

MAME: A compression function with reduced hardware requirements ^{*}

Hiroataka Yoshida¹, Dai Watanabe¹, Katsuyuki Okeya¹, Jun Kitahara¹,
Hongjun Wu², Ozgul Kucuk², and Bart Preneel²

¹ Systems Development Laboratory, Hitachi, Ltd.,
1099 Ohzenji, Asao-ku, Kawasaki-shi, Kanagawa-ken, 215-0013 Japan

² Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium

Abstract. This paper describes a new compression function, MAME designed for hardware-oriented hash functions which can be used in applications reduced hardware requirements. MAME takes a 256-bit message block and a 256-bit chaining variable as input and produces a 256-bit output. Seen in the light of attacks on MD5 and SHA-1, our design strategy enables us to evaluate the security of MAME against known attacks. Our design considers the security of side channel attacks against HMAC constructions employing MAME. The main use of logical operations in the design allows us to achieve a hardware implementation of MAME requiring 8.1 Kgates on 0.18 μm technology.

Keywords: hash functions, compression functions, low-resource implementation

1 Introduction

Ubiquitous systems have a wide variety of applications such as secure supply-chain automation, proving genuineness of goods, and enhanced privacy of customers. In ubiquitous applications, there are security problems about confidentiality and, more importantly, authentication and privacy. The awareness for cryptographic techniques in ubiquitous systems has been recently increased. However, in order to develop secure ubiquitous systems, cryptographic algorithms must be implemented under restricted source environments, such as low-cost or low-power environments. As for authentication, what has been commonly used is cryptographic hash functions and their applications.

A cryptographic hash function is an algorithm that takes input strings of arbitrary (typically very large) length and maps these to short fixed length output strings. Most hash functions proposed so far are called *iterated* hash functions, which are constructed from a compression function. They work as follows. Let h

^{*} This work was supported in part by a consignment research from the National Institute on Information and Communications Technology (NiCT), Japan. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government.

be a compression function. The message m is padded to a multiple of the block length and subsequently divided into t blocks M_1, \dots, M_t . Then the hash value is taken as H_t , where $H_i = h(H_{i-1}, M_i)$ for $H_0 = IV$ an initial value. The values $\{H_i\}$ are called the chaining variable.

For a cryptographic hash function, the following properties are required:

- **preimage resistance:** it is computationally infeasible to find any input which hashes to any pre-specified output.
- **second preimage resistance:** it is computationally infeasible to find any second input which has the same output as any specified input.
- **collision resistance:** it is computationally infeasible to find a collision, i.e. two distinct inputs that hash to the same result.

For an ideal hash function with an m -bit output, finding a preimage or a second preimage requires about 2^m operations and the fastest way to find a collision is the birthday attack which needs approximately $2^{m/2}$ operations.

In order to satisfy those security requirements, most iterated hash functions use the Merkle-Damgård (MD) strengthening, which fixes IV and appends to a message fixed number of blocks containing the message length.

For the last years, there has been much progress in cryptanalysis of iterated hash functions. Attacks regarding collision resistance have been reported on most widely used iterated hash functions such as MD5 [22] and SHA-1 [18] while certain kind of generic attacks regarding second preimage resistance have been reported on iterated hash functions with the MD strengthening [11].

We argue that the design strategy of hash functions and security evaluation methods must be revisited. As for security, we limit ourselves to collision resistance because the above second preimage attack still requires more complexities than the birthday attack does. One way of viewing the collision attacks mentioned the above is that these attacks exploit weaknesses in the compression functions and essentially apply differential cryptanalysis [5] to find collisions. One could claim that a new hash function is only taken seriously if it is accompanied with evidence that it resists differential cryptanalysis.

What we discussed the above motivates us to develop a new compression function, MAME with strongness in security and compactness in implementation aspect. One can construct light weight hash functions based on MAME by using the MD strengthening or an alternative mode of it with higher security. MAME takes a 256-bit message block as input and a 256-bit chaining variable as input and produces a 256-bit output.

In ubiquitous applications, some protocols with the HMAC constructions [1] could be very useful to solve the authentication problems while side channel attacks on them are of particular importance. It has been pointed out that HMACs constructed from some of block-cipher based hash functions are vulnerable to side channel attacks in [19]. The design of MAME considers the security of HMAC employing this function against this kind of attacks.

The outline of this paper is as follows. In Sect. 2, we give a specification of the MAME compression function. In Sect. 3, we explain our design strategy. In

Sect. 4, we evaluate the security of MAME. We then discuss the performance issue in Sect. 5. Our conclusions are given in Sect. 6.

2 Specification

2.1 Notation

The specification uses the following notations.

\oplus bit-wise exclusive-or	$\ggg n$ n -bit rotation to the right (32 bit register length)
\parallel concatenation	$\lll n$ n -bit rotation to the left (32 bit register length)

2.2 The algorithm of MAME

The MAME compression function denoted by h is constructed from the block cipher f_E defined below in the following manner known as the Matyas-Meyer-Oseas (MMO) mode ([15], pp 340), $h(H, M) = f_E(H, M) \oplus M$.

