Meet-in-the-Middle Attack on QARMA Block Cipher

Rui Zong[1] and Xiaoyang Dong[1]

Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education,
ShandongUniversity, China
{zongrui,dongxiaoyang}@mail.sdu.edu.cn

Abstract. QARMA is a recently published lightweight tweakable block cipher, which has been used by the ARMv8 architecture to support a software protection feature. In this paper, using the method of MITM, we give the first distinguisher of QARMA block cipher. It is made up of the *Pseudo-Reflector* construction with two forward rounds and three backward rounds. By adding two rounds on the top and three rounds on the bottom of the distinguisher, together with the idea of the differential enumeration technique and the key-dependent sieve skill, we achieve a 10-round (of 16-round) key recovery attack with memory complexity of 2^{116} 192-bit space, data complexity of 2^{53} chosen plaintexts and time complexity of $2^{70.1}$ encryption units. Furthermore, we use the same distinguisher to attack QARMA-128 which also includes 10 (of 24) round functions and the *Pseudo-Refector* construction. The memory complexity is 2^{232} 384-bit space, the data complexity is 2^{105} chosen plaintexts and the time complexity is $2^{141.7}$ encryption units. These are the first attacks on QARMA and do not threaten the security of full round QARMA.
Keywords: QARMA, Lightweight Tweakable Block Cipher, Meet-in-the-Middle Attack

1 Introduction

Symmetric cryptography, usually including block cipher, stream cipher and hash function, is the earliest known method to provide confidentiality and integrity of susceptible data. Hash functions are utilized widely in various applications, such as digital signatures, to guarantee integrity. In 2005, the breakthrough works by Xiaoyun Wang, Hongbo Yu [5,6,7,8] on several international hash standards, including MD5 [3], SHA-1 [4] et al, make the cryptanalysis and design of hash algorithm a hot spot. As another symmetric cipher, block ciphers guarantee the confidentiality of data. The most widely known two algorithms are Data Encryption Standard(DES) [22] and Advanced Encryption Standard(AES) [14]. In the early 1970s, DES was proposed at IBM and submitted to the National Bureau of Standards. Finally in 1977, it was published as an official FIPS for the United States. But unfortunately, DES is not considered to be secure as its 56-bit key size could not satisfy the security requirements for many applications. So in January 1997, the US National Institute of Standards and Technology(NIST) announced the start of an initiative to develop a new encryption standard: the AES. This standard would become a new FIPS, taking the place of the old Data Encryption Standard(DES) and triple-DES. Finally in October 2000, NIST officially declared that Rijndael to be the AES. Till now, AES is widely used and attracting many researchers.

Recently, a security concern when using a block cipher attracts the attention of researchers: when using the same key to encrypt the same message in different cases, one get the same ciphertext, which means that the attacker knows that the messages are the same when getting the same ciphertexts. So if the attacker get the message in one case, he can detect messages in other cases. To solve this problem, in 2002, the concept of tweakable block cipher were proposed by Moses Liskov, Ronald L. Rivest and David Wagner [9]. The encrypt process of tweakable block cipher can be expressed as: $C = E_{key}(P, Tweak)$ while that of the traditional is: $C = E_{key}(P)$. Addition to the plaintext and the key, a "tweak" T is needed to yield a ciphertext. The "tweak" should be easily changed, and even public.

QARMA[10] is a lightweight tweakable block cipher proposed by Roberto Avanzi designed for usage in special hardware cases, such as memory encryption, generation of very short tags by truncation. It has two versions: QARMA-64 uses a 64-bit state with a 128-bit key, the other one: QARMA-128 uses a 128-bit state with a 256-bit key. In 2016[11], the QARMA block cipher has been introduced by the ARMv8-A architecture to be used in a software protection called Pointer Authentication Code(PAC).

Meet-in-the-Middle(MITM) attack was first proposed by Diffle and Hellman[12]. The basic process is as follows: a block cipher is firstly divided into two parts: $E_K = E_{K_1} \circ E_{K_2}$. Given a plaintext and its ciphertext (P, C), an adversary guesses the value of (K_1, K_2) and checks whether $E^1_{K_1}(P) = (E^2_{K_2})^{-1}(C)$. If the equality holds, the guessed key might be the right key; otherwise, it must be a wrong key. Demirci et al.[13] in 2008 reformed the standard MITM attack to AES[14]. They treated the block cipher E as $E_K = E^2_{K_2} \circ E^m \circ E^1_{K_1}$. In this strategy, the adversary needs to build a distinguisher to describe the cryptographic property of E^m associated with the so-called δ-set: a set including 2^8 states, where a byte traverses all values and the other bytes are constants. If one considers encryptions expressed as a function of a δ-set: $(X^0, \cdots, X^{255})$ for the input of E^m, then the output value: $(E^m(X^0), \cdots, E^m(X^{255}))$ can be expressed as a function of some intermediate variables of X^i and partial subkeys involved in E^m. Similarly as in [15], we symbol the sequence $(E^m(X^0), \cdots, E^m(X^{255}))$ as S, and all intermediate variables along with the key information as V. If the total number of bits of S is small enough, then the distinguisher will work, and the adversary pre-compute all values of S and stores them in a table H. Then at ASIACRYPT 2010[16], Dunkleman, Keller and Shamir develop new ideas to reduce the memory complexity. One of these ideas used in this paper is the differential enumeration technique which uses a special property on a truncated differential trail to reduce the number of parameters in V. They also set up a correlation in V and a truncated differential characteristic D with average probability. That means if a pair conforms to D, where they assume one pair of message belongs to the δ-set:$(X^0, \cdots, X^{255})$, then the possible values of V will be restricted to a small subset of the value space. At FSE 2014, Li et al.[17] introduced the key-dependent sieve technique, which filters wrong states based on the key schedule to further reduce the complexity in the precomputation phase. In ASIACRYPT 2014[18], Jian Guo studied the security of Generic Feistel Constructions against meet-in-the-middle attack and gave insights on lower bounds on the number of rounds a secure Feistel should have.

