Security Analysis of Michael: the IEEE 802.11i Message Integrity Code

Abstract. The IEEE 802.11b standard employs a data security mechanism known as Wired Equivalent Privacy (WEP). WEP uses RC4 stream cipher for its data encryption and CRC-32 to check its message integrity. Recent research shows that WEP is not secure as it does not use RC4 and CRC-32 correctly. The latest IEEE 802.11i draft uses a new keyed hash function, called *Michael*, as the message integrity code. This paper describes some properties and weaknesses of Michael. We provide a necessary and sufficient condition for finding collisions of Michael. Our observation reveals that the collision status of Michael only depends on the second last block message and the output of the block function in the third last round. We show that Michael is not collision-free by providing a method to find collisions of this keyed hash function. Moreover, we develop a simple method to find fixed points of Michael, and our results demonstrate that the percentage of the existence of the fixed points is extremely high based on our randomly chosen samples. If the output of the block function in any round is equal to any of these fixed points, a packet forgery attack could be mounted against Michael.

1 Introduction

Wireless networks and mobile devices provide ubiquitous computing environments to users. Based on specific transmissions mediums, wireless networks play an important role in cyber world. Along with its popularity, wireless connectivity brings new problems as security issues need to be considered.

Wireless devices based on IEEE 802.11b standard [3] are widely in use nowadays. The IEEE 802.11b defines an encryption scheme called Wired Equivalent Privacy (WEP). It is well known that WEP has several serious security flaws. Fluhrer, Mantin, and Shamir [7] (FMS) proposed an attack on the WEP encryption protocol. By exploiting weaknesses of the RC4 [9] key scheduling algorithm, the FMS attack demonstrated that the RC4 encryption key can be easily derived by an eavesdropper who can intercept several million encrypted WEP packets whose first byte of plaintext is known. Stubblefield, Ioannidis, and Rubin [10] practically implemented the FMS attack, and showed that the real systems could be defeated. Borisov, Goldberg, and Wagner [5] showed that the WEP data integrity could be compromised as encrypted messages could be modified freely by

an attacker without being detected. Moreover, Arbaugh, Shankar, and Wan [4] showed that the WEP authentication mechanism is vulnerable to attack.

To address the WEP vulnerabilities, the IEEE 802.11 Task Group i (TGi) provides a short-term solution and a long-term solution. The short-term solution has adopted the Temporal Key Integrity Protocol (TKIP). TKIP is a group of algorithms that wraps the WEP protocol to address the known weaknesses. TKIP includes three components: a message integrity code called *Michael*, a packet sequencing discipline, and a per-packet key mixing function. Figure 1 illustrates the TKIP encryption procedure. TKIP is considered as a temporary solution, and it is designed for legacy hardware. For the long-term solution, the IEEE 802.11 TGi recommends two modes of operation: WRAP (Wireless Robust Authenticated Protocol) and CCMP (Counter-Mode-CBC-MAC Protocol). Both WARP and CCMP are based on AES cipher [2], and they require new hardware.

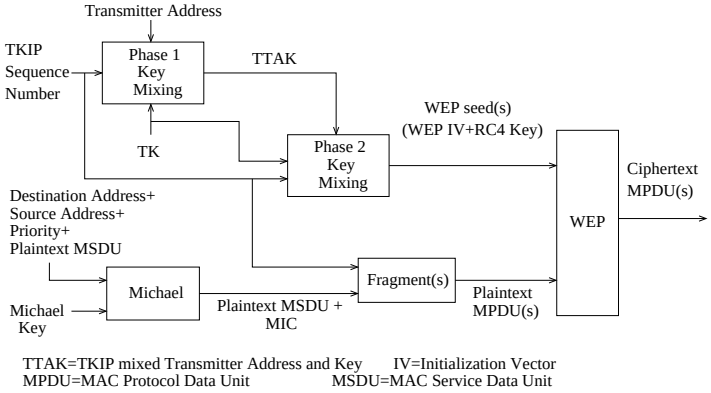

Fig. 1. *TKIP Encryption Diagram*

Our Contributions

In this paper, we investigate the security issues of Michael. First, we present a necessary and sufficient condition for finding collisions of Michael, showing that the collision status of Michael *only* depends on the second last block message and the output of the block function in the third last round. Second, by employing the necessary and sufficient condition, we provide a method to find collisions of Michael and show that Michael is not collision-free. Furthermore, we develop a method to find fixed points of Michael, and a packet forgery attack could be mounted against Michael if the output of the block function in any round is equal to any of these fixed points.

Notations

A 64-bit Michael key K is converted to two 32-bit subkeys, k_0 and k_1, written as $K = (k_0, k_1)$. An n-block message M is written as $M = (m_0,$

$m_1, ..., m_{n-1}$). L_0^i, R_0^i, L_1^i, R_1^i, L_2^i, R_2^i, L_3^i, R_3^i, L_4^i, R_4^i, and L_5^i are variables used in the $(i+1)$-th round of Michael(K, M) procedure. For an n-round Michael(K, M) procedure, we represent the $(i+1)$-th $(0 \leq i \leq n-1)$ round output of the Michael block function as (L_5^i, R_4^i), where L_5^i stands for the left half of the output and R_4^i stands for the right half of the output. Some other notations used in this paper are listed as follows:

Symbol	Description
MAC	Medium access control
$<<<$	Left rotation
$>>>$	Right rotation
$\oplus$	Exclusive-or
$\boxplus$	addition modulo 2^{32}
$\|\|$	Concatenation
$\Longrightarrow$	imply

Organization

The rest of this paper is organized as follows. Section 2 provides the overview of the Michael keyed hash function. Section 3 describes one previous work on Michael, which shows that Michael is invertible. We provide a necessary and sufficient condition for finding collisions of Michael in Section 4. In Section 5, we propose a method to find collisions of Michael, and based on our method, we show that Michael is not collision-free. In Section 6, we introduce a simple method to find fixed points of Michael and propose a packet forgery attack against Michael. Finally, we conclude this paper in Section 7.

