{(r+1)} = x_{(r-1)} \oplus f(x_{(r)}, k_{(r)}) \tag{4}$$

Put simply, the standard Feistel network takes a function from n bits to n bits and produces an invertible function from $2n$ bits to $2n$ bits. FL-Structure takes a function from n bites to n bites and produces a one-way function from n bits to n bites. Figure illustration is given in Fig.1.

2.2 The FL-Construction

A block cipher is a map $E : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$ for each $k \in \{0,1\}^\kappa$, where the round function of E is a map of $f : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, the functions $E_k(\cdot) = E(\cdot, k)$ is transformation on $\{0,1\}^n$. If E is a permutation then E^{-1} is its inverse, where $E^{-1}_k(y)$ is the string x such that $E_k(x) = y$.

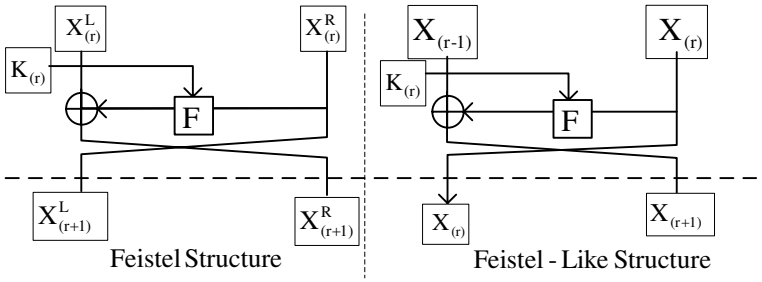

Fig. 1. Construct Between Feistel Structure and FL-Sreucture

Definition 2 (*Block*(n, κ))**.** *Let E be block cipher $E : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, f be round function, Block(n, κ) is the set of all block cipher with form of E, where E has form of Eq(5) and its round function f is a permutation.*

$$x_{(r+1)} = f(x_{(r)}, k_{(r)}),\ r = 1, ..., R \tag{5}$$

Definition 3 (FL-Cipher). *Let $E \in Block(n, \kappa)$ be a A block cipher, let f be round function of E and let R be rounds of E. Then we called F is FL-Cipher based on E, if $y = F(x, k)$ has the form of that:*

$$x_{(2)} = f(x_{(1)}, k_{(1)}) \tag{6}$$

$$x_{(r+1)} = x_{(r-1)} \oplus f(x_{(r)}, k_{(r)}), r = 1, ..., R' \tag{7}$$

Definition 4 (Feistel Cipher and FL-Cipher). *Let F be FL-Cipher hash function with round function f and rounds R, $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, $\widetilde{E} : \{0,1\}^{2n} \times \{0,1\}^\kappa \to \{0,1\}^{2n}$ be Feistel block cipher with round function f and rounds R, then we call that F is instance of $\widetilde{E}$.*

Definition 5 (No Weak Hash Key). *Let F be FL-Cipher, $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, and F is instance of $\widetilde{E} : \widetilde{E}_k(x\|x') = (y\|y')$, where $x, x', y, y' \in \{0,1\}^n$, if there are no weakness in round function f, key schedule ,Feistel structure and the key schedule satisfy*

$$Pr[x, x', y, y' \xleftarrow{\$} \{0,1\}^n; k \leftarrow \{0,1\}^n : \widetilde{E}_k(x\|x') = y\|y'] \approx 1/2^n \tag{8}$$

we call the FL-Cipher F has no weak hash key.

In this paper all discussions about FL-Cipher are on condition that the FL-Cipher has no weak hash key.

Definition 6 (FL-Construction). *Let $Feist(n, \kappa)$ be the set of all FL-Cipher $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$ with no weak hash key. We call the iterated hash function is a FL-Construction hash function, if the iterated hash function H's compression function is a FL-Cipher and denoted H_F.*

We write $x \xleftarrow{\$} S$ for the experiment of choosing a random element from the finite set S and calling it x. An adversary is an algorithm with access to one or more oracles.

2.3 Definition of Collision Resistant

There are many kinds of descriptions and notations about hash function, we use the descriptions given by John Black[7].

To quantify the collision resistance of a hash function H_F, we instantiate the compression function by a randomly chosen $F \in Feist(n, \kappa)$ with round function f. An adversary A is given oracles for $F(\cdot, \cdot)$ and wants to find a collision for H_F that is, M, M' where $M \neq M'$ but $H_F(M) = H_F(M')$. We look at the number of queries that the adversary makes and compare this with the probability of finding a collision.