Our Contribution. QARMA block cipher has a sophisticated central construction which includes two complete round functions and a "*Pseudo-Reflector*" construction that involves half of the master key. We present a distinguisher on QARMA including 5 round functions and the *Pseudo-Reflector* construction. Based on this distinguisher, we mount an MITM attack on 10-round QARMA-64 with complexities of 2^{116} 192-bit space, 2^{53} chosen plaintexts and $2^{70.1}$ encryption units. Then we use the same distinguisher to attack QARMA-128: a 10-round attack with memory complexity of 2^{232} 384-bit space, data complexity of 2^{105} chosen plaintexts, time complexity of $2^{141.7}$ encryption units.

Outline. The rest of this paper is organized as follows. We start with a brief introduction of QARMA block cipher and some definitions and properties in Section 2. In Section 3, we give our distinguisher and the specific process of the attack on QARMA-64 and QARMA-128. In Section 4, we conclude this paper.

2 Preliminaries

In this section we represent a brief introduction of QARMA and then give the definitions and propositions used throughout this paper.

2.1 Brief Description of QARMA

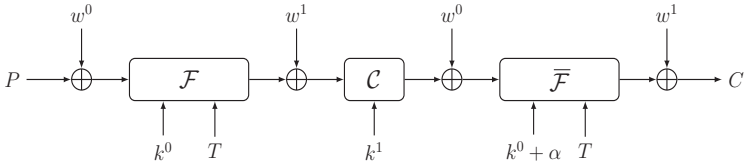

Fig. 1. The Overall Scheme

QARMA[10] is a three-round Even-Mansour construction (Figure 1) where the permutations are parameterized by a core key, and all round keys are derived from the a whitening key. The first and third permutations are functionally the inverse of each other. Different rounds are further parameterized by a tweak. A more detailed introduction is depicted in Figure 2. In this paper, a bar over an operation or a function (e.g. $\bar{\mathrm{F}}$) denotes its inverse.

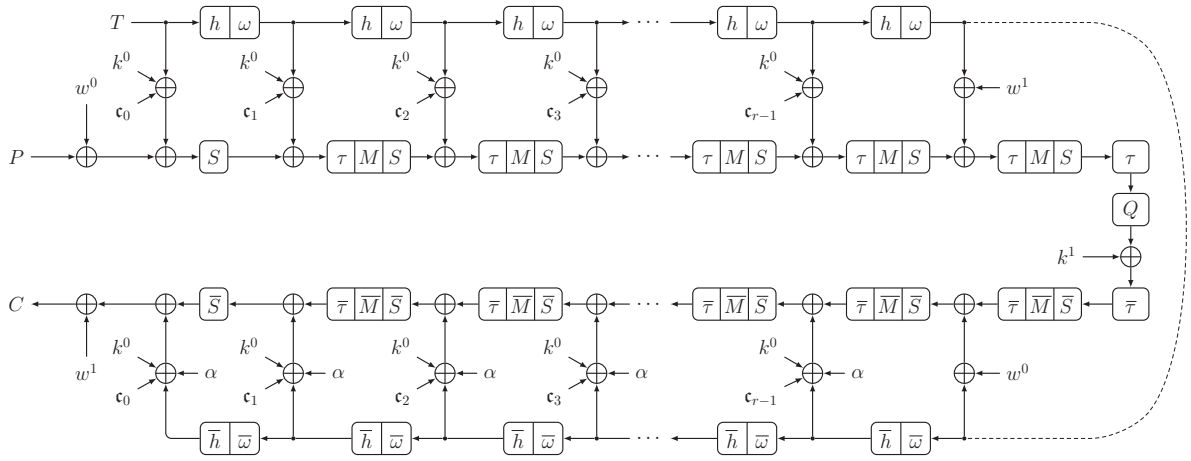

Fig. 2. The Structure of QARMA

Following we will only focus on the 64-bit version of QARMA as the difference of the two versions of QARMA do not influence the attack process. For more details, we refer to [10].

The 128 bit master key K is firstly split into two parts, $w^0 \| k^0$, where w^0 and k^0, the whitening and core keys respectively are both 64 bits. For encryption, the keys are specialised as:$w^1 = o(w^0) = (w^0 \ggg 1) + (w^0 \gg (63))$ and $k^1 = k^0$.

QARMA-64 is a bricklayer SPN with 14 rounds and the central construction(two round functions and a *Pseudo-Reflector* construction). The encryption process can be seen as a sequence of operations on the 64-bit internal state together with a tweak and the key. Internal state size of QARMA-64 can be represented as sixteen 4-bit cells, which are indexed in big endian order, while the bits in a cell are ordered in little endian order, e.g., a plaintext P can be expressed as:

$$\mathbf{P} = p_0 \| p_1 \| p_2 \| \cdots \| p_{15} = \begin{pmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 & p_7 \\ p_8 & p_9 & p_{10} & p_{11} \\ p_{12} & p_{13} & p_{14} & p_{15} \end{pmatrix}$$

The forward round function F includes the following 4 operations:

KeyAddition(K): The i^{th} 64-bit round key K_i is XORed to the state S with the round tweak t and round constant c_i.