2 The Michael Keyed Hash Function

Michael is the message integrity code (MIC) of TKIP in the IEEE 802.11i draft [1], and it was designed by Ferguson [6]. Michael is a keyed hash function, whose inputs are a 64-bit Michael key and an arbitrarily long message, and output is a 64-bit Michael value. The 64-bit key is converted to two key 32-bit words, and the message is partitioned into 32-bit blocks. The message is padded at the end with a single byte with the hexadecimal value *0x5a* and then followed by between 4 and 7 zero bytes. The number of zero bytes is chosen so that the overall length of the message plus the padding is a multiple of 4. The padding method is illustrated in Figure 2. We note that the last block of the padded message is zero, and the second last block of the padded message is

not zero. The details of Michael are described in Algorithm 2.1 and 2.2.

Algorithm 2.1: MICHAEL$((k_0, k_1), (m_0, ..., m_{n-1}))$

Input : **Key**(k_0, k_1)
Input : **Padded message** $(m_0, ..., m_{n-1})$
Output : **MIC value** (L, R)
$(L, R) \leftarrow (k_0, k_1)$
for $i \leftarrow 0$ **to** $n - 1$
do $\begin{cases} L \leftarrow L \oplus m_i \\ (L, R) \leftarrow B(L, R)(Algorithm 2.2) \end{cases}$
return (L, R)

Algorithm 2.2: B(L, R)

Input : (L, R)
Output : (L, R)
$R \leftarrow R \oplus (L <<< 17)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus XSWAP(L)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus (L <<< 3)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus (L >>> 2)$
$L \leftarrow (L + R) \bmod 2^{32}$
return (L, R)

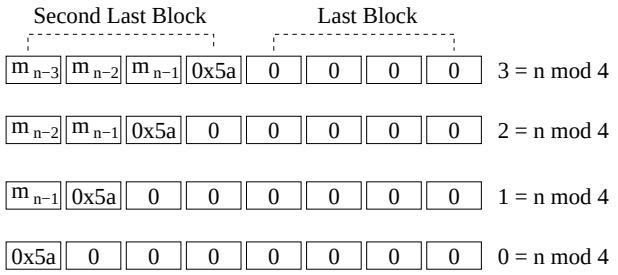

Fig. 2. *The Padding Method of Michael*

Michael employs several operations, including exclusive-or, left rotation, right rotation, addition modulo 2^{32} and swapping ($XSWAP$). Function $XSWAP$ swaps the position of the two least significant bytes and the position of the two most significant bytes in a word, i.e., $XSWAP(ABCD) = BADC$ where A, B, C, D are bytes. The block function given in Algorithm 2.2 is an unkeyed 4-round Feistel-type construction.

The TKIP frame appends the MIC value as a tag after the message body. The message body together with the MIC value are encrypted by RC4 at the transmitter and then sent to the receiver. The receiver recomputes the MIC value and compares the computed result with the tag coming with the message. If these two MIC values match, the receiver accepts the message; if not, the receiver rejects the message.

3 Related Work

3.1 Michael is Not One-Way

Wool found one weakness of Michael: it is *not* one-way, in fact, it is *invertible* [11]. There exists a simple function, InvMichael, which can recover the secret Michael key K, given a known message M and its corresponding Michael value MIC = Michael(K, M). The details of Function InvMichael are shown in Appendix A. We note that the block function is unkeyed, and every step in Michael is invertible, therefore the whole Michael algorithm is invertible.

3.2 A Related-Message Attack

The security of Michael relies on the fact that a message and its hash are encrypted by RC4, and thus the hash value is unknown to the attacker. Wool proposed a related-message attack on Michael [11].

Remark: Michael is invertible is known by the inventor of Michael, and this security flaw is mentioned implicitly on Page 14 in [6]:

"A known-plaintext attack will reveal the key stream for that IV, and if the second packet encrypted with the same IV is shorter than the first one, the MIC value is revealed, which can then be used to derive the authentication key."

4 Finding Collisions of Michael

We explore the collision-resistance of Michael in this section. By providing Theorem 1, we prove that the collision status of Michael only depends on the second last block message and the output of the block function in the third last round. We would like to point out that Condition 1 and 2 in Theorem 1 are a necessary and sufficient condition for finding collisions of Michael.

Theorem 1. *Given two pairs of keys and messages, (Key_1, M_1) and (Key_2, M_2), Michael(Key_1, M_1) = Michael(Key_2, M_2) if and only if the following two conditions hold:*

1. $R_4^{x-3} = R_4'^{y-3}$
2. $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$

where M_1 has x 32-bit blocks, M_2 has y 32-bit blocks, and both x and y are ≥ 3.

Proof. The last three rounds of Michael are illustrated in Figure 3. We

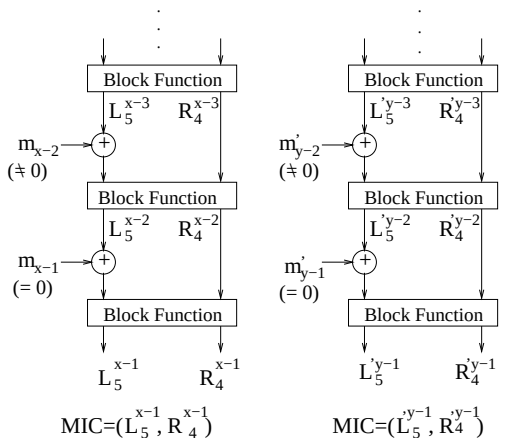

Fig. 3. *Last Three Rounds of Michael*

provide the last round and the second last round of Michael($Key_1,\ M_1$) in Algorithm 4.1 and Algorithm 4.2 respectively.

Algorithm 4.1: LAST ROUND (Key_1, M_1)

$1.L_0^{x-1} = L_5^{x-2}$
$2.R_0^{x-1} = R_4^{x-2}$
$3.L_1^{x-1} = L_0^{x-1} \oplus m_{x-1}$
$4.R_1^{x-1} = R_0^{x-1} \oplus (L_1^{x-1} <<< 17)$
$5.L_2^{x-1} = (L_1^{x-1} + R_1^{x-1}) \bmod 2^{32}$
$6.R_2^{x-1} = R_1^{x-1} \oplus XSWAP(L_2^{x-1})$
$7.L_3^{x-1} = (L_2^{x-1} + R_2^{x-1}) \bmod 2^{32}$
$8.R_3^{x-1} = R_2^{x-1} \oplus (L_3^{x-1} <<< 3)$
$9.L_4^{x-1} = (L_3^{x-1} + R_3^{x-1}) \bmod 2^{32}$
$10.R_4^{x-1} = R_3^{x-1} \oplus (L_4^{x-1} >>> 2)$
$11.L_5^{x-1} = (L_4^{x-1} + R_4^{x-1}) \bmod 2^{32}$
$(Note: Michael(Key_1, M_1) = (L_5^{x-1}, R_4^{x-1}))$