Definition 7 (Collision resistance[7]). *Let H_F be a FL-Construct hash function, $H_F : Feist(n, \kappa) \times \mathcal{D} \to \mathcal{R}$, and let A be an adversary. Then the advantage of A in finding collisions in H_F is the real number*

$$Adv^{coll}_{H_F}(A) = Pr[F \xleftarrow{\$} Feist(n, \kappa); (M, M') \leftarrow A^F :$$

$$M \neq M' \wedge H_F(M) = H_F(M')].$$

For $q \geq 1$ we write $Adv^{coll}_{H_F}(q) = max_A\{Adv^{coll}_{H_F}(A)\}$ where the maximum is taken over all adversaries that ask at most q oracle queries. Other advantage functions are silently extended in the same way.

We use the following measure for the difficulty of inverting a hash function at a random point.

Definition 8 (Inverting random points[7]). *Let H_F be FL-Construct hash function, $H_F : Feist(n, \kappa) \times \mathcal{D} \to \mathcal{R}$, and let A be an adversary. Then the advantage of A in inverting H_F is the real number.*

$$Adv^{inv}_{H_F}(A) = Pr[F \xleftarrow{\$} Feist(n, \kappa); \sigma \xleftarrow{\$} \mathcal{R};$$

$$M \leftarrow A^F(\sigma) : H_F(M) = \sigma].$$

Definition 9 (Conventional definition of a OWF[7]). *Let H_F be hash function, $H_F : Feist(n, \kappa) \times \mathcal{D} \to \mathcal{R}$, and let l be a number such that $\{0, 1\}^l \subseteq \mathcal{D}$. Let A be an adversary. Then the advantage of A in inverting H_F on the distribution induced by applying H_F to a random l-bit string is the real number.*

$$Adv^{owf}_{H_F}(A) = Pr[F \xleftarrow{\$} Feist(n, \kappa); M \xleftarrow{\$} (\{0, 1\}^n)^l; \sigma \xleftarrow{\$} H_F(M);$$

$$M' \leftarrow A^F(\sigma) : H_F(M') = \sigma].$$

We also define the advantage of an adversary in finding collisions in a FL-Cipher. Naturally (k, m) and (k', m') collide under F if they are distinct and $F(k, m) = F(k', m')$.

Definition 10 (Collision resistance of FL-Cipher). *Let F be a FL-Cipher $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, and let A be an adversary. Then the advantage of A in finding collisions in F is the real number.*

$$Adv_F^{coll}(A) = Pr[F \xleftarrow{\$} Feist(n,\kappa); ((m,k),(m',k')) \leftarrow A^F :$$

$$(m \neq m' \vee k \neq k') \wedge F_k(m) = F_{k'}(m')].$$

Definition 11 (One Way of FL-Cipher). *Let F be FL-Cipher hash function, $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, and let A be an adversary. Then the advantage of A in inverting F on the distribution induced by applying F is the real number.*

$$Adv_F^{owf}(A) = Pr[F \xleftarrow{\$} Feist(\kappa,n); k \xleftarrow{\$} \{0,1\}^\kappa; m \xleftarrow{\$} \{0,1\}^n; \sigma \xleftarrow{\$} F_k(m);$$

$$m' \leftarrow A^F(\sigma) \vee k' \leftarrow A^F(\sigma) : F_k(m) = F_k(m') \vee F_k(m) = F_{k'}(m)].$$

Definition 12 (Inverting random points of FL-Cipher). *Let F be FL-Cipher hash function, $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, and let A be an adversary. Then the advantage of A in inverting F is the real number.*

$$Adv_F^{inv}(A) = Pr[F \xleftarrow{\$} Feist(\kappa,n); \sigma \xleftarrow{\$} \{0,1\}^n;$$

$$(m,k) \leftarrow A^F(\sigma) : F_k(m) = \sigma].$$

2.4 The Specification of Rijndael

Rijndael is a substitution-linear transformation network with $10 \sim 14$ rounds, depending on the plaintext and key size. A data block to be processed using Rijndael is partitioned into an array of bytes, and each of the cipher operations is byte-oriented. For Rijndael, the block length and the key length can be independently specified to any multiple of 32 bits, with a minimum of 128 bits, and a maximum of 256 bits.

Rijndael's round function consists of four layers. In the first layer an 8×8 S-box is applied to each byte called 'ByteSub'. The second and third layers are linear mixing layers , in which the rows of the array are shifted, and the columns are mixed, called 'ShiftRow' and 'MixColumn' respectively. In the fourth layer, subkey bytes are XORed into each byte of the array called 'AddRoundKey' by EXOR of Sub-key k_i where $i = 1, ..., N_r$. In the last round, the column mixing is omitted, where Nr are $10 \sim 14$ when key size κ are $(N_k = 4 \sim 8) \times 32$ bits and block size n are $(N_b = 4 \sim 8) \times 32$ bits.

