A NEW MAC: LAMA

Li An-Ping

Beijing 100085, P.R.China
apli0001@sina.com

Abstract

In this paper, we will propose a MAC with two versions, which is called LAMA. The proposal MAC has the input size of 128 bits and 256 bits in the versions 1, 2 respectively, and output size of 128 bits. There have not been found a attack better than the exhaustive search attack for the MAC, and it has a fast implementations in about $5\,cycles/byte$ in the both versions.

Keywords: MAC, hash function, pre-image resistant, 2^{nd} pre-image resistant, collision, MAC forgery.

1. Introduction

Hash functions play a very important role in data integrity, message authentication and digital signature. MAC (message authentication codes) is one of keyed hash functions with the specific purpose of message authentication. There are two kinds of general methods to construct a MAC, one is that applying a block cipher as a compression function, and the other is that remoulding a ready unkeyed hash function. For the detail materials about these aspects may refer to see [2]. In this paper, we will propose a MAC with two versions, which is called LAMA, that means the main tools applied are key-defined transformations in linear algebra. In LAMA, the sizes of input are 128 bits and 256 bits in the versions 1, 2 respectively, and the sizes of output are 128 bits. LAMA has the security strength of 128 bits and a fast implementations in about $5\ cycles/byte$ in the both versions.

2. Construction

Denoted by $\mathbb{F}$ the finite field consisted of the two elements $\{0,1\}$, that is, the finite field $GF(2)$, and $\mathbb{F}[x]$ is the polynomial ring of unknown x over the field $\mathbb{F}$, the symbol $\oplus$ represents the addition in the field $\mathbb{F}$.

Let $p(x) \in \mathbb{F}[x],\ p(x) = x^8 \oplus x^6 \oplus x^5 \oplus x \oplus 1$, which is a primitive polynomial of degree eight, and $\mathbb{K} = \mathbb{F}[x]/\ p(x)$ is a finite field $GF(2^8)$. In LAMA will apply a S-box defined as following

$$S_0(x) = 5 \cdot (x \oplus 3)^{127}, \quad x \in \mathbb{K}. \tag{2.1}$$

We also adopt the representation $S_0(\zeta)$ for a bytes string ζ to represent that S-box S_0 substitute each byte of the string ζ.

Now we show a result related to the construction later. Denoted by Ω_u and Ω_l the sets of the non-singular upper-triangular $n \times n$ matrices and the lower-triangular $n \times n$ matrices on the finite field $\mathbb{F}$ respectively. Suppose that P is a permutation matrix of order n, denoted by $\mathcal{M}_P = \{A \cdot P \cdot B \mid A \in \Omega_l, B \in \Omega_u\}$. We will also use the same symbol P to represent the corresponding permutation on the indices (of rows or columns).

Proposition 1 Suppose that P is a permutation matrix, let $\tau(P)$ be the number of the pairs $(i, j), 0 \le i < j < n,$ with that $P(i) > P(j)$, then

$$\left|\mathcal{M}_P\right| = 2^{n^2-n-\tau(P)}. \tag{2.2}$$

Moreover, if P and R are two distinct permutation matrices, then

$$\mathcal{M}_P \cap \mathcal{M}_R = \varnothing \ . \tag{2.3}$$

Proof. Suppose that there are $A, X \in \Omega_l$ and $B, Y \in \Omega_u$ such that $APB = XPY$, then $P^{-1}(X^{-1}A)P = YB^{-1}$. Let $\Delta = (P^T \cdot \Omega_l \cdot P) \cap \Omega_u$, then we have

$$\left|\mathcal{M}_P\right| = \left|\Omega_u\right| \cdot \left|\Omega_l\right| / \left|\Delta\right| = 2^{n^2-n} / \left|\Delta\right|.$$

Hence, to prove formula (2.2) is suffice to prove the equation

$$\left|\Delta\right| = 2^{\tau(P)}. \tag{2.4}$$

Suppose that $A \in \Omega_l$, $(P^T AP) = B \in \Delta$, $A = (a_{i,j})$, $B = (b_{i,j})$, it has

$$b_{P(i),P(j)} = a_{i,j}, \quad 0 \le i, j < n \ .$$

Thus, it must be that

$$P(i) > P(j), \ i < j, \quad \text{if} \ \ a_{i,j} = 1.$$

In other words, an entry $a_{i,j}, 0 \le i < j,$ may take the values 1, or 0, iff $P(i) > P(j)$, otherwise $a_{i,j} = 0$. This has proven the equation (2.4) and (2.2).

