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

Jianyong Huang, Jennifer Seberry, Willy Susilo, and Martin Bunder

University of Wollongong, NSW 2522, Australia
{jyh33, jennie, wsusilo, mbunder}@uow.edu.au

Abstract. The latest IEEE 802.11i uses a 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 method to find fixed points of Michael. 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. Since the Michael value is encrypted by RC4, the proposed packet forgery attack does not endanger the security of the whole TKIP system.

1 Introduction

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 [8] 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 [9] 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 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. TKIP is considered as a temporary solution, and it is designed for legacy hardware. For the

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 423–432, 2005.

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.

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: $\lll$ is left rotation, $\ggg$ represents right rotation, $\oplus$ is exclusive-or, $\boxplus$ stands for addition modulo 2^{32}, $||$ is concatenation, and $\Longrightarrow$ means "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 [6] is the message integrity code (MIC) of TKIP in the IEEE 802.11i [1]. 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. 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)
$R \leftarrow R \oplus (L \lll 17)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus XSWAP(L)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus (L \lll 3)$
$L \leftarrow (L + R) \bmod 2^{32}$
$R \leftarrow R \oplus (L \ggg 2)$
$L \leftarrow (L + R) \bmod 2^{32}$
return (L, R)

Michael employs several operations, including exclusive-or, left rotation, right rotation, addition modulo 2^{32} and swapping ($XSWAP$). Swapping 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

Wool found one weakness of Michael: it is *not* one-way, in fact, it is *invertible* [10]. There exists a simple inverse function, which can recover the secret Michael key K, given a known message M and its corresponding Michael value MIC = Michael(K, M). We note that the block function is unkeyed, and every step in Michael is invertible, therefore the whole Michael algorithm is invertible.

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 [10].

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 study 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 1 in Appendix A. We provide the last round and the second last round of Michael(Key_1, M_1) in Algorithm 4.1 and Algorithm 4.2 respectively. Similarly, the last round and the second last round of Michael(Key_2, M_2) are shown in Algorithm B.1 and Algorithm B.2 in Appendix B respectively.

Algorithm 4.1: LAST ROUND (Key_1, M_1)

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

Algorithm 4.2: 2RD LAST (Key_1, M_1)

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

Necessary condition: If Michael(Key_1, M_1) = Michael(Key_2, M_2), namely the collisions occur, we then backtrack from Step 11 and 10 in Algorithm 4.1 and B.1.

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

$$L_2^{x-1} = L_2'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow L_1^{x-1} = L_1'^{y-1},$$
$$L_1^{x-1} = L_1'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow R_0^{x-1} = R_0'^{y-1}.$$

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).

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

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 B.2.

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

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 $m'_{y-1} = 0$.

$$L_5^{x-2} = L_5'^{y-2},\ L_0^{x-1} = L_5^{x-2} \text{ and } L_0'^{y-1} = L_5'^{y-2} \Longrightarrow L_0^{x-1} = L_0'^{y-1},$$
$$R_4^{x-2} = R_4'^{y-2},\ R_4^{x-2} = R_0^{x-1} \text{ and } R_4'^{y-2} = R_0'^{y-1} \Longrightarrow R_0^{x-1} = R_0'^{y-1},$$
$$L_0^{x-1} = L_0'^{y-1} \text{ and } m_{x-1} = m'_{y-1} \Longrightarrow L_1^{x-1} = L_1'^{y-1},$$
$$L_1^{x-1} = L_1'^{y-1} \text{ and } R_0^{x-1} = R_0'^{y-1} \Longrightarrow R_1^{x-1} = R_1'^{y-1},$$
$$L_1^{x-1} = L_1'^{y-1} \text{ and } R_1^{x-1} = R_1'^{y-1} \Longrightarrow L_2^{x-1} = L_2'^{y-1},$$

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

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).