The ByteSub Transformation is a non-linear byte substitution, operating on each of the State bytes independently. The substitution table (or S-box) is invertible and is constructed by the composition of two transformations: First, taking the multiplicative inverse in $GF(2^8)$, then, applying an affine (over $GF(2)$)

transformation.

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{9}$$

In ShiftRow, the rows of the State are cyclically shifted over different offsets. Row 0 is not shifted, Row 1 is shifted over C_1 bytes, row 2 over C_2 bytes and row 3 over C_3 bytes. The $C_1 = 1$,$C_2 = 2$,$C_3 = 3$, when the block length $N_b = 4 \sim 6$, the $C_1 = 1$,$C_2 = 2$,$C_3 = 4$, when the block length $N_b = 7$, the $C_1 = 1$,$C_2 = 3$,$C_3 = 4$, when the block length $N_b = 8$.

In MixColumn, the columns of the State are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x)$, given by $c(x) =' 03'x^3 +' 01'x^2 +' 01'x +' 02'$. This polynomial is coprime to $x^4 + 1$ and therefore invertible. This can be written as a matrix multiplication. Let $b(x) = c(x) \otimes a(x)$.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{10}$$

In AddRoundKey, a Round Key is applied to the State by a simple bitwise EXOR. The Round Key is derived from the Cipher Key by means of the key schedule. The Round Key length is equal to the block length N_b.

3 Collision Resistance of FL-Construction

3.1 Collision Resistance of FL-Cipher

In this section we discuss the hash properties of FL-Cipher with no weak hash key, there are some notations which will be used in following descriptions. Let $Pr(x|f(x) = y)$ be the probability of that a randomly selected x satisfies the equation of $f(x) = y$ for any given y, $(m_1 \| m_2)$ be concatenate of m_1 and m_2, $Ri(m, n)$ is the right n bits of sequence m and Δ, Δ' be $\Delta = \widetilde{E}_k(m \| f_k(m)), \Delta' = \widetilde{E}_k(m' \| f_k(m'))$.

Theorem 1 (Inverting random points of FL-Cipher). *Let F be FL-Cipher hash function, $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, and let A be an adversary. Then the advantage of A in inverting F is $Adv_F^{inv}(q) = q/2^{n-1}$.*

Proof. Let an adversary A for F, adversary A takes oracle F and input σ and, when successful, it outputs k, m and have

$$F(m, k) = \sigma \Leftrightarrow Ri(\widetilde{E}_k(m \| f_k(m)), n) = \sigma$$

Since we have

$$Pr[\sigma \xleftarrow{\$} \{0,1\}^n; m,k \leftarrow \{0,1\}^n : Ri(\Delta, n) = \sigma] \approx 1/2^n \tag{11}$$

$$Pr[\sigma \xleftarrow{\$} \{0,1\}^n; m,k,y \leftarrow \{0,1\}^n : \widetilde{E}_k^{-1}(y\|\sigma) = (m|f_k(m)] \approx 1/2^n \tag{12}$$

So we have $Adv^{owf}_{F(q)} \leq q/2^{n-2}$. □

Theorem 2 (One Way Property of FL-Cipher). *Let F be FL-Cipher with round function f, $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, then $Adv^{owf}_F(q) \leq q/2^{n-1}$.*

Proof. Let an adversary A for F: adversary A takes oracle F and input m, k, σ and, when successful, it outputs m' or k' such that $F(m,k) = F(m',k)$ or $F(m,k) = F(m,k')$. If adversary A find m' such that $F(m,k) = F(m',k)$ then

$$\begin{aligned} & F(m,k) = F(m',k) \\ \Leftrightarrow\ & Ri(\widetilde{E}_k(m\|f_k(m)), n) = Ri(\widetilde{E}_k(m'\|f_k(m')), n) \end{aligned}$$

For block cipher $\widetilde{E}$,

$$Pr[m,k \xleftarrow{\$} \{0,1\}^n; m' \leftarrow \{0,1\}^n : Ri(\Delta, n) = Ri(\Delta', n)] = 1/2^n \tag{13}$$

$$Pr[m,k \xleftarrow{\$} \{0,1\}^n; y \leftarrow \{0,1\}^n : \widetilde{E}_k^{-1}(y\|Ri(\Delta, n)) = m'\|f_k(m')] = 1/2^n \tag{14}$$

For any $i \in [1..q]$, let C_i be the event that the randomly selected m_i from $\{0,1\}^n$, where $m_i \neq m_j$ such that $F(m,k) = F(m_i,k)$. Since $Pr(C_i) \leq 1/(2^n - i)$. we thus have $Pr(c_1 \vee ... \vee c_q) \leq q/2^{(n-1)}$.