ShuffleCell(τ): This operation is the same as the cell permutation of MIDORI[21], i.e.

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

MixColumn(M): Each column of the cipher internal state array is multiplied by the matrix M, i.e., $S = M \cdot S$. The matrix M is as follows:

$$\begin{pmatrix} 0 & \rho & \rho^2 & \rho \\ \rho & 0 & \rho & \rho^2 \\ \rho^2 & \rho & 0 & \rho \\ \rho & \rho^2 & \rho & 0 \end{pmatrix}$$

Moreover, the matrix M is involutory, i.e. $M^2 = I$. The multiplication of an element of the array with ρ^i is just a simple left circular rotation of the element by i bits.

Table 1. Sbox of QARMA-64

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S(x)	0	14	2	10	9	15	8	11	6	4	3	7	13	12	1	5

SubCell(S): Apply the non-linear 4×4 S-box in parallel on each nibble of the state.

The backward round function is the inverse of the forward round function.

A short version of the forward round function exists in the first forward round and the last backward round which omits the ShuffleCell and Mixcolumn operation.

The central construction is made up of one forward round, one backward round and a *Pseudo-Reflector* constructions. Key bits involved in these two rounds are deduced by the whitening key instead of the core key. The *Pseudo-Reflector* construction includes four parts which is essentially a ShuffleCell-MixColumn-KeyAddition-Inverse ShuffleCell operation. In addition, the matrix Q used here in the MixColumn operation is the same as M.

2.2 Definitions and Propositions

In[15], Li et al. extended the definition of δ-set[14] to T active cells and got T-δ-set. Here we give its definition:

Definition 1 *(T-δ-set). A T-δ-set includes $2^{T\times 4}$ values which traverses all values of T active nibbles while keeps the values of the other nibbles constants (the inactive nibbles).*

Super S-box was first used in [19] to analysis AES, similarly, the Super S-box for QARMA-64 is as follows:

Definition 2 *(Super S-box). For each value of the the key bits involved, a QARMA-64 Super S-box maps one column of M_0 to one column of M_4 as shown in Figure 3. It consists of one SubCell, one KeyAddition, one MixColumn and one Inverse SubCell. Note that we swap the order of the KeyAddition operation and the MixColumn operation and $k^1 = M \cdot ruk^1$.*

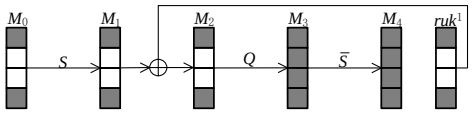

Fig. 3. Super S-box for QARMA-64

Proposition 1 *(Differential Property of S-box,[20]). Given the input and output differences of a S-box, there exists only one pair of actual values on average to satisfy these two differences.*

This proposition is easy to know and also applies to the inverse of S-box and Super S-box.

Proposition 2 *(Differential Property of Super S-box). Given ΔM_0 and ΔM_4 two non-zero differences in $F_{2^{16}}$, once the difference characteristic through the Super S-box is possible, the equation of super S-box:*

$$Super - S(x) \bigoplus Super - S(x \bigoplus \Delta M_0) = \Delta M_4,$$

has two equivalent solutions in average for each involved key value, regardless of the specific location of the active nibbles.

Proof: Without loss of generality, we set $M_0[0,3]$ and $M_4[0,1,2,3]$ the active nibbles and the involved key nibble to be $ruk^1[0,3]$, then the difference characteristic will be shown as in Figure 3. Given $\Delta M_4[0,1,2,3]$ and for each $M_4[0,1,2,3]$, one can get $M_3[0,1,2,3]$ and $\Delta M_3[0,1,2,3]$. After a inverse of the MixColumn operation, the number of pairs that satisfy $\Delta M_2[1,2] = 0$ is about 2^8. Since $ruk^1[0,3]$ is

known, one can get $M_1[0,3]$. Therefore, for each value of $\Delta M_0[0,3]$ and $\Delta M_4[0,1,2,3]$, the average number of pairs satisfying the difference characteristic of Super S-box is $2^{16-8-4\times 2} = 1$ for each corresponding subkey. □

Proposition 4 *(Property of the Mixcolumn Operation[10]). For QARMA-64, the matrix used in the MixColumn operation has 67 column-to-column state transitions.*

All possible transitions is shown in Figure 4.

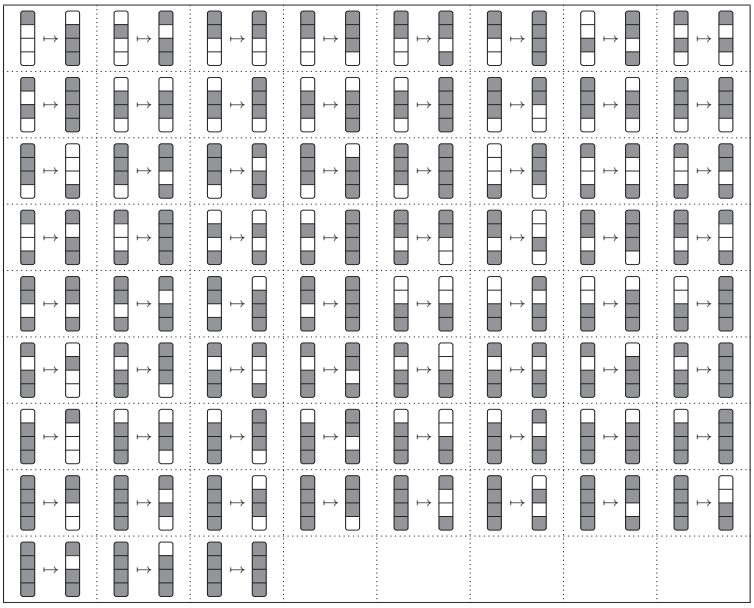

Fig. 4. The Column-wise Active State Transitions for the Matrice in QARMA-64

2.3 The MITM Attack Scheme

In this section, we give a unified description of meet-in-the-middle attack. The integrated attack on QARMA is presented in Section 3.

