2005

Security analysis of Michael: the IEEE 802.11i message integrity code

Jianyong Huang
jyh33@uow.edu.au

Jennifer Seberry
University of Wollongong, jennie@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Martin W. Bunder
University of Wollongong, mbunder@uow.edu.au

Publication Details
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.

Keywords
Security, Analysis, Michael, IEEE, 802, 11i, Message, Integrity, Code

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/1598
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.

![TKIP Encryption Diagram](image)

**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 $M = (m_0,$
$m_1, \ldots, m_{n-1})$. $L^i_0, R^i_0, L^i_1, R^i_1, L^i_2, R^i_2, L^i_3, R^i_3, L^i_4, R^i_4$, and $L^i_5$ 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 $(L^i_5, R^i_4)$, where $L^i_5$ stands for the left half of the output and $R^i_4$ stands for the right half of the output. Some other notations used in this paper are listed as follows:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC</td>
<td>Medium access control</td>
</tr>
<tr>
<td>&lt;&lt;&lt;</td>
<td>Left rotation</td>
</tr>
<tr>
<td>&gt;&gt;&gt;</td>
<td>Right rotation</td>
</tr>
<tr>
<td>⊕</td>
<td>Exclusive-or</td>
</tr>
<tr>
<td>⊕</td>
<td>Addition modulo $2^{32}$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>⊸</td>
<td>Implies</td>
</tr>
</tbody>
</table>

**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**

Input: Key $(k_0, k_1)$  
Input: Padded message $(m_0, ..., m_{n-1})$  
Output: MIC value $(L, R)$

$(L, R) = (k_0, k_1)$

for $i \leftarrow 0$ to $n - 1$

\[
\begin{align*}
(L, R) & \leftarrow B(L, R) (\text{Algorithm 2.2}) \\
& \text{return } (L, R)
\end{align*}
\]

**Algorithm 2.2: B(L, R)**

Input: $(L, R)$  
Output: $(L, R)$

\[
\begin{align*}
R & \leftarrow R \oplus (L <<< 17) \\
L & \leftarrow (L + R) \mod 2^{32} \\
R & \leftarrow R \oplus XSWAP(L) \\
L & \leftarrow (L + R) \mod 2^{32} \\
R & \leftarrow R \oplus (L <<< 3) \\
L & \leftarrow (L + R) \mod 2^{32} \\
R & \leftarrow R \oplus (L >>> 2) \\
L & \leftarrow (L + R) \mod 2^{32}
\end{align*}
\]

\[
\text{return } (L, R)
\]

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 $MIC = \text{Michael}(K, M)$. 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, \((\text{Key}_1, M_1)\) and \((\text{Key}_2, M_2)\), \(\text{Michael}(\text{Key}_1, M_1) = \text{Michael}(\text{Key}_2, M_2)\) if and only if the following two conditions hold:

1. \(R_{x-3}^x = R_{y-3}^y\)
2. \(L_{x-2}^x \oplus L_{y-2}^y = m_{x-2} \oplus m_{y-2}'\)

where \(M_1\) has \(x\) 32-bit blocks, \(M_2\) has \(y\) 32-bit blocks, and both \(x\) and \(y\) are \(\geq 3\).

**Proof.** The last three rounds of Michael are illustrated in Figure 3. We
provide the last round and the second last round of Michael(\textit{Key}_1, M_1) in Algorithm 4.1 and Algorithm 4.2 respectively.

\textbf{Algorithm 4.1: Last Round (Key}_1, M_1)\begin{align*}
1. & L_{x-1} = L_{x-1}^y - 2 \\
2. & R_{0}^{x-1} = R_{4}^{x-2} \\
3. & L_{x}^{1} = L_{x}^{1} \oplus m_{x-1} \\
4. & R_{4}^{y-1} = R_{0}^{y-1} \oplus (L_{x-1}^{1} <<< 17) \\
5. & L_{2}^{x-1} = (L_{x}^{1} + R_{x}^{y-1}) \text{ mod } 2^{32} \\
6. & R_{2}^{x-1} = R_{2}^{x-1} \oplus XSWAP(L_{2}^{x-1}) \\
7. & L_{5}^{x-1} = (L_{2}^{x-1} + R_{2}^{x-1}) \text{ mod } 2^{32} \\
8. & R_{5}^{x-1} = R_{5}^{x-1} \oplus (L_{x}^{1} <<< 3) \\
9. & L_{4}^{x-1} = (L_{x}^{1} + R_{x}^{y-1}) \text{ mod } 2^{32} \\
10. & R_{4}^{x-1} = R_{4}^{x-1} \oplus (L_{x-1}^{1} >>> 2) \\
11. & L_{5}^{x-1} = (L_{x}^{1} + R_{x}^{y-1}) \text{ mod } 2^{32} \\
(Note: Michael(\textit{Key}_1, M_1) = (L_{x-1}^{1}, R_{4}^{x-1}))
\end{align*}

Similarly, the last round and the second last round of Michael(\textit{Key}_2, M_2) are shown in Algorithm 4.3 and Algorithm 4.4 respectively.