If adversary A find k' such that $F(m,k') = F(m,k)$ then

$$\begin{aligned} & F(m,k) = F(m,k') \\ \Leftrightarrow\ & Ri(\Delta, n) = Ri(\widetilde{E}_{k'}(m\|f_{k'}(m)), n) \end{aligned}$$

For block cipher $\widetilde{E}$, since there is no weak key, then

$$Pr[m,k \xleftarrow{\$} \{0,1\}^n; k' \leftarrow \{0,1\}^n : Ri(\Delta, n) = Ri(\widetilde{E}_{k'}(m\|f_{k'}(m)), n)] \approx 1/2^n \tag{15}$$

$$Pr[m,k \xleftarrow{\$} \{0,1\}^n; k', y \leftarrow \{0,1\}^n : \widetilde{E}_{k'}^{-1}(y\|Ri(\Delta, n)) = m\|f_{k'}(m)] \approx 1/2^n \tag{16}$$

Similar as description of $F(m,k) = F(m',k)$, we get $Adv^{owf}_F(q) \leq q/2^{n-1}$. □

Theorem 3 (Collision resistance of FL-Cipher). *Let F be a FL-Cipher $F : \{0,1\}^n \times \{0,1\}^\kappa \to \{0,1\}^n$, then $Adv^{coll}_F(q) \leq q/2^{n-2}$*

Proof. Let an adversary A for F, adversary A takes oracle F and , when successful, it outputs k, m and k', m' such that $F(m,k) = F(m',k')$

$$\begin{aligned} & F(m,k) = F(m',k') \\ \Leftrightarrow\ & Ri(\Delta, n) = Ri(\widetilde{E}_{k'}(m'\|f_{k'}(m')), n) \end{aligned}$$

For block cipher $\widetilde{E}$, since there is no week key, we have

$$Pr[m,k,m',k' \leftarrow \{0,1\}^n : Ri(\Delta,n) = Ri(\widetilde{E}_{k'}(m' \| f_{k'}(m')),n)] \approx 1/2^n \quad (17)$$

$$Pr[m,k,m',k',y \leftarrow \{0,1\}^n : E_{k'}^{-1}(y \| Ri(\Delta,n)) = m' \| f_{k'}(m')] \approx 1/2^n \quad (18)$$

So we have $Adv_{F(q)}^{coll} \leq q/2^{n-1}$ □

3.2 Collision Resistance of FL-Construction

In this section we discuss the security of FL-Construction.

Theorem 4. *If $F : Feist(n,n) \times (\{0,1\}^n \times \{0,1\}^n) \rightarrow \{0,1\}^n$ is a FL-Cipher, if F is one way and collision resistant function, then the FL-Construction hash function H_F is OWHF and CRHF.*

Lemma 1 (Damgård-Merkle[8]). *Let F be a compression function of FL-Construction hash function H_F, $F : Feist(n,n) \times (\{0,1\}^n \times \{0,1\}^n) \rightarrow \{0,1\}^n$. Then $Adv_{H_F}^{coll}(q) \leq Adv_F^{coll}(q)$ for all q.*

Proof. Let A be a collision-finding adversary for H_F that take oracles F. We construct from A a collision-finding adversary B for F. Adversary B also takes oracle F. Let B run A. When A makes a F query, B forwards it to F and returns to A the result. For $i \in [1..q]$, we say that the *ith* triple is (x_i, k_i, y_i) if A's *ith* oracle query was an F-query of (k_i, x_i) and this returned y_i. Algorithm B records the list of triples. Eventually A halts with an output $(M, M') = (m_1 \cdots m_a, m'_1 \cdots m'_b)$.

Have B compute $H_F(M)$ and $H_F(M')$. According to our conventions, all of the necessary queries for B to use in this computation are already recorded in B's list of triples, so no new oracle calls are needed to compute $H(M)$ and $H(M')$.

Adversary B inspects its list of triples to see if there exists distinct (x,k,y) and (x',k',y'). If so, B outputs this pair of points. Otherwise, B inspects its list of triples to see if there exists a triple (x,k,h_0). If so, B outputs (k,x), (k,x).