Overview of the encryption function of the block cipher In the remainder of this paper, we denote the message block by M for simplicity. The encryption function of the block cipher f_E takes as input a 256-bit message block M and the key and then outputs a cipher-text E . The number of rounds of f_E is 96. The structure of the encryption function $f_E(\cdot, \cdot)$ is shown in Figure 1.

The f_E function is broken into three parts to process data, the constant generation function, the key schedule function, the mixing function, each of which uses a sub-function iteratively. Therefore, we denote the corresponding sub-functions to the constant generation function, the key schedule function, and the mixing function, by f_C , f_K , and f_R respectively.

The constant generator takes the initial constant value $c^{(0)}$ and generates a round constant $C^{(r)}$ after r -th round, which becomes an input parameter to the round-key generation function f_K . The key schedule function takes the key as input and generates the round key $K^{(r)}$ after r -th round, which becomes input parameter to the round function f_R . The mixing function takes a message block as input and then outputs a ciphertext.

The mixing function The input variables of f_R are denoted by x_0, x_1, \dots, x_7 . For the plaintext $P = (p_0, p_1, \dots, p_7)$, the ciphertext $E = (e_0, e_1, \dots, e_7)$, the mixing function is defined in the following:

$$\begin{aligned}
 (x_0^{(0)}, x_1^{(0)}, \dots, x_7^{(0)}) &= (p_0, p_1, \dots, p_7), \\
 (x_0^{(r)}, x_1^{(r)}, \dots, x_7^{(r)}) &= f_R(x_0^{(r-1)}, x_1^{(r-1)}, \dots, x_7^{(r-1)}), \\
 &1 \leq r \leq 96, \\
 (e_0, e_1, \dots, e_7) &= (x_0^{(96)}, x_1^{(96)}, \dots, x_7^{(96)}).
 \end{aligned}$$

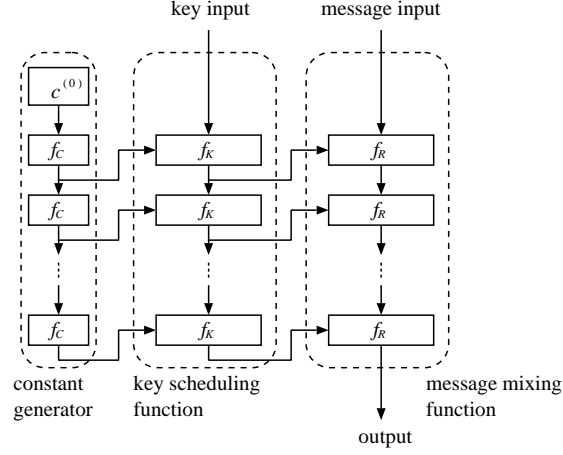


Fig. 1. The structure of the encryption function

The round function f_R consists of word-wise permutation and the non-linear transformation F with 2-word input, taking the round key as the input parameter. For the high word of the output of the F function denoted by F_H and the low word of it denoted by F_L , the round function f_R is defined as follows:

$$\begin{aligned}
 x_0^{(r)} &= x_6^{(r-1)} \oplus F(x_4^{(r-1)} \oplus K^{(r)}, x_5^{(r-1)})_H, \\
 x_1^{(r)} &= x_7^{(r)} \oplus F(x_4^{(r-1)} \oplus K^{(r)}, x_5^{(r-1)})_L, \\
 x_2^{(r)} &= x_0^{(r-1)}, x_3^{(r)} = x_1^{(r-1)}, x_4^{(r)} = x_2^{(r-1)}, \\
 x_5^{(r)} &= x_3^{(r-1)}, x_6^{(r)} = x_4^{(r-1)}, x_7^{(r)} = x_5^{(r-1)}.
 \end{aligned}$$

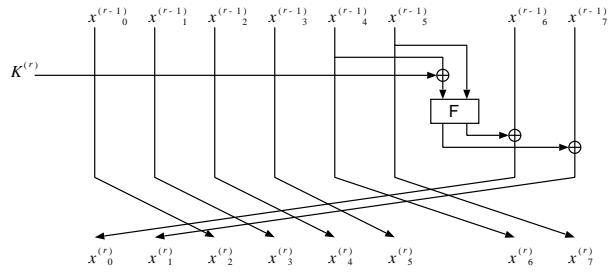


Fig. 2. The round function f_R

The F function consists of the S-box layer \mathcal{S} and the linear diffusion layer \mathcal{L} . The F function is defined by $F = \mathcal{L} \circ \mathcal{S}$.