For the meet-in-the-middle attack, the encryption algorithm can be divided into three consecutive parts: $E_K = E^2_{K_2} \circ E^m \circ E^1_{K_1}$ and there must be a specific property for the middle part E_m, which is used as the distinguisher to get the correct value of (K_1, K_2).

The general attack scheme is made up of two phases: precomputation phase and online phase. During the precomputation phase, we concentrate on building a table T containing all possible output sequences from the input sequences satisfying the differential property of the distinguisher. In the online phase, we encrypt chosen plaintexts and partially decrypt the corresponding ciphertexts to get pairs that satisfy the differential property of the distinguisher by guessing the involved key bits. Finally, we partially decrypt the associated T-δ-set through the third part of E_K to check whether it belongs to T. If so, then the guessed value of the key is right; otherwise, it's wrong.

3 MITM Attacks on Reduced-Round QARMA-64/128

In Section 3.1, we first propose a distinguisher of QARMA-64 that includes the *Pseudo-Reflector* construction together with two forward rounds and three backward rounds. In Section 3.2, we extend the distinguisher to attack 10-round QARMA-64 by extending two rounds on the top and three rounds on the bottom. After that in Section 3.3, we use a similar distinguisher as in Section 3.1 to attack QARMA-128.

3.1 The 5-Round with *Pseudo-Reflector* Distinguisher for QARMA-64

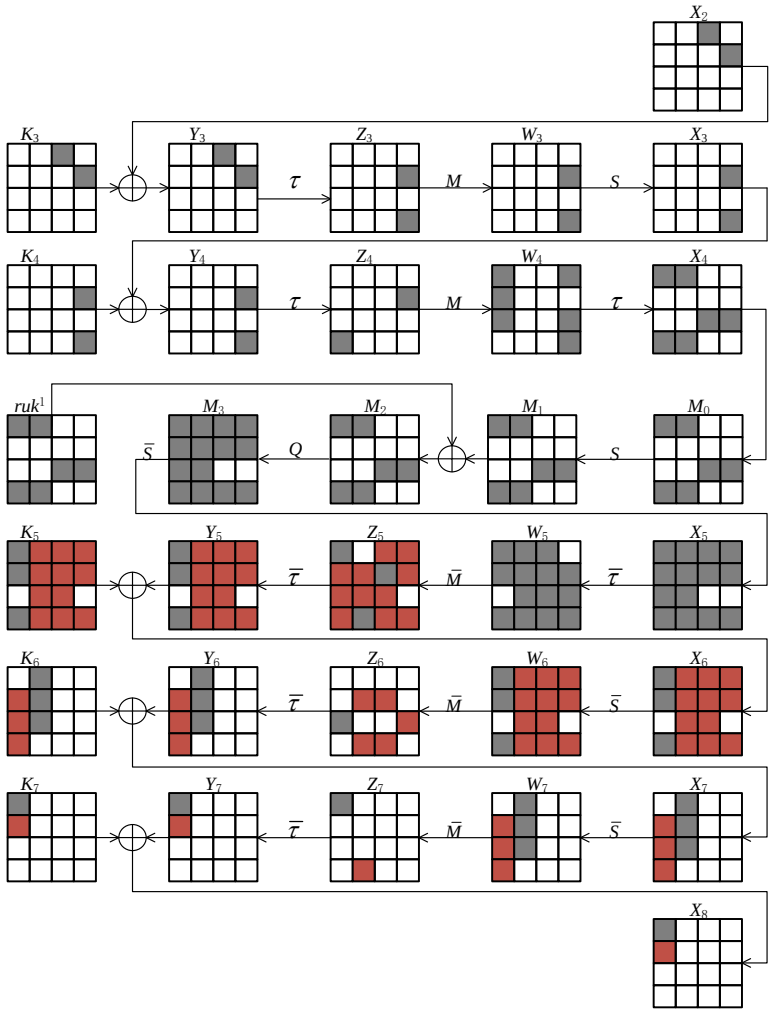

Fig. 5. Distinguisher of 5-Round with *Pseudo-Reflector* on QARMA-64

Using the methodology of meet-in-the-middle, we find a property that covers 5 round functions and the *Pseudo-Reflector* construction as shown in Figure 5. Note that in order to construct the Super S-box structure, we interchange the orders of the second SubCell operation and the third ShuffleCell operation and the orders of the first Inverse SubCell and the first Inverse ShuffleCell in the *Pseudo-Reflector* construction.

There, and throughout the paper, for internal state nibbles, the colored boxes in a figure represent active nibbles, white boxes represent inactive nibbles; for key nibbles, colored boxed means that the key nibbles are involved in the differential trail.