Now we come to prove (2.3). On the contrary, if (2.3) is not true, from the shown above, then there are $A \in \Omega_l$ and $B \in \Omega_u$ such that

$$P^T AR = B \ .$$

Suppose that $A = (a_{i,j})$ and $B = (b_{i,j})$, then it has that

$$b_{P(i),R(j)} = a_{i,j}, \quad 0 \le i, j < n \ .$$

Especially,

$$b_{P(i),R(i)} = a_{i,i} = 1, \quad 0 \le i < n.$$

As the assumption $B \in \Omega_u$, we deduce that

$$P(i) \le R(i), \quad \forall \ \ 0 \le i < n.$$

But

$$\{P(i)\}_0^{n-1} = \{i\}_0^{n-1} = \{R(i)\}_0^{n-1}.$$

Thus we conclude that

$$P(i) = R(i), \quad 0 \le i < n.$$

That is, $P = R$. □

In the next is the description of the detail construction. Firstly, we will introduce two key-defined transformations $F(\zeta)$ and $G(\zeta)$, which are the main cryptographic transformations applied in present algorithm.

The expression $(x_1\ x_2 \cdots x_r)$ as usual stands for the circular permutation. Suppose that a and b are two non-negative integers, denoted by

$$(a \cdots b) = \begin{cases} (a\,(a+1) \cdots b) & if\ a \le b, \\ (a\,(a-1) \cdots b) & if\ a > b. \end{cases}$$

For an 8-bits vector v with weight s, denoted by $I_v = \{\, i \mid v[i] = 1, 0 \le i < 8 \}$, $i_1 < i_2 < \cdots < i_s \in I_v$. From the vector v, we make a permutation P_v as following

$$P_v = \begin{cases} (i_1 \cdots i_2)(i_3 \cdots i_4) \cdots (i_{s-1} \cdots i_s) & if\ s\ even, \\ (i_1 \cdots i_2) \cdots (i_{s-2} \cdots i_{s-1})(7 \cdots i_s) & if\ s\ odd. \end{cases} \tag{2.5}$$

The symbol P_v will be also used to represent the corresponding permutation matrix.

For a string ρ of 8 bytes, we define an 8-bits vector v_ρ and a non-singular 8×8 matrix M_ρ:

$$v_\rho[i] = \rho[8i + i]_{bit}, 0 \le i < 8, \qquad M_\rho = T_u \cdot P_{v_\rho} \cdot T_l\,. \tag{2.6}$$

where $T_u = (a_{i,j})_{8\times 8}$ and $T_l = (b_{i,j})_{8\times 8}$ are the upper-triangular matrix and the lower-triangular matrix respectively,

$$a_{i,j} = \begin{cases} \rho[8i + j]_{bit} & if\ i < j, \\ 1 & if\ i = j, \\ 0 & if\ i > j, \end{cases} \qquad b_{i,j} = \begin{cases} \rho[8i + j]_{bit} & if\ i > j, \\ 1 & if\ i = j, \\ 0 & if\ i < j, \end{cases} \tag{2.7}$$

Suppose that K is the secret key, let $\lambda = K[0,15]_{byte} \oplus K[16,31]_{byte}$, if $|K| = 256$, else $\lambda = K[0,15]_{byte}$, and $\lambda' = \lambda[0,7]_{byte}$, $\lambda'' = \lambda[8,15]_{byte}$, define two affine transformations on $\mathbb{K}$

$$A(x) = M_{\lambda'}(x), \qquad B(x) = M_{\lambda''}(x), \qquad x \in \mathbb{K}\,. \tag{2.8}$$

Denoted by $V_1 = V_{\lambda'} \oplus V_{\lambda''}$, and $V_2 = V_{\lambda'} \oplus ROTL8(V_{\lambda''}, 1)$, and then define a new S-box $S(x)$ and a transformation L on $\mathbb{K}^4$,

$$S(x)=S_0(x\oplus V_2)\oplus V_1,\qquad L=\begin{pmatrix} A & B & A & A\oplus B\\ B & A & A\oplus B & A\\ A & A\oplus B & A & B\\ A\oplus B & A & B & A\end{pmatrix}. \tag{2.9}$$