Algorithm 4.2: 2RD LAST (Key_1, M_1)

$1.L_0^{x-2} = L_5^{x-3}$
$2.R_0^{x-2} = R_4^{x-3}$
$3.L_1^{x-2} = L_0^{x-2} \oplus m_{x-2}$
$4.R_1^{x-2} = R_0^{x-2} \oplus (L_1^{x-2} <<< 17)$
$5.L_2^{x-2} = (L_1^{x-2} + R_1^{x-2}) \bmod 2^{32}$
$6.R_2^{x-2} = R_1^{x-2} \oplus XSWAP(L_2^{x-2})$
$7.L_3^{x-2} = (L_2^{x-2} + R_2^{x-2}) \bmod 2^{32}$
$8.R_3^{x-2} = R_2^{x-2} \oplus (L_3^{x-2} <<< 3)$
$9.L_4^{x-2} = (L_3^{x-2} + R_3^{x-2}) \bmod 2^{32}$
$10.R_4^{x-2} = R_3^{x-2} \oplus (L_4^{x-2} >>> 2)$
$11.L_5^{x-2} = (L_4^{x-2} + R_4^{x-2}) \bmod 2^{32}$

Similarly, the last round and the second last round of Michael(Key_2, M_2) are shown in Algorithm 4.3 and Algorithm 4.4 respectively.

Algorithm 4.3: LAST ROUND (Key_2, M_2)

$1.L_0'^{y-1} = L_5'^{y-2}$
$2.R_0'^{y-1} = R_4'^{y-2}$
$3.L_1'^{y-1} = L_0'^{y-1} \oplus m'_{y-1}$
$4.R_1'^{y-1} = R_0'^{y-1} \oplus (L_1'^{y-1} <<< 17)$
$5.L_2'^{y-1} = (L_1'^{y-1} + R_1'^{y-1}) \bmod 2^{32}$
$6.R_2'^{y-1} = R_1'^{y-1} \oplus XSWAP(L_2'^{y-1})$
$7.L_3'^{y-1} = (L_2'^{y-1} + R_2'^{y-1}) \bmod 2^{32}$
$8.R_3'^{y-1} = R_2'^{y-1} \oplus (L_3'^{y-1} <<< 3)$
$9.L_4'^{y-1} = (L_3'^{y-1} + R_3'^{y-1}) \bmod 2^{32}$
$10.R_4'^{y-1} = R_3'^{y-1} \oplus (L_4'^{y-1} >>> 2)$
$11.L_5'^{y-1} = (L_4'^{y-1} + R_4'^{y-1}) \bmod 2^{32}$
$(Note: Michael(Key_2, M_2) = (L_5'^{y-1}, R_4'^{y-1})$

Algorithm 4.4: 2RD LAST (Key_2, M_2)

$1.L_0'^{y-2} = L_5'^{y-3}$
$2.R_0'^{y-2} = R_4'^{y-3}$
$3.L_1'^{y-2} = L_0'^{y-2} \oplus m'_{y-2}$
$4.R_1'^{y-2} = R_0'^{y-2} \oplus (L_1'^{y-2} <<< 17)$
$5.L_2'^{y-2} = (L_1'^{y-2} + R_1'^{y-2}) \bmod 2^{32}$
$6.R_2'^{y-2} = R_1'^{y-2} \oplus XSWAP(L_2'^{y-2})$
$7.L_3'^{y-2} = (L_2'^{y-2} + R_2'^{y-2}) \bmod 2^{32}$
$8.R_3'^{y-2} = R_2'^{y-2} \oplus (L_3'^{y-2} <<< 3)$
$9.L_4'^{y-2} = (L_3'^{y-2} + R_3'^{y-2}) \bmod 2^{32}$
$10.R_4'^{y-2} = R_3'^{y-2} \oplus (L_4'^{y-2} >>> 2)$
$11.L_5'^{y-2} = (L_4'^{y-2} + R_4'^{y-2}) \bmod 2^{32}$

Necessary Condition: If Michael($Key_1,\ M_1$) = Michael($Key_2,\ M_2$), namely the collision occurs, we then backtrack from Step 11 and 10 in Algorithm 4.1 and 4.3.

$$
\begin{aligned}
&L_5^{x-1} = L_5'^{y-1} \text{ and } R_4^{x-1} = R_4'^{y-1} \Longrightarrow L_4^{x-1} = L_4'^{y-1},\\
&L_4^{x-1} = L_4'^{y-1} \text{ and } R_4^{x-1} = R_4'^{y-1} \Longrightarrow R_3^{x-1} = R_3'^{y-1},\\
&L_4^{x-1} = L_4'^{y-1} \text{ and } R_3^{x-1} = R_3'^{y-1} \Longrightarrow L_3^{x-1} = L_3'^{y-1},\\
&L_3^{x-1} = L_3'^{y-1} \text{ and } R_3^{x-1} = R_3'^{y-1} \Longrightarrow R_2^{x-1} = R_2'^{y-1},\\
&L_3^{x-1} = L_3'^{y-1} \text{ and } R_2^{x-1} = R_2'^{y-1} \Longrightarrow L_2^{x-1} = L_2'^{y-1},\\
&L_2^{x-1} = L_2'^{y-1} \text{ and } R_2^{x-1} = R_2'^{y-1} \Longrightarrow R_1^{x-1} = R_1'^{y-1},\\
&L_2^{x-1} = L_2'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow L_1^{x-1} = L_1'^{y-1},\\
&L_1^{x-1} = L_1'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow R_0^{x-1} = R_0'^{y-1}.
\end{aligned}
$$

As $L_1^{x-1} = L_0^{x-1} \oplus m_{x-1}$, $L_1'^{y-1} = L_0'^{y-1} \oplus m'_{y-1}$, $L_0^{x-1} = L_5^{x-2}$, $L_0'^{y-1} = L_5'^{y-2}$, $R_0^{x-1} = R_4^{x-2}$, $R_0'^{y-1} = R_4'^{y-2}$, $m_{x-1} = 0$ and $m'_{y-1} = 0$, therefore $L_5^{x-2} = L_5'^{y-2}$ and $R_4^{x-2} = R_4'^{y-2}$.