Property: *If there exists a pair* $(X_2^i, \tilde{X}_2)$ *which satisfies the following two conditions: (a)* $\Delta X_2[2]$ *is equal to* $\Delta X_2[7]$*, and (b) the active nibbles of the pair satisfies the truncated difference characteristic as shown in Figure 5, then after a 5-round and the Pseudo-Reflector encryption, the sequence* $(X_8^0[4], \cdots X_8^{47}[4])$ *will only take about* 2^{116} *values out of* $2^{48\times 4=192}$ *theoretical values.*

Proof: We consider the computation of $X_8^l[4]$ for $0 \leqslant l \leqslant 47$ and $l \neq i$. Actually, $X_8^l[4]$ is determined by the 41-nibble parameters:

$$X_2[2,7]||W_3[7,15]||W_4[0,4,8,3,11,15]||M_3||$$

$$K_5[1,5,9,13,2,6,10,14,3,7,15]||K_6[4,8,12]||K_7[4].$$

Since $X_2[2,7]$ is known and $\Delta X_2[2] = \Delta X_2[7]$, we can get $\Delta Y_3[7,15]$ and use it to deduce $\Delta W_3[7,15]$ as $\Delta Y_3[7] = \Delta Y_3[15]$. Since $W_3[7,15]$ is known, we can get $X_3[7,15]$. Similarly, since ShuffleCell and MixColumn are both linear operations, we can get the value of $\Delta W_4[0,4,8,3,11,15]$. And also as $W_4[0,4,8,3,11,15]$ is known, we can get $X_4[0,12,1,13,10,11]$ and use it to deduce $M_1[0,12,1,13,10,11]$. After another two linear operations, we get ΔM_3. From the known value of M_3, we can deduce the value of X_5 and use it to deduce $Y_5[1,5,9,13,2,6,10,14,3,7,15]$ through the inverse of a ShuffleCell-MixColumn-ShuffleCell opertion. By the knowledge of $K_5[1,5,9,13,2,6,10,14,3,7,15]$, we get $X_6[1,5,9,13,2,6,10,14,3,7,15]$. Then after the inverse of another ShuffleCell-MixColumn-ShuffleCell operation, we get $Y_6[4,8,12]$. Since $K_6[4,8,12]$ is also known, we get the value of $X_7[4,8,12]$ and $Y_7[4]$. Finally, by adding the value of $K_7[4]$, we get $X_8[4]$.

Following we show that when we find a pair of messages conforming to the truncated differential trail outlined in Figure 5, the above 41 nibble-parameters are only determined by the following 31- nibble parameters:

$$K_3[2,7]||Y_3[2,7,8,13]||\Delta Y_5[0,4,12,1,5,9,13,2,6,10,14,3,7,15]||$$

$$\Delta Y_6[4,8,12,1,9]||\Delta Y_7[4]||ruk^1[12,1,13,10,11]$$

Firstly, $X_2[2,7]$ can be directly deduced by $Y_3[2,7]$ and $K_3[2,7]$.

Since the value of $Y_3[2,7,8,13]$ can be used to deduce $Z_3[3,7,11,15]$ and thus $W_3[7,15]$ can be known, the knowledge of $Y_3[2,7,8,13]$ is sufficient to deduce $\Delta Y_4[7,15]$. Using the knowledge of $\Delta Y_4[7,15]$, we also get the value of $\Delta W_4[0,4,8,3,11,15]$.

Combining $\Delta Y_5[0,4,12,1,5,9,13,2,6,10,14,3,7,15]$ and $\Delta Y_6[4,8,12,1,9]$, we can deduce ΔX_5, $X_6[1,5,9,13,2,6,10,14,3,7,15]$, and $Y_6[4,8,12,1,9]$.

Combining $\Delta Y_6[4,8,12]$ and $Y_7[4]$, we get $X_7[4,8,12]$ and $Y_7[4]$.

As $X_7[4,8,12]$ and $Y_6[4,8,12]$ are both known, we can get the value of $K_6[4,8,12]$. According to the key schedule that k^0 is equal to k^1, $ruk^1[0]$ can be deduced by $K_6[4,8,12]$ as K_6 is only dominated by k^0 and some known values.

For the Super S-box, till now we get the value of the input difference $\Delta X_4[0,12,1,13,10,11]$ and the output difference ΔX_5, with the key value $ruk^1[12,1,13,10,11]$, we can get the value of $X_4[0,12,1,13,10,11]$ and X_5 according to its property.

$W_4[0,4,8,3,11,15]$ can be known from $X_4[0,12,1,13,10,11]$ and thus we can deduce the value of $Y_4[7,15]$; from the value of $X_3[7,15]$ and $Y_4[7,15]$, we can get $K_4[7,15]$. M_3 and $Y_5[1,5,9,13,2,6,10,14,3,7,15]$ can be known from X_5.

Since $Y_5[1,5,9,13,2,6,10,14,3,7,15]$ and $X_6[1,5,9,13,2,6,10,14,3,7,15]$ are both known, we get the value of $K_5[1,5,9,13,2,6,10,14,3,7,15]$.

And also, $K_7[4]$ can be deduced by $K_6[4]$ as they are both dominated by k^0, so the 41 nibbles can be absolutely decided by the above 31 nibbles.

Whatsmore, when considering the key-dependent sieve skill, $K_4[7,15]$ can be completely deduced by $K_5[1,5,9,13,2,6,10,14,3,7,15]$ as $w^1 = w^0 \ggg 1 + w^0 \gg 63$.

So actually the sequence has only $2^{31\times4-8} = 2^{116}$ values. □

3.2 The 10-Round with *Pseudo-Refector* Attack on QARMA-64

By applying the distinguisher in Section 3.1, we mount an key recovery attack on 10-round QARMA-64 by adding two rounds on the top and three rounds on the bottom. The extended difference characteristic is shown in Figure 6 and its characteristic probability is 2^{-76}.