As usually, in the present MAC, H_t will represent the internal chaining variable, $|H_t|=256$, sometimes we divide it into two parts $H_t^{(1)}$ and $H_t^{(2)}$, $H_t=(H_t^{(1)},H_t^{(2)})$, $|H_t^{(1)}|=|H_t^{(2)}|=128$. IV is a value for the initialization, $h(x)$ is the hash value with the size of 128 bits. Denoted by x_t the message in the time $t,(t>0)$, which with size of 128 bits and 256 bits in the versions 1, 2 respectively. The expression $\overline{x}$ as usual stands for the *complement* of a binary string x.

Version 1:

In this version it is assumed that the input size $|x_t|=128$ and $|K|=|IV|=128$. Suppose that ζ is a 16-bytes string, which is also viewed as a 4×4 matrix of bytes in the ordinary way，and ζ^T is the transposition of the matrix ζ. Define

$$F(\zeta)=L\cdot S(\zeta^T). \tag{2.10}$$

Let $\vartheta_1=F(F(IV)\oplus\overline{IV}),\quad \vartheta_2=F(F(\overline{IV})\oplus IV)$. The recurrence relations of H_t are defined as following,

$$\begin{aligned} &H_0^{(1)}=F(F(F(IV)\oplus K)\oplus\bar{K}),\quad H_0^{(2)}=F(F(F(\overline{IV})\oplus\bar{K})\oplus K),\\ &H_i^{(1)}=F(H_{i-1}^{(1)}\oplus x_i),\qquad H_i^{(2)}=F(H_{i-1}^{(2)}\oplus H_i^{(1)}),\quad 1\le i\le t,\\ &h(x)=F(F(H_t^{(2)}\oplus\vartheta_1)\oplus\vartheta_2). \end{aligned} \tag{2.11}$$

Version 2:

Now it is assumed that the input size $|x_t|=256$, and $|K|=|IV|=256$. For a 32-bytes string ζ, we define a bytes permutation ϕ: $\zeta^{\phi}=\phi(\zeta)$, $\zeta^{\phi}[i]=\zeta[4i \bmod 31]$, for $0\le i<31$, and $\zeta^{\phi}[31]=\zeta[31]$. A 32-bytes string ζ may be viewed as a 8-words string in ordinary way, and so the transformation L may take on the each word of ζ. Define

$$G(\zeta) = L \cdot S(\zeta^{\phi}). \tag{2.12}$$

Let $\tilde{K} = K[16,31]_{byte} \mid K[0,15]_{byte}$, $\theta = G(G(G(\overline{IV}) \oplus K) \oplus \tilde{K})$, the recurrence relations of H_t are defined as following,

$$\begin{aligned} &H_0 = G(G(G(IV) \oplus K) \oplus \tilde{K}), \\ &H_i = G^2(H_{i-1} \oplus x_i), \qquad 1 \le i \le t, \\ &h(x) = \omega_1 \oplus \omega_2. \end{aligned} \tag{2.13}$$

where $\omega = (\omega_1, \omega_2) = G^2(H_t \oplus \theta),\ |\omega_1| = |\omega_2| = 128.$

3. Security Analysis

In the following we will give a discussion about some possible attacks.

Pre-image attack and 2nd pre-image attack

We know that a secure MAC should be forgery-proof, so it should be pre-image resistant, 2nd pre-image resistant, and collision resistant. For the presented MAC, the transformations F and G are bijective maps and also key-defined, so it is easy to know that in both versions the hash functions $h(x)$ are pre-image resistant. In regard with the 2nd pre-image resistant, we give the following observations for how to get a 2nd pre-image by a internal collision for the version 1, 2 respectively.

Suppose that $H_t(y) = H_t(x)$ and assume that $x_i = y_i, 1 \le i < k \le t,\ x_k \ne y_k$, without loss the generity, assume that $k = t-1$ and $k = 1$. Let $y_1 = x_1 \oplus \varepsilon$, $y_2 = x_2 \oplus \sigma$.