We claim that B succeeds whenever A succeeds. By symmetry, we can assume without loss of generality that $a \leq b$. If $H_F(M) = H_F(M')$ then $h_a = F(h_{a-1}, m_a) = F(h'_{b-1}, m'_b) = h'_b$. If $h_{a-1} \neq h'_{b-1}$ or $m_a \neq m'_b$ then we are done, otherwise check $h_{a-1} = F(h_{a-2}, m_{a-1}) = F(h'_{b-2}, m'_{b-1}) = h'_{b-1}$. Again, if $h_{a-2} \neq h'_{b-2}$ or $m_{a-1} \neq m'_{b-1}$, then we are done. proceeding in this way, we must find some values $\alpha \in [1,a]$ and $\beta \in [1,b]$ such that either $h_\alpha = F(h_{\alpha-1}, m_\alpha) = F(h'_{\beta-1}, m'_\beta) = h'_\beta$, where $h_{\alpha-1} \neq h'_{\beta-1}$ or $m_\alpha \neq m'_\beta$ or $h_\alpha = h_0$ □

Lemma 2 (Lai-Massey[13]). *Let FL-Cipher F be a compression function of FL-Construction hash function H_F, $F : Feist(n,n) \times (\{0,1\}^n \times \{0,1\}^n) \rightarrow \{0,1\}^n$. Then $Adv_{H_F}^{OWF}(q) \leq Adv_F^{OWF}(q)$ for all q.*

Proof. Let A be an adversary for H_F: adversary A takes oracles F and an input σ and, when successful, it outputs M such that $H_F(M) = \sigma$. We construct an adversary B for F: adversary B takes oracles F and an input σ and, when successful, it outputs (h, m) such that $F(h, m) = \sigma$. Adversary B works as follows. It runs A on σ. When A makes an F query, adversary B forwards the query to its F oracle and returns to A the result. During this process, for each $i \in [1..q]$, we say that the ith triple is (x_i, k_i, y_i) if A's ith oracle query was an F-query of (k_i, x_i) and this returned y_i. Adversary B records the list of triples. Eventually A halts with an output $M = m_1 \cdots m_a$. At that point we have B compute $H_F(M)$: for $i \leftarrow 1$ to a set $h_i \leftarrow F(h_{i-1}, m_i)$. According to our conventions, all of the necessary $F(k, x)$ values that B needs will already be in Bs list of triples no new oracle calls are needed to compute $H_F(M)$. Now if $H_a = F(h_(a-1), m_a) = \sigma$ then B outputs (h_{a-1}, m_a) and wins its experiment; otherwise it outputs (h_0, m_1) and does not win. Clearly B succeeds whenever A succeeds.

Let adversary A takes oracles F and an input M, when successful, it outputs M' such that $H_F(M) = H_F(M')$. We construct an adversary B for F: adversary B takes oracles F and an input M and, when successful, it outputs (h, m) and (h', m') such that $F(h, m) = F(h', m')$. Let B run A. When A makes a F query, B forwards it to F and returns to A the result. For $i \in [1..q]$, we say that the ith triple is (x_i, k_i, y_i) if A's ith oracle query was an F-query of (k_i, x_i) and this returned y_i. Algorithm B records the list of triples. Eventually A halts with an output $(M, M') = (m_1 \cdots m_a, m'_1 \cdots m'_b)$.

Have B compute $H_F(M)$ and $H_F(M')$. According to our conventions, all of the necessary queries for B to use in this computation are already recorded in B's list of triples, so no new oracle calls are needed to compute $H(M)$ and $H(M')$.

Adversary B inspects its list of triples to see if there exists distinct (x, k, y) and (x', k', y'). If so, B outputs this pair of points. Otherwise, B inspects its list of triples to see if there exists a triple (x, k, h_0). If so, B outputs (k, x), (k, x).

We claim that B succeeds whenever A succeeds. By symmetry, we can assume without loss of generality that $a \leq b$. If $H_F(M) = H_F(M')$ then $h_a = F(h_{a-1}, m_a) = F(h'_{b-1}, m'_b) = h'_b$. If $h_{a-1} \neq h'_{b-1}$ or $m_a \neq m'_b$ then we are done, otherwise check $h_{a-1} = F(h_{a-2}, m_{a-1}) = F(h'_{b-2}, m'_{b-1}) = h'_{b-1}$. Again, if $h_{a-2} \neq h'_{b-2}$ or $m_{a-1} \neq m'_{b-1}$, then we are done. proceeding in this way, we must find some values $\alpha \in [1, a]$ and $\beta \in [1, b]$ such that either $h_\alpha = F(h_{\alpha-1}, m_\alpha) = F(h'_{\beta-1}, m'_\beta) = h'_\beta$, where $h_{\alpha-1} \neq h'_{\beta-1}$ or $m_\alpha \neq m'_\beta$, or $h_\alpha = h_0$ and $h_\beta = h_0$ □