The S-box layer uses a substitution table S with 4-bit input and 4-bit output, which is defined in the following:

$$S[16] = \{4, 14, 15, 1, 13, 9, 10, 0, 11, 2, 7, 12, 3, 6, 8, 5\}.$$

For the input words of \mathcal{S} denoted by a_H, a_L and the output words of \mathcal{S} denoted by b_H, b_L , the S-box layer \mathcal{S} is defined in the following:

$$b_{H,i+16} || b_{H,i} || b_{L,i+16} || b_{L,i} = S[a_{H,i+16} || a_{H,i} || a_{L,i+16} || a_{L,i}], \quad 0 \leq i < 16.$$

The linear diffusion layer \mathcal{L} is defined in the following:

$$\begin{aligned} b_L &= b_L \oplus (b_H \lll 1), b_H = b_H \oplus (b_L \lll 3), b_L = b_L \oplus (b_H \lll 4), \\ b_H &= b_H \oplus (b_L \lll 7), b_L = b_L \oplus (b_H \lll 8), b_H = b_H \oplus (b_L \lll 14). \end{aligned}$$

The key schedule function f_K has the similar structure to f_R . The difference is that f_K takes as input the key instead of the plaintext and takes the round constant as input parameter, instead of the round key.

$$\begin{aligned} k_0^{(r)} &= k_6^{(r-1)} \oplus F(k_4^{(r-1)} \oplus C^{(r)}, k_5^{(r-1)})_H, \\ k_1^{(r)} &= k_7^{(r-1)} \oplus F(k_4^{(r-1)} \oplus C^{(r)}, k_5^{(r-1)})_L, \\ k_2^{(r)} &= k_0^{(r-1)}, k_3^{(r)} = k_1^{(r-1)}, k_4^{(r)} = k_2^{(r-1)}, \\ k_5^{(r)} &= k_3^{(r-1)}, k_6^{(r)} = k_4^{(r-1)}, k_7^{(r)} = k_5^{(r-1)}. \end{aligned}$$

The r -th round-key $K^{(r)}$ is defined by $K^{(r)} = k_3^{(r)}$.

The generation of round-constants Starting from a fixed initial value $c^{(0)} = 0xcae1ac3f55054a96$, the round-constant generation function generates 64-bit variables $c^{(r)}$'s in the following manner:

$$\begin{aligned} t_H || t_L &= f_L(c^{(r-1)}), \\ c^{(r)} &= t_L || t_H, \end{aligned}$$

where f_L is a Linear feedback shift register (LFSR) defined by the polynomial $g(x)$ over GF(2) described in the Annex. The r -th round constant $C^{(r)}$ is defined as the lower 32-bit of $c^{(r)}$.

3 Design rationale

We set the following three requirements as a goal for our design of MAME.

- It should be easy to perform security analysis of known attacks.

- It should be possible to achieve compact implementations in hardware.
- Its software performance on the CPU's in servers should be good in practice.

To meet the goal, we take the following three items as the design principle.

- Minimize the input/output length while achieving the required security.
- Limit ourselves to relatively simple operations such as XORs to reduce hardware complexity, which makes security assessment less complicated than with most previous hash functions, which use building blocks like arithmetic operations for which the full analysis is hard.
- Ensuring security by increasing the number of rounds, the choice of which considers attacks applying the input/output whitening techniques.

Parameter (input/output) Since the output length of the MAME is 256 bits, the message block length is also at least 256 bits. From hardware implementation point of view, shorter input length implies that the number of required registers is smaller. Therefore we determined that the length of message block is 256 bits.

Structure We adopted the unbalanced Feistel structure for the round function for the lightness compared to SP-like structure.

The mode to construct the compression function from the block cipher The use of the MMO mode in its design makes the security evaluation on MAME similar to the security evaluation on block cipher. It has been pointed out that one can perform side channel attacks on HMACs with hash functions constructed from the Davies-Meyer mode which the most popular hash functions including the SHA-family adopt [19]. We adopt the MMO mode with which HMACs remains to be secure against the above side channel attacks.

The F function The function F is the most significant component in the MAME. To reduce the area requirements, 16 small 4-bit-to-4-bit S-boxes are used. To increase the software performance, those 16 small S-boxes are identical to enable bit slice implementation. The linear diffusion layer uses simple rotations and XORs to reduce hardware and software complexity. It is designed in the way that its branch number is 8.

As for the S-box, we adopted a function which is affine equivalent to the inversion function in $GF(2^4)$ for security reasons. We imposed the restriction that S has no fixed points. The S-box has the properties:

- Maximum differential and linear probabilities are 2^{-2} .
- The degree of the Boolean polynomial of every output bit is 3.
- The number of monomials of polynomial expression over $GF(2^4)$ is 14.

The key schedule function and Constants The key schedule function shares most of the part with the mixing function for reducing hardware complexity and ensuring strong non-linearity. The round constants introduce randomness, non-regularity, and asymmetry into the key schedule function.

4 Security analysis

We evaluate the security of MAME regarding various kinds of widely known attacks on block ciphers, such as the differential attacks, linear attacks, higher order differential attacks, interpolation attacks, and Square attacks. We describe the concepts of the evaluation methods here. We leave some evaluation results in the Annex which we obtained in similar ways to the one described here. In general, our results indicate that MAME has enough security margin against these attacks.

The motivation is that the theory of the PGV construction [21] [3] says that for a compression function constructed from the a block cipher by using a secure PGV mode, which include the MMO mode, if the underlying block cipher behaves as a random function, then the compression function is proved to be secure in terms of collision resistance. We consider that one of the best approaches of verifying randomness of the underlying block cipher of MAME could be that we apply block-cipher analysis tools to MAME and see if we find some weakness or some non-random behavior.