\textbf{Algorithm 4.3: Last Round (Key}_2, M_2)\begin{align*}
1. & L_{y-1}^{x} = L_{y-1}^{x} - 2 \\
2. & R_{0}^{y-1} = R_{y-2}^{x} \\
3. & L_{3}^{y-1} = L_{3}^{y-1} \oplus m_{y-1} \\
4. & R_{4}^{y-1} = R_{0}^{y-1} \oplus (L_{y-1}^{x} <<< 17) \\
5. & L_{2}^{y-1} = (L_{y}^{x-1} + R_{y}^{y-1}) \text{ mod } 2^{32} \\
6. & R_{2}^{y-1} = R_{2}^{y-1} \oplus XSWAP(L_{2}^{y-1}) \\
7. & L_{5}^{y-1} = (L_{2}^{y-1} + R_{2}^{y-1}) \text{ mod } 2^{32} \\
8. & R_{5}^{y-1} = R_{5}^{y-1} \oplus (L_{y}^{x-1} <<< 3) \\
9. & L_{4}^{y-1} = (L_{y}^{x-1} + R_{y}^{y-1}) \text{ mod } 2^{32} \\
10. & R_{4}^{y-1} = R_{3}^{y-1} \oplus (L_{y}^{x-1} >>> 2) \\
11. & L_{5}^{y-1} = (L_{y}^{x-1} + R_{y}^{y-1}) \text{ mod } 2^{32} \\
(Note: Michael(\textit{Key}_2, M_2) = (L_{y-1}^{x}, R_{4}^{y-1}))
\end{align*}

\textbf{Algorithm 4.4: 2nd Last (Key}_2, M_2)\begin{align*}
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_{y-2}^{y-2} <<< 17) \\
5. & L_{2}^{y-2} = (L_{y}^{y-2} + R_{y}^{y-2}) \text{ mod } 2^{32} \\
6. & R_{2}^{y-2} = R_{y}^{y-2} \oplus XSWAP(L_{y}^{y-2}) \\
7. & L_{3}^{y-2} = (L_{y}^{y-2} + R_{y}^{y-2}) \text{ mod } 2^{32} \\
8. & R_{3}^{y-2} = R_{y}^{y-2} \oplus (L_{y}^{y-2} <<< 3) \\
9. & L_{4}^{y-2} = (L_{y}^{y-2} + R_{y}^{y-2}) \text{ mod } 2^{32} \\
10. & R_{4}^{y-2} = R_{3}^{y-2} \oplus (L_{y}^{y-2} >>> 2) \\
11. & L_{5}^{y-2} = (L_{y}^{y-2} + R_{y}^{y-2}) \text{ mod } 2^{32} \\
(Note: Michael(\textit{Key}_2, M_2) = (L_{y-1}^{x}, R_{4}^{y-1}))
\end{align*}

\textbf{Necessary Condition}: If Michael(\textit{Key}_1, M_1) = Michael(\textit{Key}_2, M_2), namely the collision occurs, we then backtrack from Step 11 and 10 in Algorithm 4.1 and 4.3.

\begin{align*}
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies L_{x-1} = L_{y-1} - 1, \\
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies R_{x-1} = R_{y-1} - 1, \\
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies R_{x-1} = R_{y-1} - 1, \\
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies L_{x-1} = L_{y-1} - 1, \\
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies R_{x-1} = R_{y-1} - 1, \\
L_{x-1} = L_{y-1} - 1, R_{x-1} = R_{y-1} - 1 & \implies R_{x-1} = R_{y-1} - 1.
\end{align*}

As \[ L_{1}^{x-1} = L_{1}^{x-1} \oplus m_{x-1}, L_{1}^{y-1} = L_{0}^{y-1} \oplus m_{y-1}, L_{x-1} = L_{x-1}^{x-2}, L_{0}^{y-1} = L_{0}^{y-1} \oplus m_{y-1}, L_{x-1} = L_{y-2}^{x-2}, L_{0}^{y-1} = L_{0}^{y-1} \oplus m_{y-1}, m_{x-1} = 0 \text{ and } m_{y-1} = 0, \text{ therefore} \]
\[ L_{x-1}^{x-2} = L_{y-2}^{x-2} \text{ and } R_{x-1}^{x-2} = R_{y-2}^{x-2}. \]
Similarly, we use the same method in the second last rounds of Michael($Key_1$, $M_1$) and Michael($Key_2$, $M_2$).