4 The Hash Function Dolly

4.1 The Specification of Dolly

Dolly is an FL-Construction with Rijndael as underlying block cipher. As Rijndael supports $(4 \sim 8) \times 32$ bits of outputs and key bytes, Dolly is a hash function of $(4 \sim 8) \times 32$ bits outputs and with the message blocks of same length as output blocks. Let n be the length of output bits of Dolly, and N_b be $n/32$.

Let $F : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be the compression function of Dolly, and f be round function of F, we have:

$$f(x,k) = MixColumn(ShiftRow(SubByte(x))) \oplus k \tag{19}$$

$$\stackrel{\triangle}{=} M_C(S_R(S_B(x))) \oplus k. \tag{20}$$

Let x be the input block, y be the output block, k be the key, n be block length, R be the round number and $k_1, k_2, ..., k_R$ be round key from k using Rijndael key schedule.Then compression function $y = F(x,k)$ be as follow and illustrated in Fig2.

$$x_1 = x, x_2 = f(x_1, k_1) \tag{21}$$

$$x_r = x_{r-1} \oplus f(x_r, k_r), r = 2, ..., R \tag{22}$$

$$y = x_R, N_b = n/32, R = N_b + 7 \tag{23}$$

Let H_F be the Hash function Dolly, let $h_i(m_i, h_{i-1})$ be round function of H_F, let M be the message to hash and assume that it includes padding bits and the message length. Let $M = m_1, m_2, ..., m_q$, where each m_i is n bits. Let $IV =' 123456789ABCDEF123456789ABCDEF'$ be initial value, then the hash function Dolly is given as, Dolly is illustrated in Fig3:

$$h_0 = left(IV, N_b \times 32), N_b = n/32 \tag{24}$$

$$h_i(m_i, h_{i-1}) = F(h_{i-1}, m_i) \ for \ i = 1, ..., q. \tag{25}$$

$$H_F(h_0, M) = h_q \tag{26}$$

In the algorithm of Dolly we can distinguish a number of steps is similar to Pelican[11]-an improved design of ALPHA-MAC[6].

Message Padding: pad the message by appending a single 1 bit followed by the minimum number of 0 bits so that the resulting length is a multiple of $n = N_b \times 32$ bits. Split the message M in n-bit message words $m_1, m_2, ..., m_q$.

Chaining: get the fist message word as state h_0, applying the compression function F to the state h_i and using the m_i as key.

Finalization: the last round output of compression function are the tag.

4.2 Security Claims

The security of Dolly depend on the security of FL-Construction, FL-Cipher and Rijndael. Since the fist two factors have been discussed in previous section, the inner security of Dolly's compression function F is discussed in this section.

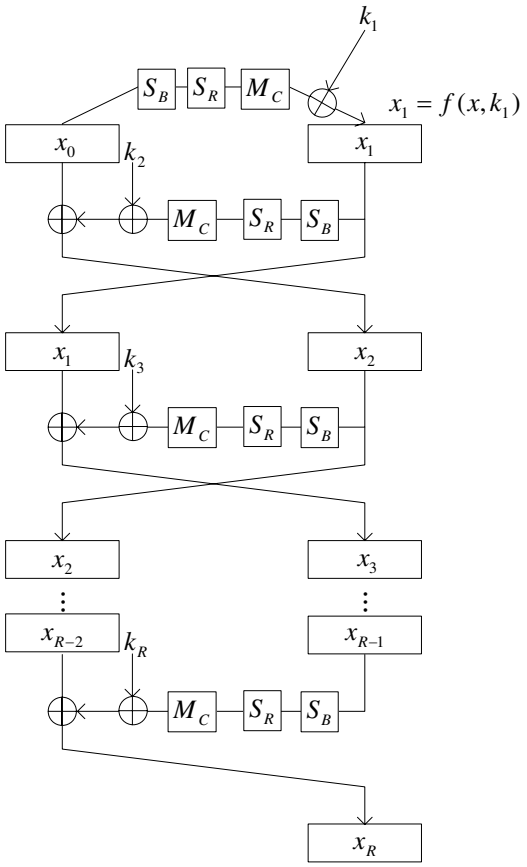

Fig. 2. Encryption procedure of Dolly's Compression function F

Design Motivation The hash function is far more sensitive to defects on designing than that of block cipher, for finding a weak key does not results in failure of that block cipher, but finding a collision in hash function always means the failure of hash function. Rijndael is expected, for all key and block lengths defined, to behave as good as can be expected from a block cipher with the given block and key lengths[10]. The most efficient key-recovery attack for Rijndael is exhaustive key search. Obtaining information from given plaintext-ciphertext pairs about other plaintext-ciphertext pairs cannot be done more efficiently than by determining the key by exhaustive key search. The rationale for this is that a considerable safety margin is taken with respect to all known attacks. All those design principles are needed for compression function of FL-Construction.