For the version 1, it has that

$$H_2^{(1)}(y_2) \oplus H_2^{(1)}(x_2) = H_1^{(2)}(y_1) \oplus H_1^{(2)}(x_1).$$

For the algebraic equation above there have not been found non-trivial solution for variables y_1, y_2, and the both sides of the equation may take over all block for any $\varepsilon \ne 0,$ or any $\sigma \ne 0$. That is, they are 128-bits unknowns , and so in this way will result a attack same as the exhaustive search attack.

For the version 2, it is clear that

$$\sigma = H_1(x_1) \oplus H_1(y_1).$$

On the other hand, it is easy to know the activated bytes by the compression function $G^2(\zeta)$ will be at least $16\,bytes$ if $0 < |\zeta| < 32$. This means that the range of σ will be as large as

$16\ bytes$ if $|\varepsilon| \le 32$, hence the complexity to match the difference σ will be 128 bits, that same as the exhaustive search attack.
From the investigation, we can know that a good compression function should have sufficiently large diffusion range.

MAC forgery attack
This kind of attack usually is applying the birthday paradox to find an internal collision, and then made a MAC forgery by the found collision. In this aspect, Bart Preneel and P.C. van Oorschot [3] show two generic attacks as following

Proposition 2 Let h be an iterated MAC with n-bit chaining variable and m-bit result. An internal collision for h can be found using u known text-MAC pairs and v chosen texts. The expected values for u and v as following: $u = \sqrt{2} \cdot 2^{n/2}$ and $v = 0$ if output transformation is a permutation; otherwise, v is approximately

$$2 \cdot 2^{n-m} + 2 \cdot \left\lceil \frac{n}{m} \right\rceil .$$

Proposition 3 Let h be an iterated MAC with n-bit chaining variable, m-bit result, a compression function f which behaves like a random function (for fixed x_i), output transformation g. An internal collision for h can be found using u known text-MAC pairs, where each text has same substring of $s \ge 0$ trailing block, and v chosen texts. The expected values for u and v as following: $u = \sqrt{2/(s+1)} \cdot 2^{n/2}; v = 0$ if output transformation is a permutation or $s + 1 \ge 2^{n-m+6}$, and otherwise v is approximating

$$2 \cdot \frac{2^{n-m}}{s+1} + 2 \cdot \left\lceil \frac{n - \log_2(s+1)}{m} \right\rceil .$$

Maybe, it should be mentioned that in the version 1 the compression function is $H_t = (H_t^{(1)}, H_t^{(2)})$, rather than $H_t^{(1)}$ or $H_t^{(2)}$, so to find an internal collision will require about $O(2^{112})$ known text-MAC pairs and $O(2^{96})$ chosen text queries with the bit-length of the texts more than 2^{32}, or require about $O(2^{96})$ known text-MAC pairs and $O(2^{64})$ chosen text queries with the bit-length of the texts more than 2^{64} (taking $s = 2^{32}$ or $s = 2^{64}$ in Proposition 3 respectively), hence the complexity is more than 2^{128} operations.

Correlation attack

The main idea of correlation attack is that by cryptanalysis to find significant advantage of some correlations between the input and output, and then find linear equations of the secret key in a number of statistic tests, and so recover the secret key by solving the found linear system. This kind of attacks may occur in the scenarios of chosen-plaintext attacks and chosen-IV attacks. In LAMA the transformations F and G are key-defined, and the advantage of a linear approximation will be key-dependent, so an adversary is not able to form a definitive correlation attack.

Algebraic attack

As the highest degree of the Boolean functions appeared in the hash function is at least equal to 49, so the number of variables after linearization is about 2^{115}, hence require 2^{108} text-MAC pairs, and so about $O(2^{248})$ operations to solve the linear system. About this kind of attack, in the paper [1] we gave a little more detail discussion.

4. Implementation

The performances of LAMA are in rate about $5\ cycles/byte$ for the both versions(have not included about $64\ cycles$ and $128\ cycles$ spent by the output function $h(x)$ in versions 1, 2 respectively), and with the startup time about of $7000\ cycles$.

Note. If for the simplicity, the permutation matrix P_v in the construction may be simply taken as Id.

References

[1] A.P. Li, A New Stream Cipher: DICING, http://eprint.iacr.org/2006/354

[2] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptograpgy, CRC Press,1997.

[3] Bart Preneel and P.C. van Oorschot, MDx-MAC and Building Fast MACs from Hash Functions, Advances in Cryptology-CRYPTO 1995, LNCS 963, pp 1-14, 1995.