Differential/linear attacks Seeing that the most attacks on hash functions apply differential cryptanalysis and considering that differential/linear cryptanalysis [16] have been most powerful and important cryptanalytic tool in the block cipher analysis, resistance against differential/linear attacks must be ensured. In order to estimate the upper bounds of maximum differential/linear characteristic probabilities, we will apply the following theorem:

Theorem 1. *Let D_{min} and L_{min} be the minimum numbers of total differential/linear active S-boxes. The maximum differential/linear characteristic probabilities are upper bounded by $p_s^{D_{min}}$ and $q_s^{L_{min}}$, respectively, where p_s/q_s called the maximum differential/linear probabilities of S-box are defined as follows:*

$$p_s = \max_{\Delta x \neq 0, \Delta y} \text{Prob}(S(x) \oplus S(x \oplus \Delta(x)) = \Delta(y))$$

$$q_s = \max_{\Gamma y \neq 0, \Gamma x} (2\text{Prob}(x \cdot \Gamma x = S(x) \cdot \Gamma y) - 1)^2$$

Hereafter, we only explain our method of evaluating the security against differential cryptanalysis as we can apply a similar method regarding linear cryptanalysis because of its duality to differential cryptanalysis [6].

In the case of MAME, we estimate the lower bounds of the number of active S-boxes by applying the Viterbi algorithm often used in the error correction codes. This algorithm considers a set of states where each of two states has distance and then, it exhaustively searches for paths with minimum distance. In our case, each state is defined as the intermediate state of f_E after certain round, the distance between a state at round r and a state at round $r + 1$ is measured by the number of active S-box which has been increased through an application of r -th round.

However, we had a problem of too large memory requirement of 2^{256} in the the Viterbi algorithm. To solve this, we truncate the 256-bit input space of f_E

into 20-bit space in the following way. We view the 256-bit input to f_R as four 64-bit words instead of eight 32-bit words. For each of 64-bit words, we truncated data of 4 bits of it into that of 1 bit in the way that at input of F , 4 bits taken into the input to the same S-box are viewed as 1 bit, which truncates the original eight 32-bit words into four 16-bit words, x_0, x_1, x_2, x_3 . For a 64-bit word x_i , by \tilde{x}_i we denote the corresponding 16-bit data of x_i truncated in this way. For such a 16-bit word \tilde{x}_i , $\text{Ham}(\tilde{x})$ ranges from 0 to 16 and it can be represented as a 5-bit string. In the end, we manage to truncate the 256-bit space into the 20-bit space, which results in a practical usage of memory 2^{20} in carrying out the Viterbi algorithm.

Carrying out the Viterbi algorithm requires us to construct some table representing the propagation of the weight of the differences through F which is shown in Table 1 where 0 indicates the corresponding transition of Hamming weight of a difference is not possible, otherwise we put 1. For the row i , the column j , the element $a_{i,j}$ in Table 1 is determined in the following way:

- Case 1 ($i \leq 6$): For any 64-bit x such that $\text{Ham}(x) = i$, compute $\text{Ham}(\widetilde{\mathcal{L}(x)})$. If there exists x such that $\text{Ham}(\widetilde{\mathcal{L}(x)}) = j$, then let $a_{i,j}$ be 1. Otherwise let $a_{i,j}$ be 0.
- Case 2 ($j \leq 6$): For any 64-bit y such that $\text{Ham}(y) = j$, compute $\text{Ham}(\widetilde{\mathcal{L}^{-1}(y)})$. If there exists y such that $\text{Ham}(\widetilde{\mathcal{L}^{-1}(y)}) = i$, then let $a_{i,j}$ be 1. Otherwise let $a_{i,j}$ be 0.
- Case 3 (Otherwise): Let $a_{i,j}$ be 1.

It took us several hours on a PC to perform experiments for each case. Table 1 tells us that the branch number of \mathcal{L} is equal to 8, which is defined as follows:

Definition 1. *The branch number $B_{\mathcal{L}}$ of linear transformation \mathcal{L} is defined by*

$$B_{\mathcal{L}} = \min_{x \neq 0} (\text{Ham}(\tilde{x}) + \text{Ham}(\widetilde{\mathcal{L}(x)})),$$

where we denote the Hamming weight of y by $\text{Ham}(y)$.

In the Viterbi algorithm, for an input difference, one standard way of computing the Hamming weight of the output difference is to use the branch number. In this way, we estimate that the value D_{min} is more than the required number 131 for MAME reduced to 80 rounds.

In order to improve the precision of estimation, we next used the Table 1 instead of the branch number when we performed the Viterbi algorithm. In addition, we captured information on how the weights of the differences change through two applications of F . We experimentally obtained information on how $\text{Ham}(\widetilde{F \circ F(x)})$ behaves. This limits the possibilities for the output difference of the second application of F , compared to what we expect from the case of single application of F , the Table 1³. During performing the Viterbi algorithm

³ e.g. if $\text{Ham}(\tilde{x}) = 3$ and $\text{Ham}(\widetilde{F(x)}) = 5$, then $\text{Ham}(\widetilde{F \circ F(x)}) = 3$ is not possible

if an output difference of the first application is not influenced at XOR which is processed just after F , we can use the above information. In this way, the Viterbi algorithm found us some better result that the value D_{min} is 130 for MAME reduced to 58 rounds.