The MITM attack process is detailed as follows:

Precomputation Phase:

We need to construct a table containing all possible sequences described in the Property.

1. For each pair $(Y_3[2,7,8,13], Y_3^{'}[2,7,8,13])$ that satisfies $\Delta Y_3[2] = \Delta Y_3[7]$ and $\Delta Y_3[8,13] = 0$, deduce $X_3[7,15]$ and $\Delta Y_4[7,15]$. Store the value of $(Y_3[2,7], X_3[7,15])$ with the index of $\Delta Y_4[7,15]$ in table T_1.
2. Construct table T_2 that lists all values of $(ruk^1[0,12,1,13,10,11], \Delta Y_4[7,15], \Delta Y_5[0,4,12,1,5,9,13,2,6,10,14,3,7,15])$ and the values of the active nibbles in the corresponding pairs $(Y_4, Y_5, Y_4^{'}, Y_5^{'})$ deduced from $(X_4, X_5, X_4^{'}, X_5^{'})$ that satisfies the difference characteristic inside the Super S-box.
3. For all possible values of $\Delta Y_5[1,5,9,13,2,6,10,14,3,7,15]$ and $\Delta Y_6[4,8,12,1,9]$, deduce $(Y_6[4,8,12], X_6[1,5,9,13,2,6,10,14,3,7,15])$. Store the value of $(\Delta Y_6[4,8,12], Y_6[4,8,12], X_6[1,5,9,13,2,6,10,14,3,7,15])$ in table T_3 with the index of $\Delta Y_5[1,5,9,13,2,6,14,3,7,15]$.
4. For all possible values of $\Delta Y_6[4,8,12]$ and $\Delta Y_7[4]$, deduce the value of $Y_7[4]$ and $X_7[4,8,12]$. Store the values of $(\Delta Y_7[4], Y_7[4], X_7[4,8,12])$ in table T_4 with index of $\Delta Y_6[4,8,12]$.
5. For each possible value of difference $\Delta Y_6[4,8,12]$, perform the following operations:
   (a) Query table T_4 to get the corresponding value of $Y_7[4]$ and $X_7[4,8,12]$.
   (b) Query table T_3 to get the value of $(Y_6[4,8,12], X_6[1,5,9,13,2,6,10,14,3,7,15])$ and all possible values of $\Delta Y_5[0,4,12,1,5,9,13,2,6,10,14,3,7,15]$.
   By now we can also deduce value of $K_6[4,8,12]$. According to the key schedule, we also get $K_7[4]$ and $ruk^1[0]$.
   (c) For each possible value of $ruk^1[12,1,13,10,11]$, query table T_2 with $\Delta Y_5[0,4,12,1,5,9,13,2,6,10,14,3,7,15]$ to get $\Delta Y_4[7,15]$. Meanwhile, we also get the value of $Y_5[1,5,9,13,2,6,10,14,3,7,15]$. Combining $Y_5[1,5,9,13,2,6,10,14,3,7,15]$ and $X_6[1,5,9,13,2,6,10,14,3,7,15]$, we deduce the value of $K_5[1,5,9,13,2,6,10,14,3,7,15]$.
   (d) Query table T_1 with the value of $\Delta Y_4[7,15]$ to get $Y_3[2,7]$ and $X_3[7,15]$.
   The value of $K_4[7,15]$ can be deduced from $X_3[7,15]$ and $Y_4[7,15]$.
   (e) Make sure that the deduced and guessed key value do not contradict with the key schedule.
   Guess $K_3[2,7]$, we get $X_2[2,7]$ and thus all needed 116-bit information.
   (f) On the basis of the 116-bit information, we construct a 2-δ-set, compute the corresponding value of sequence $(X_8^0[4], X_8^1[4], \cdots, X_8^{47}[4])$ and store it in a table T.

Till now, we finish the precomputation phase.

Online Phase:

During the online phase, we first look for at least one pair which satisfies the extended truncated differential trail in Figure 6, and then retrieve the master key by guessing some key bits and exhaustively search the other bits.

1. Encrypt 2^{29} structures of 2^{24} plaintexts. The plaintexts of a structure traverses nibbles P[8,12,9,13,2,3] while the other nibbles are constants. For each structure, store the ciphertexts in a table indexed by nibbles C[0,1,2,3], the pairs indexed by the same value are the right pairs.
   After this step, about $2^{47-4\times4} = 2^{31}$ pairs leave in each structure and 2^{60} pairs in total.
2. (a) Guess $K_1[3,9,12]$.
   For all remaining pairs, compute $W_1[3,9,12]$ to deduce $X_1[3,9,12]$. According to the key schedule, the value of $K_2[3,9,12]$ can be easily derived by $K_1[3,9,12]$, so we can compute the value of $Y_2[3,9,12]$. After a ShuffleCell operation, we get the value of $Z_2[6,10,14]$, then use it to compute $\Delta W_2[2]$ as $\Delta Z_2[2] = 0$ and delete the pairs which do not satisfy $\Delta W_2[6,10,14] = 0$.
   Guess $K_1[6]$, as the value of $K_9[6,9,12]$ and $K_{10}[6,9,12]$ can be easily derived from $K_1[6,9,12]$. Partially decrypt the corresponding ciphertexts and get $W_9[6,9,12]$. Delete pairs which do not satisfy $\Delta W_9[2,6,14] = 0$.
   This step performs a 16-bit filter, the expected number of remaining pairs is about $2^{60-16} = 2^{44}$.