Similarly, we use the same method in the second last rounds of Michael(Key_1, M_1) and Michael(Key_2, M_2).

$$
\begin{gathered}
L_5^{x-2} = L_5'^{y-2} \text{ and } R_4^{x-2} = R_4'^{y-2} \Longrightarrow L_4^{x-2} = L_4'^{y-2}, \\
L_4^{x-2} = L_4'^{y-2} \text{ and } R_4^{x-2} = R_4'^{y-2} \Longrightarrow R_3^{x-2} = R_3'^{y-2}, \\
L_4^{x-2} = L_4'^{y-2} \text{ and } R_3^{x-2} = R_3'^{y-2} \Longrightarrow L_3^{x-2} = L_3'^{y-2}, \\
L_3^{x-2} = L_3'^{y-2} \text{ and } R_3^{x-2} = R_3'^{y-2} \Longrightarrow R_2^{x-2} = R_2'^{y-2}, \\
L_3^{x-2} = L_3'^{y-2} \text{ and } R_2^{x-2} = R_2'^{y-2} \Longrightarrow L_2^{x-2} = L_2'^{y-2}, \\
L_2^{x-2} = L_2'^{y-2} \text{ and } R_2^{x-2} = R_2'^{y-2} \Longrightarrow R_1^{x-2} = R_1'^{y-2}, \\
L_2^{x-2} = L_2'^{y-2} \text{ and } R_1^{x-2} = R_1'^{y-2} \Longrightarrow L_1^{x-2} = L_1'^{y-2}, \\
L_1^{x-2} = L_1'^{y-2} \text{ and } R_1^{x-2} = R_1'^{y-2} \Longrightarrow R_0^{x-2} = R_0'^{y-2}.
\end{gathered}
$$

As $L_1^{x-2} = L_0^{x-2} \oplus m_{x-2}$, $L_1'^{y-2} = L_0'^{y-2} \oplus m'_{y-2}$, $L_0^{x-2} = L_5^{x-3}$ and $L_0'^{y-2} = L_5'^{y-3}$, therefore $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$. As $R_0^{x-2} = R_4^{x-3}$ and $R_0'^{y-2} = R_4'^{y-3}$, therefore $R_4^{x-3} = R_4'^{y-3}$.

Thus, Michael(Key_1, M_1) = Michael(Key_2,M_2) $\Longrightarrow R_4^{x-3} = R_4'^{y-3}$ and $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$.

Sufficient Condition: If $R_4^{x-3} = R_4'^{y-3}$ and $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$ hold, we start from Step 1 and 2 in Algorithm 4.2 and 4.4.

$$
\begin{gathered}
L_5^{x-3} = L_0^{x-2},\ L_5'^{y-3} = L_0'^{y-2} \text{ and } L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2} \Longrightarrow \\
L_1^{x-2} = L_1'^{y-2}, \\
R_4^{x-3} = R_4'^{y-3},\ R_4^{x-3} = R_0^{x-2} \text{ and } R_4'^{y-3} = R_0'^{y-2} \Longrightarrow R_0^{x-2} = R_0'^{y-2}, \\
L_1^{x-2} = L_1'^{y-2} \text{ and } R_0^{x-2} = R_0'^{y-2} \Longrightarrow R_1^{x-2} = R_1'^{y-2}, \\
L_1^{x-2} = L_1'^{y-2} \text{ and } R_1^{x-2} = R_1'^{y-2} \Longrightarrow L_2^{x-2} = L_2'^{y-2}, \\
L_2^{x-2} = L_2'^{y-2} \text{ and } R_1^{x-2} = R_1'^{y-2} \Longrightarrow R_2^{x-2} = R_2'^{x-2}, \\
L_2^{x-2} = L_2'^{y-2} \text{ and } R_2^{x-2} = R_2'^{x-2} \Longrightarrow L_3^{x-2} = L_3'^{y-2}, \\
L_3^{x-2} = L_3'^{y-2} \text{ and } R_2^{x-2} = R_2'^{x-2} \Longrightarrow R_3^{x-2} = R_3'^{y-2}, \\
L_3^{x-2} = L_3'^{y-2} \text{ and } R_3^{x-2} = R_3'^{y-2} \Longrightarrow L_4^{x-2} = L_4'^{y-2}, \\
L_4^{x-2} = L_4'^{y-2} \text{ and } R_3^{x-2} = R_3'^{y-2} \Longrightarrow R_4^{x-2} = R_4'^{y-2}, \\
L_4^{x-2} = L_4'^{y-2} \text{ and } R_4^{x-2} = R_4'^{y-2} \Longrightarrow L_5^{x-2} = L_5'^{y-2}.
\end{gathered}
$$

Finally, we bring the above results from the second last rounds to the last rounds. According to the padding method, we note that $m_{x-1} = 0$ and $m'_{y-1} = 0$.

$$
\begin{gathered}
L_5^{x-2} = L_5'^{y-2},\ L_0^{x-1} = L_5^{x-2} \text{ and } L_0'^{y-1} = L_5'^{y-2} \Longrightarrow L_0^{x-1} = L_0'^{y-1}, \\
R_4^{x-2} = R_4'^{y-2},\ R_4^{x-2} = R_0^{x-1} \text{ and } R_4'^{y-2} = R_0'^{y-1} \Longrightarrow R_0^{x-1} = R_0'^{y-1}, \\
L_0^{x-1} = L_0'^{y-1} \text{ and } m_{x-1} = m'_{y-1} \Longrightarrow L_1^{x-1} = L_1'^{y-1}, \\
L_1^{x-1} = L_1'^{y-1} \text{ and } R_0^{x-1} = R_0'^{y-1} \Longrightarrow R_1^{x-1} = R_1'^{y-1}, \\
L_1^{x-1} = L_1'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow L_2^{x-1} = L_2'^{y-1}, \\
L_2^{x-1} = L_2'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow R_2^{x-1} = R_2'^{y-1}, \\
L_2^{x-1} = L_2'^{y-1} \text{ and } R_2^{x-1} = R_2'^{y-1} \Longrightarrow L_3^{x-1} = L_3'^{y-1}, \\
L_3^{x-1} = L_3'^{y-1} \text{ and } R_2^{x-1} = R_2'^{y-1} \Longrightarrow R_3^{x-1} = R_3'^{y-1}, \\
L_3^{x-1} = L_3'^{y-1} \text{ and } R_3^{x-1} = R_3'^{y-1} \Longrightarrow L_4^{x-1} = L_4'^{y-1}, \\
L_4^{x-1} = L_4'^{y-1} \text{ and } R_3^{x-1} = R_3'^{y-1} \Longrightarrow R_4^{x-1} = R_4'^{y-1}, \\
L_4^{x-1} = L_4'^{y-1} \text{ and } R_4^{x-1} = R_4'^{y-1} \Longrightarrow L_5^{x-1} = L_5'^{y-1}.
\end{gathered}
$$

Therefore, $R_4^{x-3} = R_4'^{y-3}$ and $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2} \Longrightarrow$ Michael(Key_1, M_1) = Michael(Key_2, M_2).