Therefore, $R_4^{x-3} = R_4'^{y-3}$ and $L_5^{x-3} \oplus L_5'^{y-3} = m_{x-2} \oplus m'_{y-2}$ 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 2 in Appendix A. 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. □

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 3 in Appendix C illustrates one round of Michael. In Figure 3, we note 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. 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 takes 2-3 minutes to decide whether there exists a fixed point for a given R_i. For example, $(L_i, R_i, m_i) = (0x3f651087, 0x2, 0xbbac8b1a)$ is a fixed point. A more complete fixed-point table is provided in the full paper.

6.2 A Packet Forgery Attack

A packet forgery attack (depicted in Figure 4 in Appendix C) 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.We note that the packet forgery attack does not endanger the entire TKIP system 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. 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. The packet forgery attack does not endanger security of the whole TKIP system as the Michael value is encrypted by RC4. To make the proposed forgery attack practical to TKIP, the attacker needs to consider the combination of Michael and RC4.

References

1. 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 Amendment 6: Medium Access Control (MAC) Security Enhancements. 23 July 2004.

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. Rivest. The RC4 Encryption Algorithm, RSA Data Security Inc., (Proprietary). March 1992.
9. 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.
10. A. Wool. A Note on the Fragility of the "Michael" Message Integrity Code. *IEEE Transactions on Wireless Communications*, 2004.

A Three-Round Diagrams

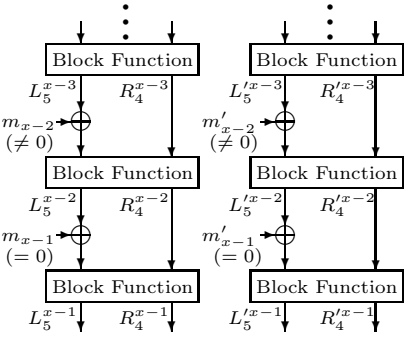

Fig. 1. *Last Three Rounds of Michael*

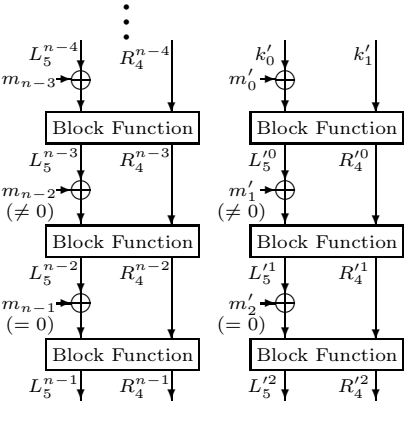

Fig. 2. *The Construction of* (K', M')

B Algorithms in Section 4

Algorithm B.1: LAST ROUND (Key_2, M_2)

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

Algorithm B.2: 2RD LAST (Key_2, M_2)

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

C Figures

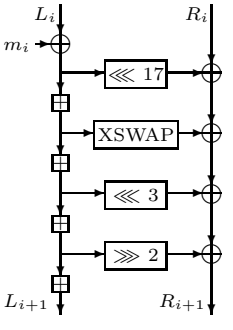

Fig. 3. *One Round of Michael*

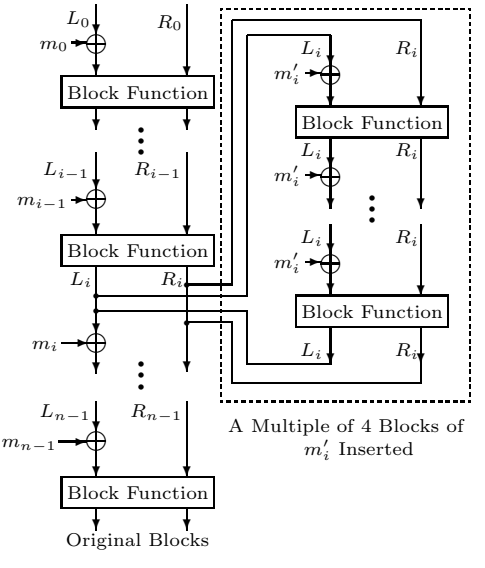

Fig. 4. *The Packet Forgery Attack*