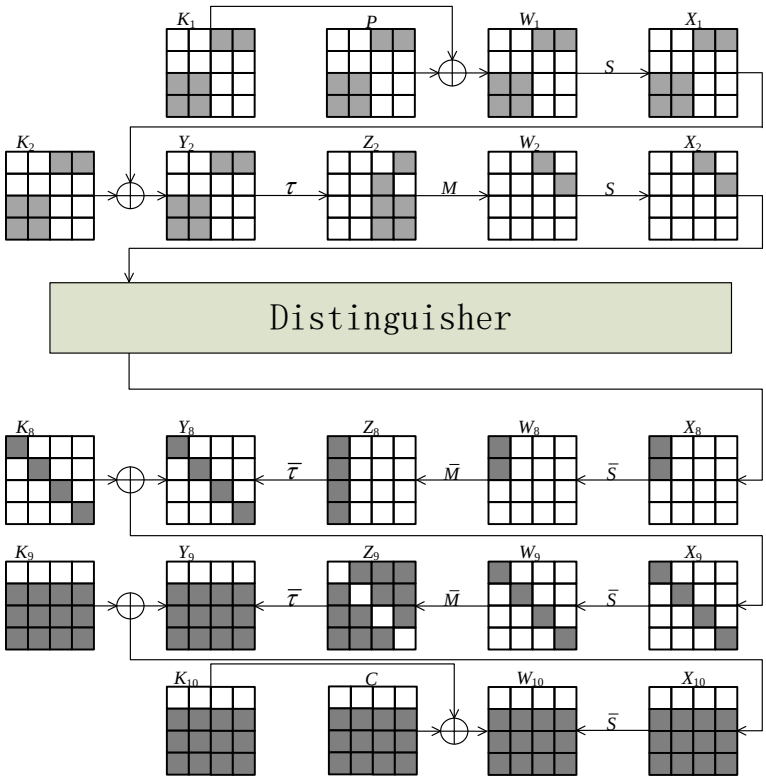

Fig. 6. The Online Phase

(b) Guess $K_1[2, 8, 13]$ and deduce the value of $K_2[2, 8, 13]$. Use the known key information to partially encrypt the remaining pairs to get $W_2[7]$ and remain the pairs that satisfy $\Delta W_2[3, 11, 15] = 0$ and $\Delta W_2[2] = \Delta W_2[7]$.

As same as the operation in step(a), guess $K_1[7]$, we can use $K_1[7, 8, 13]$ to deduce $K_9[7, 8, 13]$ and $K_{10}[7, 8, 13]$. Partially decrypt the remaining pairs to get $W_9[15]$ and delete the pairs that do not satisfy $\Delta W_9[3, 7, 11] = 0$.

After this step, about 2^{24} pairs leave.

(c) Guess $K_{10}[5, 10, 15]$ and deduce $K_9[5, 10, 15]$, and then, we decrypt the remaining pairs to compute $W_9[0]$. Delete pairs that do not satisfy $\Delta W_9[4, 8, 12] = 0$.

Here we get another 8-bit filter, so there are about 2^{16} remaining.

(d) Guess $K_{10}[4, 11, 14]$ and deduce $K_9[4, 11, 14]$, then decrypt the pairs to compute $W_9[5]$ and delete pairs that do not satisfy $\Delta W_9[1, 9, 13] = 0$.

After this step, about 2^8 pairs leave.

(e) Using $K_{10}[5, 10, 15]$ to deduce the value of $K_8[5, 10, 15]$, guess $K_8[0]$ and partially decrypt the pairs and only choose that $\Delta W_8[8, 12] = 0$.

This is a 8-bit filter, so after this, there will be only one pair remaining.

3. Finally we get one pair satisfying the truncated differential trail for every 60-bit key guess. Select one message of the pair and compute the value $X_2[2, 7]$, then deduce the first 48 messages of the corresponding plaintexts for the 2-δ-set:$(X_2^0, X_2^1, \cdots, X_2^{47})$.

Encrypt the plaintexts to get the corresponding ciphertexts and then, partially decrypt the ciphertexts to get the sequence $(X_8^0[4], X_8^1[4], \cdots, X_8^{47}[4])$ and check whether the sequence $(X_8^0[4], X_8^1[4], \cdots, X_8^{47}[4])$ lies in the table T, if not, go back to step 1 for the next subkey guess; Otherwise, we exhaustively search for the other 68-bit key bits and recover the master key.

Complexity analysis: During the precomputation phase, the memory complexity is obviously dominated by table T. It contains 2^{116} 192-bit sequences, so the memory complexity is 2^{116} 192-bit space. The time complexity is about 2^{124} simple computations. In the online phase, we use 2^{29} structures to construct 2^{76} pairs, the data complexity is $2^{29} \times 2^{24} = 2^{53}$ plaintexts. The detail of the time complexity is depicted in Table 2. Here we consider the *Pseudo-Reflector* construction denoted by P as one complete round function. The 10-round with *Pseudo-Reflector* can be seen as a 11-round function in terms of the expense of computation. In total, the time complexity is about $2^{73.6}R \approx 2^{70.1}$ encryptions of a 10-round function with the *Pseudo-Reflector* construction.