Table 1. Branch table for differential attacks

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 = $Ham(\mathcal{L}(\hat{x}))$
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1
2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
3	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
4	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
5	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
12	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16= $Ham(\hat{x})$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

As for the linear attack, we obtain the same value for the branch number, 8. We perform similar approach to the case of differential cryptanalysis and we estimate that the value L_{min} is 129 for MAME reduced to 53 rounds.

From the above theorem, we estimate that the maximum differential/linear characteristic probabilities are upper bounded by 2^{-260} and 2^{-258} , respectively. It follows that there is no effective differential/linear characteristic for MAME reduced to 58 rounds.

A Dedicated Differential Attack Here we perform certain kind of dedicated differential attack. We assign four 64-bit words of the internal state at round r , y_0, y_1, y_2, y_3 . Suppose that $F(\Delta_0)=\Delta_1, F(\Delta_1)=\Delta_2, F(\Delta_2)=\Delta_3, F(\Delta_3)=\Delta_0$. We can obtain the 15-round difference propagation depicted in Table 2. The probability for the above 15-round differential is $p_0^3 \times p_1^2 \times p_2^2 \times p_3$. For 60 rounds, the differential probability is

$$p_0^3 \times p_1^2 \times p_2^2 \times p_3 \times p_1^3 \times p_2^2 \times p_3^2 \times p_0 \times p_2^3 \times p_3^2 \times p_0^2 \times p_1 \times p_3^3 \times p_0^2 \times p_1^2 \times p_2.$$

So far we found 14,045 differences $\Delta_i (i = 0, 1, 2, 3)$, satisfying the above conditions. And we only deal with the relatively simple cases where there are only 3 active S-boxes in Δ_0 and 3 active S-boxes in Δ_2 , and at most 15 active

Table 2. Difference propagation for 15 rounds

round	y_0	y_1	y_2	y_3
0	0	0	0	Δ_0
4	Δ_1	0	0	Δ_0
7	Δ_2	0	Δ_0	Δ_1
10	Δ_3	Δ_0	0	Δ_2
12	Δ_1	Δ_2	Δ_3	Δ_0
15	0	0	0	Δ_1

S-boxes in Δ_1 , and at most 15 in Δ_3 . We obtained so many paths in order to find out how many multiple differential paths are there. The result is that we found at most 6 multiple differential paths among those 14,045 paths.

But the probability for each of the differential path is very small. Each set of Δ_i ($i = 0, 1, 2, 3$) involves at least 34 active S-boxes. The probability of each differential path for 60 rounds is less than 2^{-700} , which shows MAME has a large security margin against this kind of attacks.

Higher Order Differential Attack In the higher order differential attacks [12], the attacker constructs Boolean polynomial expression for the cipher to be attacked. In the encryption process, each bit of each intermediate state can be expressed as a Boolean polynomial in terms of bits of the plaintext.

The idea of the attack is that if the intermediate bits are expressed by Boolean polynomials of degree at least d , the $(d+1)$ -th order differential in polynomial sense of the Boolean polynomial would be 0. Therefore if the value d is small enough, the attack would be feasible.

In the case of MAME, we found that every output bit of the S-box S can be expressed as a Boolean polynomial of degree 3 in terms of input bits. One naive approach for a higher order differential attack is to construct a Boolean polynomial of degree 256 for the 256-bit block cipher in MAME by assigning one variable for 1 bit. However, the attacker could construct a more simple expression of smaller degree by substituting 0 into certain variables, which makes various possibilities for the variables in the polynomial expression.

We performed experiments dealing with all of these possibilities in order to observe how the S-box applications increase the degree of Boolean functions as the number of rounds are increased. We confirmed that the degree of such polynomials for MAME with 21 rounds reaches to the required degree, which depends on how many variables the polynomial has. This prohibits higher order differential attacks on the full rounds of MAME.

4.1 Analysis of the iterated hash function based on MAME with the MD strengthening

In order to use MAME in practice, we specify certain iterated hash function based on MAME with the MD strengthening with the 256-bit initial vector

$H_0 = (H_{0,0}, H_{0,1}, \dots, H_{0,7})$ which is given in the following:

$H_{0,0} = 0xbc18bf6d, H_{0,1} = 0x369c955b, H_{0,2} = 0xbb271cbc, H_{0,3} = 0xdd66c368,$
 $H_{0,4} = 0x356dba5b, H_{0,5} = 0x33c00055, H_{0,6} = 0x50d2320b, H_{0,7} = 0x1c617e21.$

We investigate the security of the hash function against the collision attacks by Wang *et al.* In the collision attacks on MAME, choosing the message input to MAME corresponds to choosing the plaintext to the underlying block cipher. For any differential characteristic the attacker finds, its differential probability is upper bounded by 2^{-256} . The attacker next tries to build a system of equations for this characteristic which is called sufficient conditions and then tries to satisfy them by controlling the chaining variable input and the message block input. However, direct control over the chaining variable input should be very difficult because the key schedule input is the output of the previous application of MAME. Therefore, all the attacker can control should be 256 bits of plaintext with which we consider it is very difficult in order to fulfill the conditions. Therefore, we consider the attacks by Wang *et al* is very unlikely to be feasible to the MAME based hash function specified here.