In conclusion, $R_4^{x-3} = R_4'^{y-3}$ and $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$ are a necessary and sufficient condition of Michael(Key_1, M_1) = Michael(Key_2, M_2).

5 Michael is Not Collision-Free

In this section, we show that Michael is not collision-free by providing a simple method to find collisions of Michael. Intuitively, for a given arbitrarily length message M and a key K, a 96-bit block message M' and a key K' can be computed such that Michael(K, M) = Michael(K', M').

Theorem 2. *Given an arbitrarily length message M and a specific key K, a 96-bit block message M' distinct from M and a key K' can always be computed such that Michael(K, M) = Michael(K', M'), where M has n 32-bit blocks and n is any integer ≥ 3.*

Proof. We write M as $(m_0, m_1, ..., m_{n-1})$, and M' as (m'_0, m'_1, m'_2). We represent the outputs of the last, second last, third last and fourth last round of Michael(K, M) as (L_5^{n-1}, R_4^{n-1}), (L_5^{n-2}, R_4^{n-2}), (L_5^{n-3}, R_4^{n-3}) and (L_5^{n-4}, R_4^{n-4}) respectively. The outputs of the last, second last and third last round of Michael(K', M') are represented as $(L_5'^2, R_4'^2)$, $(L_5'^1, R_4'^1)$ and $(L_5'^0, R_4'^0)$ respectively. K' is written as (k'_0, k'_1). K', m'_0, m'_1 and m'_2 are constructed as follows.

1. Choose $m'_2 = 0$ (as $m_{n-1} = 0$ according to the padding method).
2. Choose $m'_1 = m_{n-2}$.
3. Choose m'_0 arbitrarily, but $m'_0 \neq m_{n-3}$ if $n = 3$.
4. Choose $k'_0 = L_5^{n-4} \oplus m_{n-3} \oplus m'_0$ and $k'_1 = R_4^{n-4}$. K' is constructed as $K' = (k'_0, k'_1) = (L_5^{n-4} \oplus m_{n-3} \oplus m'_0, R_4^{n-4})$.

The construction is illustrated in Figure 4. The soundness of this construction is shown as follows.

$$k'_0 = L_5^{n-4} \oplus m_{n-3} \oplus m'_0 \Longrightarrow k'_0 \oplus m'_0 = L_5^{n-4} \oplus m_{n-3},$$
$$k'_0 \oplus m'_0 = L_5^{n-4} \oplus m_{n-3} \text{ and } k'_1 = R_4^{n-4} \Longrightarrow R_4^{n-3} = R_4'^0 \text{ and } L_5^{n-3} = L_5'^0,$$
$$L_5^{n-3} = L_5'^0 \text{ and } m_{n-2} = m'_1 \Longrightarrow L_5^{n-3} \oplus L_5'^0 = m_{n-2} \oplus m'_1.$$

Therefore, Michael(K, M) = Michael(K', M') holds because $R_4^{n-3} = R_4'^0$ satisfies Condition 1 in Theorem 1 and $L_5^{n-3} \oplus L_5'^0 = m_{n-2} \oplus m'_1$ satisfies Condition 2 in Theorem 1.

Theorem 3. *Michael is not collision-free.*

Proof. Can be deduced from Theorem 2.

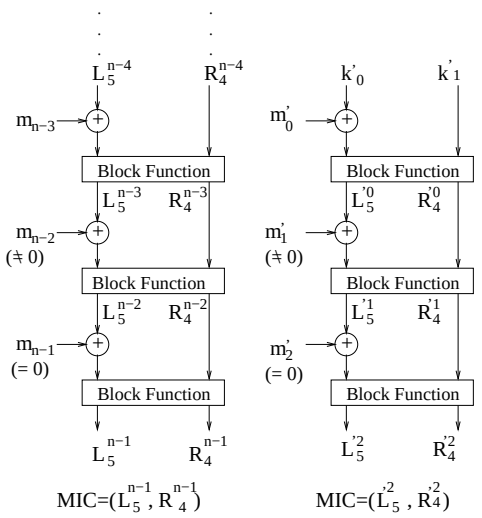

Fig. 4. *The Construction of (K′, M′)*

6 Finding Fixed Points of Michael

In this section, we present a method to find fixed points of Michael. A fixed point of Michael is a triple $(L_i,\ R_i,\ m_i)$ such that Michael$((L_i,\ R_i),\ m_i) = (L_i,\ R_i)$. The procedure is described in Section 6.1. A packet forgery attack could be mounted against Michael if the output of the Michael block function is equal to any of the fixed points. The packet forgery attack is shown in Section 6.2.

6.1 The Fixed-Point Finding Procedure

To find fixed points of Michael, we only need to focus on one round of Michael. Figure 5 illustrates one round of Michael. In Figure 5, we note

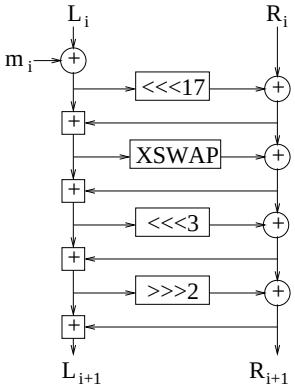

Fig. 5. *One Round of Michael*

that Michael$((L_i,\ R_i),\ m_i) = (L_{i+1},\ R_{i+1})$. In the finding procedure, our goal is to find a triple $(L_i,\ R_i,\ m_i)$ such that Michael$((L_i,\ R_i),\ m_i) =$

$(L_{i+1}, R_{i+1}) = (L_i, R_i)$. The procedure is described as follows.