Table 2. Time Complexity of Online Phase

Step	Time Complexity
1	$2^{29} \cdot 2^{24}(10R + P) \approx 2^{56.5}R$
2(a)	$3/16 \times 2 \times (2^{60} \times 2^{12} + 2^{52} \times 2^{16})(R + ADK + S) \approx 2^{71.6}R$
2(b)	$3/16 \times 2 \times 2^{16} \times (2^{44} \times 2^{12} + 2^{32} \times 2^{16})(R + ADK + S) \approx 2^{71.6}R$
2(c)	$3/16 \times 2 \times 2^{32} \times 2^{24} \times 2^{12}(R + ADK + S) \approx 2^{65.6}R$
2(d)	$3/16 \times 2 \times 2^{44} \times 2^{16} \times 2^{12}(R + ADK + S) \approx 2^{71.6}$
2(e)	$4/16 \times 2 \times 2^{56} \times 2^{8} \times 2^{4}R \approx 2^{67}R$
3	$2^{60} \times 48 \times (10R + P + 2R + 3R) + 2^{68}(10R + P) \approx 2^{71.5}R$
SUM	$(2^{56.5} + 3 \times 2^{71.6} + 2^{65.6} + 2^{67} + 2^{71.5}) \approx 2^{73.6}R$

3.3 Attack on QARMA-128

In QARMA-128, $M = circ(0, \rho, \rho^2, \rho^5)$(which admits inverse $\bar{M} = \text{circ}(0, \rho^5, \rho^6, \rho)$) with $Q = circ(0, \rho, \rho^4, \rho)$. Although the matrix is different from that in QARMA-64, they share the same transition property as shown in Figure 4.

As a result, we can use a same distinguisher used to cryptanalysis QARMA-64 to attack QARMA-128, and the difference characteristic of the attack is totally the same as in QARMA-64 as both version have the same structure and key schedule.

During the precomputation phase, we need to use 2^{248} simple computations to construct the table T that occupies a space of 2^{232} 384-bit sequence. The data complexity of the online phase is 2^{105} chosen plaintexts while the time complexity is $2^{141.7}$ encryption units.

4 Conclusion

In this paper, we explore the security of QARMA-64 and QARMA-128 against truncated differential attack using the MITM methodology. By using the differential enumeration and key-dependent sieve technique, we propose a MITM distinguisher on QARMA-64. After that, we extend the distinguisher to achieve a 10-round attack on both two versions of QARMA. To the best of our knowledge, this is the first result that analysis the security of QARMA against MITM attack.

Moreover, there are many further works to do: other attacks on QARMA or the way to get better attack complexity with MITM method. We will welcome all possible cooperation to improve this paper.

References

1. Merkle R C. A Fast Software One-way Hash Function. Journal of Cryptology, 1990, Vol.3, No.1, pp:43-58.
2. Rivest R L. The MD_4 Message Digest Algorithm. Advances in Cryptology-CRYPTO'90, 1991:303-311
3. Rivest R L. The MD_5 Message Digest Algorithm. RFC1321, 1992.
4. Secure Hash Standard. National Bureau of Standards FIPS Publication 180, 1993.
5. Yu H B, Wang G L, Zhang G Y, Wang X Y. The Second-Preimage Attack on MD4. CANS, 2005:1-12.
6. Wang X Y, Yu H B, Yin L Y. Efficient Collision Search Attacks on SHA-0. CRYPTO, 2005:1-16.

7. Wang X Y, Yin L Y, Yu H B. Finding Collisions in the Full SHA-1. CRYPTO, 2005:17-36.
8. Wang X Y, Yu H B. How to Break MD5 and Other Hash Functions. EUROCRYPT, 2005:19-35.
9. Liskov M, Rivest R L,Wagner D. Tweakable Block Ciphers. CRYPTO, 2002:31-46.
10. Avanzi R. The QARMA Block Cipher Family. IACR Cryptology ePrint Archive, 2016:444.
11. https://community.arm.com/groups/processors/blog/2016/10/27/armv8-a-architecture-2016-additions
12. Diffie W, Hellman E M. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. IEEE Computer,1977,10(6):74-84.
13. Demirci H, Selçuk A A. A Meet-in-the-Middle Attack on 8-Round AES. FSE, 2008:116-126.
14. Daemen J, Rijmen V. The Design of Rijndael: AES-The Advanced Encryption Standard. ISBN:978-3-642-07646-6, Germany, Springer Berlin Heidelberg, 2002.
15. Li Leibo, Jia Keting, Wang Xiaoyun. Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE. IACR Cryptology ePrint Archive, 2013:573.
16. Dunkelman O, Keller N, Shamir A. Improved Single-key Attacks on 8-Round AES-192 and AES-256. ASIACRYPT, 2010:158-176.
17. Li Leibo, Jia Keting, Wang Xiaoyun. Improved Single-Key Attacks on 9-Round AES-192/256. FSE, 2014:127-146.
18. Guo J, Jean J, Nikolić I, Sasaki Y. Meet-in-the-Middle Attacks on Generic Feistel Constructions. ASIACRYPT, 2014:458-477.
19. Daemen J, Rijmen V. Understanding Two-Round Differentials in AES. In *Security and Cryptography for Networks*,Germany,Springer Berlin Heidelberg,2006,pp:78-94.
20. Derbez P, Fouque P A, Jean J. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. EUROCRYPT,2013:371-387.
21. Banik S, Bogdanov A, Isobe T, Shibutani K, Hiwatari H, Akishita T, Regazzoni F. Midori: A Block Cipher for Low Energy. ASIACRYPT, 2015, Vol.9453, pp:411C436.
22. FIPS 46-3. Data Encryption Standard. In National Institute of Standards and Technology, 1977.