4.2 Regularity Analysis of Reduced MAME

It has been pointed out in [2] that if a hash function has some irregularity then the complexity for the birthday attack for finding collision would be less than expected. We examine the regularity of reduced versions of MAME.

Let $h : D \rightarrow R$ be a hash function where the range R contains $r \geq 2$ points R_1, \dots, R_r . For $i = 1, \dots, r$ let $h^{-1}(R_i) = \{x \in D | h(x) = R_i\}$ be the pre-image of R_i under h . Let $d_i = |h^{-1}(R_i)|$ and $d = |D|$ be the cardinality of this pre-image set and the domain respectively. Balance of h is defined as $\mu(h) = \log_r \left[\frac{d^2}{d_1^2 + \dots + d_r^2} \right]$.

Let $C_h(q)$ be the probability that the birthday attack on hash function h succeeds in finding a collision in q trials. Then by [2]: $C_h(q) = \binom{q}{2} \frac{1}{r^{\mu(h)}}$. A collision is expected in about $r^{\mu(h)/2}$ trials. We calculated μ values for three different reduced versions such as with a linear transformation (as in MAME), without a linear transformation, and finally with an MDS matrix.

Table 3. μ values for MAME-32 with different diffusion layers

Rounds with no diffusion	similar diffusion	with MDS matrix
96	0.938165	0.968750
		0.968751

MAME-32 without a diffusion layer has lower balance. As we can observe from Table 3, regularity with a diffusion layer consisting of shifts and XORs does not differ from the one with an MDS matrix and it is reasonably high.

5 Performance

5.1 Hardware performance

The use of logical operations in the most part of the design allows us to achieve a hardware implementation of MAME requiring 8.1 Kgates on 0.18 μm technology. In the implementation, f_K and f_R share the same circuit and processing one round takes one cycle. We present our hardware implementation comparison of MAME with SHA-256 [18] as shown in Table 4.

Table 4. Comparison of hardware implementation of MAME with SHA-256

Algorithm	Area (KGates)	throughput (Mbps)	Max Frequency (MHz)
MAME	8.1	440	300
SHA-256	18.0	2600	300

5.2 Software performance

We present our software implementation of MAME on some microcomputer made by Renesas for smart cards. In software implementation, we partially unroll the round functions code to increase the speed and we take the approach described in [20] to achieve a bit slice implantation where S-box is transformed into 20 logical operations. We present our software implementation comparison of MAME with SHA-256 as shown in Table 5.

Table 5. Comparison of software implementation of MAME with SHA-256

Algorithm	Time (ms)	RAM (Bytes)
MAME	49.4	96
SHA-256	31.4	128

6 Conclusion

We presented a new compression function, MAME designed for a hardware-oriented hash function. We make it clear what the design rational we adopt and evaluate its security applying techniques from block cipher analysis and confirm that there is no weakness in MAME. Our implementation shows some sort of compactness of MAME but this leaves room for further optimizations.

References

1. M. Bellare, R. Canetti, H. Krawczyk, "Keying Hash Functions for Message Authentication," *Advances in Cryptology - CRYPTO 96*, LNCS1109, (1996), 1-15.
2. M. Bellare, T. Kohno, "Hash Function Balance and Its Impact on Birthday Attacks," *Advances in Cryptology- Eurocrypt 2004*, Springer-Verlag, LNCS 3027, (2004).
3. J. Black, P. Rogaway, and T. Shrimpton, "Black-box analysis of the block cipher-based hash-function constructions from PGV," *Advances in Cryptology - CRYPTO 2002*, Springer-Verlag, LNCS 2442, (2002), 320-335.
4. A. Biryukov, D. Wagner, "Advanced slide attacks," in *Proceedings of Eurocrypt 2000*, LNCS 1807, B. Preneel, Ed., Springer-Verlag, pp. 589-606, 2000.
5. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
6. F. Chabaud and S. Vaudenay, "Links between Differential and Linear Cryptanalysis," in *Proceedings of Eurocrypt '94*, LNCS 950, Springer-Verlag, pp. 356-365, 1995.
7. J. Daemen, L.R. Knudsen and V. Rijmen, "The block cipher Square," *Fast Software Encryption*, LNCS 1267, E. Biham, Ed., Springer-Verlag, 1997, pp. 149-165. Also available as <http://www.esat.kuleuven.ac.be/rijmen/square/fse.ps.gz>.
8. I. Damgård, "A design principle for hash functions," in *Proceedings of Crypto'89*, LNCS 435, G. Brassard, Ed., Springer-Verlag, pp. 416-427, 1990.
9. B. R. Gladman,
http://fp.gladman.plus.com/cryptography_technology/
10. T. Jakobsen, L. R. Knudsen, "The interpolation attack on block ciphers," in *Fast Software Encryption*, Israel, LNCS 1267, pp. 28-40, Springer-Verlag, 1997
11. J. Kelsey, B. Schneier, "Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work," in *Advances in Cryptology-Eurocrypt'2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474-490, Springer-Verlag, 2005.
12. L.R. Knudsen, "Truncated and Higher Order Differentials," *Proceedings of the Second International Workshop on Fast Software Encryption*, Leuven, Belgium, 1995, LNCS 1008, Springer, pp.196-211.
13. C. Kocher, J. Jaffe, B. Jun, "Differential Power Analysis," *Advances in Cryptology - CRYPTO 99*, LNCS1666, (1999), 388-397.
14. K. Lemke, K. Schramm, C. Paar, "DPA on n-Bit Sized Boolean and Arithmetic Operations and Its Application to IDEA, RC6, and the HMAC Construction," *Cryptographic Hardware and Embedded Systems (CHES 2004)*, LNCS3156, (2004), 205-219.
15. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
16. M. Matsui, "Linear Cryptanalysis Method for DES cipher," in *Proceedings of EUROCRYPT'93*, LNCS 765, pp.386-397, 1994.
17. T.S. Messerges, E.A. Dabbish, R.H. Sloan, "Investigations of Power Analysis Attacks on Smartcards," *USENIX Workshop on Smartcard Technology* (1999).
18. National Institute of Standards and Technology, FIPS-180-2: "Secure Hash Standard (SHS)," August 2002.
19. K. Okeya, "Side Channel Attacks against HMACs based on Block-Cipher based Hash Functions," *ACISP 2006 Conference, Proceedings*, pp. 317-329, 2006.
20. D. A. Osvik, "Speeding up Serpent," *The 3rd AES Conference, Proceedings*, pp. 317-329, 2000.