1. Let $X_i = L_i \oplus m_i$, and choose a value for R_i. Define a counter c and set it to zero.
2. FOR ($X_i = 0$; $X_i \leq 2^{32}$; X_i++)
   (a) Call block function B(X_i, R_i)
   (b) IF $R_i = R_{i+1}$ THEN
      i. There exists an X_i such that $R_i = R_{i+1}$. For a found X_i, there exists a corresponding L_{i+1} because the mapping from (X_i, R_i) to (L_{i+1}, R_{i+1}) is bijective. The reason why the mapping from (X_i, R_i) to (L_{i+1}, R_{i+1}) is bijective is that Michael is invertible. Choose $L_i = L_{i+1}$.
      ii. Choose $m_i = X_i \oplus L_i$.
      iii. Increase counter c by one.
3. IF counter $c = 0$ THEN no fixed point found for this R_i.
4. ELSE There are c fixed points for this R_i.

The key point of this procedure is in Step 2 (b). Given an X_i, if $R_i = R_{i+1}$ holds, there exists a fixed point (m_i, L_i, R_i) such that Michael((L_i, R_i), m_i) = (L_i, R_i). For a specific value of R_i, the time complexity of deciding whether there exists a fixed point of Michael is $O(2^{32})$. To search the complete space of R_i for all fixed points, the time complexity is $O(2^{64})$ since R_i is 32-bit.

We have implemented the fixed-point finding procedure on a personal computer whose processor is an Intel Pentium 4 2.8 GHz, and the program only takes 2-3 minutes to decide whether there exists a fixed point for a given R_i. We provide the first 32 fixed points found by using our method in Table 1 (Numbers are hexadecimal and listed in increasing order according to R_i). A more complete table is provided in the Appendix D.

X_i	m_i	L_i	R_i	X_i	m_i	L_i	R_i
0	0	0	0	b207d8fd	ac29ffed	1e2e2710	8
6c06529a	f886b395	9480e10f	0	6e66938	d44d5dd2	d2ab34ea	9
4e91dea2	7161872	4987c6d0	1	8381416d	5fcc4b0d	dc4d0a60	9
54efbc34	69bd6b8e	3d52d7ba	1	f209915c	ba9f2472	4896b52e	9
84c99b9d	bbac8b1a	3f651087	2	8fbdb557	8ebe3dff	10388a8	c
5c8fc604	a02eb006	fca17602	3	9989f930	44951984	dd1ce0b4	c
5c9f83fa	16443902	4adbbaf8	3	35848ac	4bfe1c3b	48a65497	d
a93ee58c	1c398e95	b5076b19	3	5557549c	649f92b1	31c8c62d	d
b5db2ba7	d4a38f0	b8911357	3	7c0332e2	153f6792	693c5570	d
a8d11268	c778177c	6fa90514	4	39183e91	9ea7035d	a7bf3dcc	e
5781003c	960dcfde	c18ccfe2	5	6eecb6a1	bd5114c5	d3bda264	e
6ac32ecf	60884be2	a4b652d	5	a9a9eedf	d7e79f0	a4d7972f	e
b2dc2a5d	d6ac3e02	6470145f	5	bc4f846a	2abd84ca	96f200a0	e
fc231200	d07d4eb9	2c5e5cb9	5	fadcef66	9d7687c1	67aa68a7	f
c5cddd7a	673b6fc6	a2f6b2bc	6	afb1715f	74cca5e1	db7dd4be	10
e5b473b1	83ea90fc	665ee34d	7	d8689e66	f2d6168a	2abe88ec	10

Table 1. First 32 Fixed Points of Michael

For a given R_i, there are three possible cases for the corresponding X_i:

1. X_i does not exist. Accordingly, there does not exist any fixed point for this R_i (e.g., $R_i = a$ not in Table 1).
2. X_i has only one value. Accordingly, there exists only one fixed point for this R_i (e.g., $R_i = 4$ in Table 1).
3. X_i has more than one values. Accordingly, there exist more than one fixed points for this R_i (e.g., $R_i = e$ in Table 1).

Remark: A complete search of the very beginning ($[0, 2^{10}]$) and very end ($[2^{32}\text{-}2^{10}, 2^{32}]$) of the possible space for R_i shows that 96.48% (see Appendix B) of the calculations yielded fixed points. In addition, unless the distribution of fixed points in $[2^{10}, 2^{32}\text{-}2^{10}]$ is grossly nonuniform, there is a 95% likelihood that the percentage of fixed points there lies between 95.02% and 97.36% (see Appendix C). In other words, there exist about $2^{32} * p$ ($95.02\% < p < 97.36\%$) fixed points for Michael. We note that the tables in the Appendix D can be precomputed and any further fixed points discovered later can be included in the tables. We consider the potential high percentage of the existence of the fixed points within the whole space of R_i as a weakness of Michael.

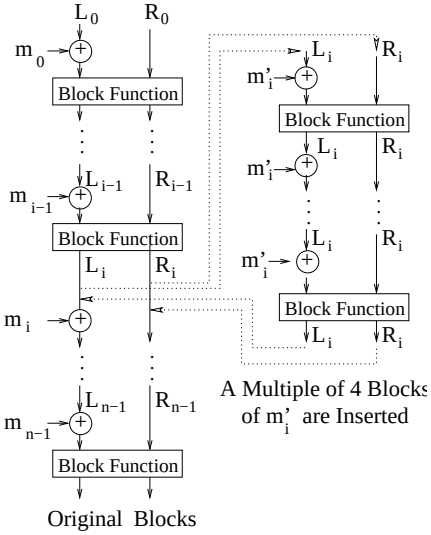

Fig. 6. *The Packet Forgery Attack*

6.2 A Packet Forgery Attack

A packet forgery attack could be mounted against Michael if the output of the block function in any round is equal to any of the fixed points.