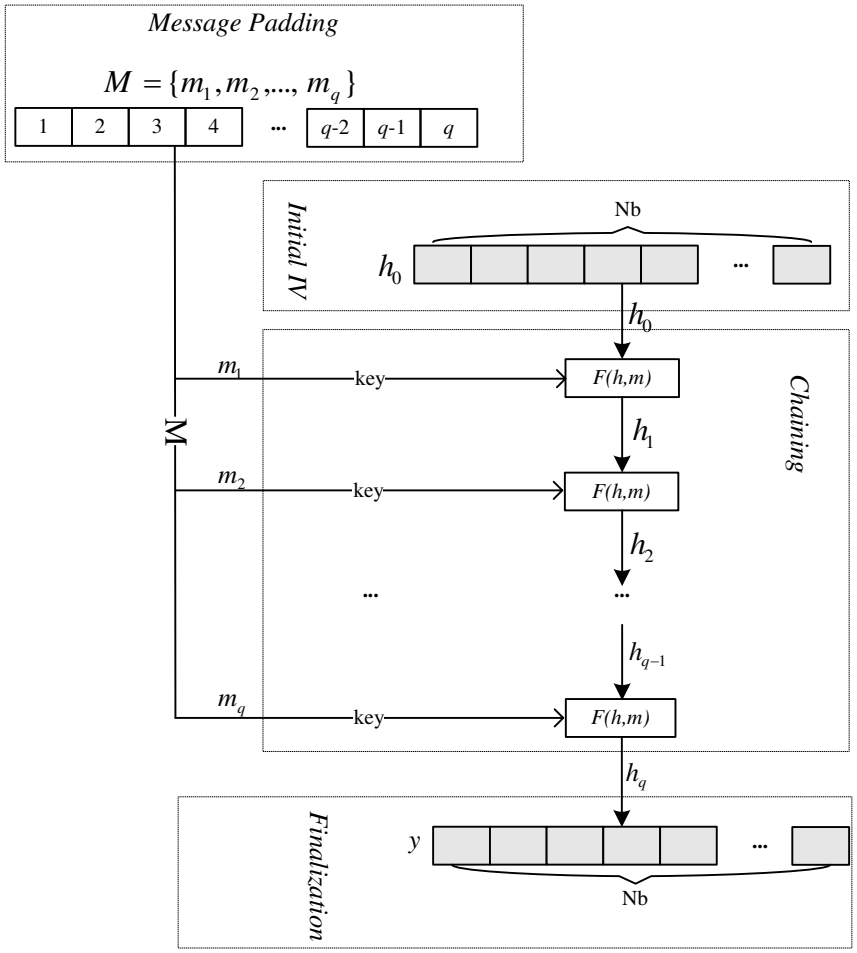

Fig. 3. Hashing Procedure of Dolly

Weak keys The key schedule of Compression function $F(m, k)$ uses that of Rijndael, for which the non-linearity of the key expansion practically eliminates the possibility of equivalent key, and there are no week keys find in rijndael.

Differential and linear cryptanalysis The Rijndael's minimum number of active S-boxes in any 4-round differential or linear trail is 25. The minimum number of active S-boxes in $F(m, k)$ is larger that that of Rijndael, when the round number is more that 4 round. This gives a maximum prop ratio of 2^{-150} for any 4-round differential trail and a maximum of 2^{-75} for the correlation for any 4-round linear trail for $F(m, k)$.

Differential and linear cryptanalysis The Rijndael's minimum number of active S-boxes in any 4-round differential or linear trail is 25. The minimum number of active S-boxes in $F(m, k)$ is larger that that of Rijndael, when the round number is more that 4 round. This gives a maximum prop ratio of 2^{-150} for any 4-round differential trail and a maximum of 2^{-75} for the correlation for any 4-round linear trail for $F(m, k)$.