21. B. Preneel, R. Govaerts, and J. Vandewalle, "Hash functions based on block ciphers: A synthetic approach," *Advanced in Cryptology, CRYPTO 93*, Springer-Verlag, LNCS 773, (1994), 368-378.
22. R. Rivest, "The MD5 message-digest algorithm," Request for Comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force, April 1992.
23. X. Wang, Y. L. Yin, H. Yu, "Finding collisions in the full SHA1," in *Proceedings of CRYPTO 2005*, LNCS 3621, V. Shoup, Ed., Springer-Verlag, pp. 17-36, 2005.
24. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," in *Proceedings of Eurocrypt 2005*, LNCS 3494, R. Cramer, Ed., Springer-Verlag, pp. 17-18, 2005.

7 Appendix

7.1 Security analysis

Interpolation attack In the interpolation attacks [10], the attacker constructs polynomials (typically over some finite field) expressing the cipher to be attacked by using pairs of plaintext and ciphertext. The idea of the attack is that if the degree of constructed polynomial is small, required plaintexts and ciphertexts are a few in order to solve for the coefficient depending on the key in the polynomial.

In the case of MAME, the S-box S can be expressed for as a polynomial over $\text{GF}(2^4)$. By applying the Lagrange interpolation technique, we found such a polynomial expression of degree 14 for S .

If we assign one variable for each 4 bits for MAME, we could construct a polynomial expression with 64 variables over $\text{GF}(2^4)$. The attacker could construct a more simple expression by substituting 0 into certain variables.

We performed experiments dealing with such attacking scenario and we confirmed that the degree of such polynomials increases to more than 255. This prohibits interpolation attacks on more than 18 rounds of MAME.

Square attack The Square attack has been developed to evaluate the security of the byte-oriented ciphers such as Square and AES [7]. Here we analyse the block cipher in MAME by applying this technique. The attack introduces the following terms. The i th byte is *passive* if and only if values of all i th byte in the collection of texts are equal. The i th byte is *active* if and only if values of all i th byte in the collection of texts are different. The i th byte is *balanced* if and only if the sum of all i th byte is 0. The byte which is not categorized to be any of these bytes is called *unbalanced*. In the attack on reduced-round AES, starting from a collection of texts with one active byte, the attacker obtains balance bytes after several rounds, which result in constructing a distinguisher leading to a successful attack.

In the case of MAME, we make 64-bit words play the same role as bytes do in the Square attacks on reduced AES. In this way, we have 4 different word positions in each intermediate states hence we have 2^4 states for plaintext, depending on the positions where words are active or passive. We confirmed that starting from any of those states, any word becomes unbalanced after 17 rounds

of MAME. Therefore we consider that the square attack is very unlikely to be feasible to the full round MAME.