Theorem 4. *Given a message M_1 and an arbitrary key K, an attacker can always construct a message M_2 distinct from M_1 such that Michael(K, M_1) = Michael(K, M_2) if the following condition holds.*

1. *The output of the block function of Michael(K, M_1) in any round is equal to any of the fixed points.*

Proof. Suppose M_1 has n blocks, and is written as (m_0, m_1, ..., m_{n-1}). Suppose the output of block function in any round, say in the $(i+1)$-th round (the corresponding message is m_i), is equal to any of the fixed points (assume this point is (L_i, R_i)). Given a fixed point (L_i, R_i), we can find a corresponding m'_i from the fixed-point table. A multiple of four blocks of message m'_i can be appended to the $(i+1)$-th round without changing the Michael value. The reason why the number of the inserted blocks of m'_i is a multiple of four is due to the padding method of Michael. In other words, we need to guarantee $length(M_1)$ mod 4 = $length(M_2)$ mod 4. Thus, M_2 can be constructed as (m_0, m_1, ..., m_i, $<$ m'_i, m'_i, ..., m'_i, $>$, m_{i+1}, ..., m_{n-1}), where the number of the inserted blocks of m'_i is a multiple of four. According to the property of fixed points, we have Michael(K, M_1) = Michael(K, M_2).

Remark:

1. If Condition 1 in Theorem 4 holds, an attacker can forge a message M_2 to replace the original message M_1 without modifying the Michael value, and this packet forgery attack can apply to any key K.
2. Although 95.02%-97.36% of R_i may give fixed points, it does not mean that the packet forgery attack would have a successful rate of 95.02%-97.36% since the attack requires both L_i and R_i to be matched with any fixed point, not just R_i.
3. We note that the packet forgery attack is still in theory as the message and the hash value are encrypted by RC4. Hence an attacker needs to know the decryption before mounting such a forgery attack against Michael.

7 Conclusions

Michael was designed as the message integrity code for the IEEE 802.11i. In this paper, by providing a necessary and sufficient condition for finding collisions of Michael, we showed that the collision status of Michael only depends on the second last block message and the output of its third last round. Therefore, to find collisions of Michael, we only need to focus on its two rounds: the third last round and the second last round. In addition, we demonstrated that Michael is not collision-free. Moreover, we proposed a simple method to find fixed points of Michael and built a fixed-point table based on our results. The high percentage of the existence of fixed points should be considered as a weakness of Michael. If the output of the block function in any round is in the fixed-point table, a packet forgery attack could be mounted against Michael.

References

1. Draft Amendment to Standard For Telecommunications and Information Exchange Between Systems - LAN/MAN Specification Requirements - Part 11: Wireless

Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Security Enhancements. Document Number IEEE Std 802.11i/D7.0. October 2003.
2. Advanced Encryption Standard. National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce. November 2001.
3. ANSI/IEEE Std 802.11, 1999 Edition. Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
4. W. Arbaugh, N. Shankar, and Y.C. Wan. Your 802.11 Wireless Network has No Clothes. In *Proceedings of IEEE International Conference on Wireless LANs and Home Networks*, pages 131–144, Singapore, 2001.
5. N. Borisov, I. Goldberg, and D. Wagner. Intercepting Mobile Communications: The Insecurity of 802.11. In *Proceedings of the 7th Annual Internaltional Conference on Mobile Computing and Networking*, pages 180–189, Rome, Italy, 2001.
6. N. Ferguson. Michael: an improved MIC for 802.11 WEP. *IEEE 802.11 doc 02-020r0*, 17 January 2002. http://grouper.ieee.org/groups/802/11/Documents/DocumentHolder/2-020.zip.
7. S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography*, pages 1–24, Toronto, Canada, 2001.
8. R. Johnson. *Miller & Freund's Probability and Statistics for Engineers*. Prentice Hall, sixth edition, 2000.
9. R. Rivest. The RC4 Encryption Algorithm, RSA Data Security Inc., (Proprietary). March 1992.
10. A. Stubblefield, J. Ioannidis, and A. Rubin. Using the Fluhrer, Mantin, and Shamir Attack to Break WEP. In *Proceedings of the 2002 Network and Distributed Systems Security Symposium*, pages 17–22, San Diego, California, 2002.
11. A. Wool. A Note on the Fragility of the "Michael" Message Integrity Code. *IEEE Transactions on Wireless Communications*, 2004.

Appendix

A Function InvMichael [11]

Algorithm A.1: INVMICHAEL$((v_0, v_1), (m_0, ..., m_{n-1}))$

Input : **Michael value** (v_0, v_1)
Input : **Padded message** $(m_0, ..., m_{n-1})$
Output : **Key** (k_0, k_1)
$(L, R) \leftarrow (v_0, v_1)$
for $i \leftarrow n-1$ **downto** 0
do $\begin{cases} (L, R) \leftarrow B^{-1}(L, R)(Algorithm A.2) \\ L \leftarrow L \oplus m_i \end{cases}$
return (L, R)

Algorithm A.2: $B^{-1}(L, R)$

Input : (L, R)
Output : (L, R)
$L \leftarrow (L - R) \bmod 2^{32}$
$R \leftarrow R \oplus (L >>> 2)$
$L \leftarrow (L - R) \bmod 2^{32}$
$R \leftarrow R \oplus (L <<< 3)$
$L \leftarrow (L - R) \bmod 2^{32}$
$R \leftarrow R \oplus XSWAP(L)$
$L \leftarrow (L - R) \bmod 2^{32}$
$R \leftarrow R \oplus (L <<< 17)$
return (L, R)

B Fixed Points in $[0, 2^{10}]$ and $[2^{32} - 2^{10}, 2^{32}]$

We found 984 fixed points for R_i in $[0, 2^{10}]$, and found 992 fixed points for R_i in $[2^{32}\text{-}2^{10}, 2^{32}]$. Therefore, the percentage of having fixed points for R_i in $[0, 2^{10}]$ and $[2^{32}\text{-}2^{10}, 2^{32}]$ is $\frac{984+992}{1024*2} * 100\% = 96.48\%$.

C Soundness of Remark in Section 6.1

By employing the fixed-point finding procedure, we found 985 fixed points for 2^{10} randomly chosen R_i (in [2^{10}, 2^{32}-2^{10}]). Let p denote the probability of having a fixed point at a particular value of R_i. To find out the large sample confidence interval for p, we use a formula described on Page 287 in [8]:

$$\frac{x}{n} - z_{\alpha/2}\sqrt{\frac{\frac{x}{n}(1-\frac{x}{n})}{n}} < p < \frac{x}{n} + z_{\alpha/2}\sqrt{\frac{\frac{x}{n}(1-\frac{x}{n})}{n}}$$

where n is the number of trials, x stands for number of fixed points within n trials, α is called confidence coefficient and (1-α)100% is the degree of confidence. In our case, $x = 985$ and $n = 1024$. To construct a 95% confidence interval for p, set $\alpha = 0.05$, and the corresponding $z_{\alpha/2}$ $= z_{0.025}$ =1.96. Thus

$$\frac{985}{1024} - 1.96\sqrt{\frac{\frac{985}{1024}(1-\frac{985}{1024})}{1024}} < p < \frac{985}{1024} + 1.96\sqrt{\frac{\frac{985}{1024}(1-\frac{985}{1024})}{1024}}$$

Therefore, $95.02\% < p < 97.36\%$. This means that we can be (1-0.05)100% = 95% certain that p lies in the range.