The Square attack The "Square" attack is a dedicated attak on Square[14] that exploits the byte-oriented structure of Square cipher and was published in the paper presenting the Square cipher itself. This attack is valid for Rijndael. Since the active byte diffusion speed of compression function $F(m, k)$ is faster that Rijndael, we conclude it is immune against square attack.

4.3 Performance

The performance of Dolly is similar to that of Rijndael, can express the performance in terms of Rijndael operations, more particularly, the Rijndael key schedule and the Rijndael encryption operation. This allows to use Rijndael benchmarks for software implementations on any platform or even hardware implementations to get a pretty good idea on the performance of Dolly.

One iteration of Dolly corresponds roughly to one more round encryption than that of Rijndael encryption, hence roughly $1 + 1/R$ of an Rijndael encryption, where R is the round number of Rijndael. Using this rough approximation, we can state that hashing a message $M = m_1, ..., m_q$ requires: $q(1 + 1/R)$ times Rijndael key schedule and $q(1+1/R)$ times Rijndael encryption. Hence, the performance of the Dolly can be estimated at $q(1 + 1/R)$ times the performance of Rijndael encryption.

5 Conclusion

In this paper we present a new way to construct hash function. Security of FL-Construction relies on the security of FL-Cipher, the security of FL-Cipher relies on the security of the block cipher which is based and the security of Feistel structure. Security requirement of FL-Cipher is more stronger than that of based block cipher. Therefore if we construct a good FL-Cipher, that means we get a good block cipher, but a good block cipher does not always mean a good hash function. The efficiency of Dolly relies on that of Rijndael, since Rijndael is fast in 8-bit and 32-bit platform, the performance of Dolly is also fast in 8-bit and 32-bit platform, that is also the principle of Message expansion algorithm. Further analysis of the security of this primitive is underway.

References

1. B.Preneel, V. Rijmen, A.Bosselaers: Recent Developments in the Design of Conventional Cryptographic Algorithms. In State of the Art and Evolution of Computer

Security and Industrial Cryptography. Lecture Notes in Computer Science, Vol 1528. Springer-Verlag, Berlin Heidelberg New York(1998) 106-131.
2. B.Preneel: The State of Cryptographic Hash Functions. In Lectures on Data Security, Lecture Notes in Computer Science, Vol. 1561. Springer-Verlag, Berlin Heidelberg New York (1999) 158-182.
3. FIPS 46-3: Data Encryption Standard. In National Institute of Standards and Technology, Oct. 1999.
4. B. Van Rompay,Analysis and design of cryptographic hash functions, MAC algorithms and block cipher, K. U. Leuven, Juni 2004
5. J. Daemen and V. Rijmen: The Design of Rijndael: AES The Advanced Encryption Standard. Springer, 2002.
6. Joan Daemen and Vincent Rijmen, "A new MAC Construction Alred and a Specific Instance Alpha-MAC,", Fast Software Encryption 2005, LNCS H. Gilbert, H. Handschuh, Eds., Springer-Verlag, to appear.
7. J.Black, P.Rogaway, and T.Shrimpton, "Black-box analysis of the block-cipher-based hashfunction constructions from PGV". In Advances in Cryptology - CRYPTO'02, volume 2442 of Lecture Notes in Computer Science. Springer-Verlag, 2002.pp.320-335.
8. I.Damgård. A design principle for hash functions. In G. Brassard, editor, Advances in Cryptology-CRYPTO' 89, volume 435 of Lecture Notes in Computer Science. Springer-Verlag, 1990.
9. B. Preneel, R. Govaerts, and J. Vandewalle, " Hash functions based on block ciphers,", In Advances in Cryptology -CRYPTO'93, Lecture Notes in Computer Science,pages 368-378. Springer-Verlag, 1994.
10. J. Daemen and V. Rijmen. The Design of Rijndael: AES-The Advanced Encryption Standard, Springer-Verlag, 2002.
11. J.Daemen and V.Rijmen, "The Pelican MAC Function,", http://eprint.iacr.org/088.pdf.
12. C.E. Shannon. "Communication theory of secrecy systems,", Bell System Technical Journal, 28:656 C 715, 1949.
13. X. Lai and J. L. Massey: Hash functions based on block ciphers. In Advances in Cryptology Eurocrypt'92, Lecture Notes in Computer Science, Vol. 658. Springer-Verlag, Berlin Hei-delberg New York (1993) 55-70.
14. J. Daemen, L. R. Knudsen and V. Rijmen.: The Block Cipher SQUARE. In Fast Software En-cryption, Lecture Notes in Computer Science, Vol. 1267. Springer-Verlag, Berlin Heidelberg New York (1997) 149-165.