7.2 Round constants

For the constant generation function, the following polynomial $g(x)$ over $\text{GF}(2)$ defining the Linear feedback shift register (LFSR) f_L is given:

$$\begin{aligned} g(x) = & x^{63} + x^{62} + x^{58} + x^{55} + x^{54} + x^{52} + x^{50} + x^{49} + x^{46} + x^{43} \\ & + x^{40} + x^{38} + x^{37} + x^{35} + x^{34} + x^{30} + x^{28} + x^{26} + x^{24} \\ & + x^{23} + x^{22} + x^{18} + x^{17} + x^{12} + x^{11} + x^{10} + x^7 + x^3 + x^2 + 1 \end{aligned}$$

Round constants $C^{(0)}, \dots, C^{(95)}$, are shown as follows:

Table 6. Round constants

$C^{(0)} = 0x51151113$,	$C^{(1)} = 0x3b4f5a2f$,	$C^{(2)} = 0x2b0e343a$,	$C^{(3)} = 0x46b151a6$,
$C^{(4)} = 0xac38d0e9$,	$C^{(5)} = 0xde130ff4$,	$C^{(6)} = 0x1b6f7abf$,	$C^{(7)} = 0xbc9a76bc$,
$C^{(8)} = 0xc631d3e6$,	$C^{(9)} = 0xf269daf1$,	$C^{(10)} = 0xdc1106f5$,	$C^{(11)} = 0xa6fd1bb3$,
$C^{(12)} = 0x1f1e6ba2$,	$C^{(13)} = 0x307857d6$,	$C^{(14)} = 0x7c79ae88$,	$C^{(15)} = 0xc1e15f59$,
$C^{(16)} = 0x3530f34d$,	$C^{(17)} = 0x68df0d12$,	$C^{(18)} = 0x7f4ff42f$,	$C^{(19)} = 0x67aa7d25$,
$C^{(20)} = 0x9265a0cb$,	$C^{(21)} = 0xf1f384e2$,	$C^{(22)} = 0xe21aba37$,	$C^{(23)} = 0x03185ae5$,
$C^{(24)} = 0xe73098aa$,	$C^{(25)} = 0xa7ed528f$,	$C^{(26)} = 0x58142bc4$,	$C^{(27)} = 0x34397327$,
$C^{(28)} = 0xa486e67c$,	$C^{(29)} = 0x7b69f586$,	$C^{(30)} = 0x921b99f1$,	$C^{(31)} = 0x29719f74$,
$C^{(32)} = 0xe3e25ede$,	$C^{(33)} = 0xa5c67dd1$,	$C^{(34)} = 0x4b5f3214$,	$C^{(35)} = 0x3c95ce5f$,
$C^{(36)} = 0xe9aa813c$,	$C^{(37)} = 0x59db0067$,	$C^{(38)} = 0x627c4d9d$,	$C^{(39)} = 0x083671eb$,
$C^{(40)} = 0xe6ab4602$,	$C^{(41)} = 0x8b55feb7$,	$C^{(42)} = 0x5e7b5164$,	$C^{(43)} = 0x86dbc3c7$,
$C^{(44)} = 0xbd3b0cfc$,	$C^{(45)} = 0xb0e33606$,	$C^{(46)} = 0xf4ec33f0$,	$C^{(47)} = 0xc38cd819$,
$C^{(48)} = 0x176686ad$,	$C^{(49)} = 0x61691012$,	$C^{(50)} = 0xf61623af$,	$C^{(51)} = 0x41720925$,
$C^{(52)} = 0xb702fecb$,	$C^{(53)} = 0x6a9254e2$,	$C^{(54)} = 0x7787c237$,	$C^{(55)} = 0x6e9f1ae5$,
$C^{(56)} = 0xb14578ab$,	$C^{(57)} = 0xd5261be2$,	$C^{(58)} = 0x6e99dbb7$,	$C^{(59)} = 0x904e26e5$,
$C^{(60)} = 0xd53d1eaa$,	$C^{(61)} = 0xeab4a28f$,	$C^{(62)} = 0x902233c5$,	$C^{(63)} = 0xc588fa4a$,
$C^{(64)} = 0xeb04f60f$,	$C^{(65)} = 0xd2f5a045$,	$C^{(66)} = 0xc349a84b$,	$C^{(67)} = 0x248cf163$,
$C^{(68)} = 0x627cd15a$,	$C^{(69)} = 0x39bffc97$,	$C^{(70)} = 0x4d250c04$,	$C^{(71)} = 0x4d73cb47$,
$C^{(72)} = 0xf042797d$,	$C^{(73)} = 0x5a955d6b$,	$C^{(74)} = 0xae539583$,	$C^{(75)} = 0x050f05da$,
$C^{(76)} = 0x12c26f16$,	$C^{(77)} = 0x143c1768$,	$C^{(78)} = 0x4b09bc58$,	$C^{(79)} = 0x50f05da1$,
$C^{(80)} = 0xe8f0b80d$,	$C^{(81)} = 0x2c9b06f3$,	$C^{(82)} = 0xcc989042$,	$C^{(83)} = 0x19e022d7$,
$C^{(84)} = 0xf6b40864$,	$C^{(85)} = 0xcc0cb247$,	$C^{(86)} = 0x1e0668fd$,	$C^{(87)} = 0x5f68b96a$,
$C^{(88)} = 0xd3959aef$,	$C^{(89)} = 0xb974acc5$,	$C^{(90)} = 0x210c1bca$,	$C^{(91)} = 0x4e5e8a0e$,
$C^{(92)} = 0x84306f29$,	$C^{(93)} = 0xfdac6154$,	$C^{(94)} = 0xbb4d85bf$,	$C^{(95)} = 0x3267cc3c$.