D The Fixed-Point Tables

Due to the limited space, we only provide the last 42 fixed points in Table 2 and some other fixed points in Table 3. All numbers are hexadecimal and listed in increasing order according to R_i.

X_i	m_i	L_i	R_i	X_i	m_i	L_i	R_i
98eaaa61	42f31305	da19b964	fffffd5	93a5505f	19b153d9	8a140386	ffffffee
83dca2c7	12954f62	9149eda5	fffffd7	dd64fd01	42fec977	9f9a3476	fffffff0
b2a06b85	a1701b04	13d07081	fffffd7	33b0f7ed	927a33bb	a1cac456	fffffff1
94f3a93a	eccb5821	7838f11b	fffffd9	4206e469	21a681e0	63a06589	fffffff1
9acdc5b9	6486c494	fe4b012d	fffffd9	5a597e04	d43809cd	8e6177c9	fffffff1
88b0d779	166c5707	9edc807e	fffffda	1f768e50	3380ef32	2cf66162	fffffff6
e89c0366	fc1d4464	14814702	fffffda	7e23ad2f	af25a2cb	d1060fe4	fffffff6
f220391c	d0c65937	22e6602b	fffffdc	f6aa7bf1	87fe7776	71540c87	fffffff6
396899ce	140c733	38285efd	fffffde	b6564d6	d1586214	da3d06c2	fffffff7
53e460ac	80e56446	d30104ea	fffffe1	765bb666	24617a70	523acc16	fffffff7
5b34996a	a0b67cb6	fb82e5dc	fffffe2	e47aaa65	a02e5e9d	4454f4f8	fffffff7
acbea3fc	e238a558	4e8606a4	fffffe2	eb1060bd	3611b2f6	dd01d24b	fffffff8
9aafb621	24caf35c	be65457d	fffffe4	b6e8f390	ee45aa4e	58ad59de	fffffff9
7192db3d	dc4dce8d	addf15b0	fffffe7	d4aeab65	7f81fe5	d356b480	fffffff9
96896acb	389c7bdc	ae151117	fffffe8	362975f	6df65365	6e94c43a	fffffffb
ca8549b9	5a21cd0d	90a484b4	fffffe8	1baa6340	a2776ccd	b9dd0f8d	fffffffb
2efef308	1d5586f9	33ab75f1	fffffe9	377c5a93	83f1d505	b48d8f96	fffffffb
b3623a87	cb850c8e	78e73609	fffffeb	9efcc309	1919c896	87e50b9f	fffffffb
c2845c57	d5dc4ece	17581299	fffffeb	10515aba	e5099741	f558cdfb	fffffffc
c5882938	37fe781f	f2765127	fffffeb	e072bac4	37455903	d737e3c7	fffffffc
3462948b	65dc31d2	51bea559	fffffec	fa88653f	44a52455	be2d416a	fffffffc

Table 2. Last 42 Fixed Points of Michael

X_i	m_i	L_i	R_i	X_i	m_i	L_i	R_i
bab59d13	7039495e	ca8cd44d	56d0f24a	156105e2	60ae19fd	75cf1c1f	b363e95c
785d0d44	9139ef44	e964e200	5876a915	508d2ea1	c2db8cc	5ca0966d	b46efd97
36b28048	5c606a1e	6ad2ea56	6ea8c9b0	454df193	745e0b81	3113fa12	b9bdf654
b7a9eda2	1757f39d	a0fe1e3f	6ea8c9b0	c665050d	94e81acf	528d1fc2	bd38d8a9
2ee2a357	f7e73708	d905945f	7065a177	e566754	d552699	30341cd	c12f786f
bb7f114d	13a126ad	a8de37e0	7122d92d	476c45bd	7fcbe95c	38a7ace1	c12f786f
73eba05e	d1da8dd0	a2312d8e	767a020c	f461e180	d04b8d83	242a6c03	c12f786f
9c34285c	c50a5358	593e7b04	767a020c	dfadad82	40633eb8	9fce933a	d2189dbf
96252cff	a6f5b76a	30d09b95	7b9d2c01	5e758dac	da68fe6a	841d73c6	d6c1f8f1
2267bbe6	fec622da	dca1993c	7f95fe04	6d1a5bae	14c8c243	79d299ed	d6c1f8f1
8aa3ff7a	4df2b026	c7514f5c	8baaa4a7	9023af12	bfb4e8f5	2f9747e7	d6f626af
5eecac7e	dc6fe8cd	828344b3	8c6b4530	31ee2570	9c9d012d	ad73245d	e029a075
a798df6f	85011375	2299cc1a	944952f2	3a3e18a4	e2534cda	d86d547e	e029a075
21a9a736	a59e7078	8437d74e	9e9da80e	181faf42	ab27a090	b3380fd2	e3b08beb
8d7679cd	783b397c	f54d40b1	9e9da80e	68565963	73e8805f	1bbed93c	f32888da
76532cab	7a676eb9	c344212	a2a7052c	3851550	13e85141	106d4411	f7f3a43a
7c3a551d	46568811	3a6cdd0c	ad38a213	633fc021	86e63d32	e5d9fd13	f7f3a43a
4778464d	92c8159d	d5b053d0	b1785966	73a02eda	92ba55b	7a8b8b81	f7f3a43a
7b91187d	5463753a	2ff26d47	b1785966	1e8dabb4	b593607f	ab1ecbcb	f9bb6d58
97af78ca	775e5f93	e0f12759	b1785966	462fd533	c8d38102	8efc5431	f9bb6d58
f38ddf1f	32abe5f7	c1263ae8	b1785966	c84c4c2e	e3bd8f2f	2bf1c301	f9bb6d58

Table 3. Some Other Fixed Points for R_i in $[2^{10}, 2^{32} - 2^{10}]$