\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies L_{x-2}^x = L_{y-2}^y,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies R_{x-2}^x = R_{y-2}^y,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies R_{x-2}^x = R_{y-2}^y,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies L_{x-2}^x = L_{y-2}^y.
\]

As \(L_1^{x-2} = L_0^{x-2} \oplus m_{x-2}, L_1^{y-2} = L_0^{y-2} \oplus m'_{y-2}, L_0^{x-3} = L_0^{y-3} \text{ and } L_0^{y-2} = L_0^{y-3}, \) therefore \(L_0^{x-3} \oplus L_0^{y-3} = m_{x-2} \oplus m'_{y-2}. \) As \(R_{x-2}^x = R_{4-2}^x \text{ and } R_{0}^{y-2} = R_{4-2}^{y-3}, \) therefore \(R_{x-2}^x = R_{4-2}^{y-2}. \)

Thus, Michael($Key_1$, $M_1$) = Michael($Key_2$, $M_2$) \(\implies R_{x-3}^4 = R_{4-3}^{y-3} \text{ and } L_{0}^{x-3} \oplus L_{0}^{y-3} = m_{x-2} \oplus m'_{y-2}. \)

**Sufficient Condition:** If \(R_{4-3}^4 = R_{4-3}^{y-3} \text{ and } L_{0}^{x-3} \oplus L_{0}^{y-3} = m_{x-2} \oplus m'_{y-2} \) hold, we start from Step 1 and 2 in Algorithm 4.2 and 4.4.

\[
L_{5-3}^x = L_{5-3}^y \quad \text{and} \quad L_{5-3}^x \oplus L_{5-3}^y = m_{x-2} \oplus m'_{y-2} \implies L_{x-2}^1 = L_{y-2}^1,
\]
\[
R_{4-3}^x = R_{4-3}^y \quad \text{and} \quad R_{4-3}^x \oplus R_{4-3}^y = m_{x-2} \oplus m'_{y-2} \implies R_{x-2}^0 = R_{y-2}^0,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies L_{x-2}^x = L_{y-2}^y,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies R_{x-2}^x = R_{y-2}^y,
\]
\[
L_{x-2}^x = L_{y-2}^y \quad \text{and} \quad R_{x-2}^x = R_{y-2}^y \implies L_{x-2}^x = L_{y-2}^y.
\]

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

\[
L_{x-2} = L_{y-2}, \quad L_{x-1} = L_{y-2} \quad \text{and} \quad L_{y-1} = L_{y-2} \implies L_{x-1} = L_{y-1},
\]
\[
R_{x-2} = R_{y-2}, \quad R_{x-2} = R_{y-2} \quad \text{and} \quad R_{y-2} = R_{y-2} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies L_{x-1} = L_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies L_{x-1} = L_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies L_{x-1} = L_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies L_{x-1} = L_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies L_{x-1} = L_{y-1},
\]
\[
L_{x-1} = L_{y-1} \quad \text{and} \quad R_{x-1} = R_{y-1} \implies R_{x-1} = R_{y-1}.
\]
Therefore, $R_4^{x-3} = R_4^{y-3}$ and $L_5^{x-3} \oplus L_5^{y-3} = m_{x-2} \oplus m'_{y-2} \implies \text{Michael}(\text{Key}_1, M_1) = \text{Michael}(\text{Key}_2, M_2)$.

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 $\text{Michael}(\text{Key}_1, M_1) = \text{Michael}(\text{Key}_2, M_2)$.

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 $\text{Michael}(K, M) = \text{Michael}(K', M')$.

**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 $\text{Michael}(K, M) = \text{Michael}(K', M')$, where $M$ has $n$ 32-bit blocks and $n$ is any integer $\geq 3$.

**Proof.** We write $M$ as $(m_0, m_1, \ldots, m_{n-1})$, 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 $\text{Michael}(K, M)$ 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 $\text{Michael}(K', M')$ are represented as $(L_5^0, R_4^0), (L_5^1, R_4^1)$ 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.

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

Therefore, $\text{Michael}(K, M) = \text{Michael}(K', M')$ 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 \(\text{Michael}((L_i, R_i), m_i) = (L_i, R_i)\). 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 that \(\text{Michael}((L_i, R_i), m_i) = (L_{i+1}, R_{i+1})\). In the finding procedure, our goal is to find a triple \((L_i, R_i, m_i)\) such that \(\text{Michael}((L_i, R_i), m_i) = (L_{i+1}, R_{i+1})\).
\((L_{i+1}, R_{i+1}) = (L_i, R_i)\). 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. \(\text{FOR } (X_i = 0; X_i \leq 2^{32}; X_i++)\)
   \(\text{(a) Call block function } B(X_i, R_i)\)
   \(\text{(b) IF } R_i = R_{i+1} \text{ THEN}\)
   \(\text{i. There exists an } X_i \text{ such that } R_i = R_{i+1}. \text{ For a found } X_i, \text{ there exists a corresponding } L_{i+1} \text{ because the mapping from } (X_i, R_i) \text{ to } (L_{i+1}, R_{i+1}) \text{ is bijective. The reason why the mapping from } (X_i, R_i) \text{ to } (L_{i+1}, R_{i+1}) \text{ is bijective is that Michael is invertible. Choose } L_i = L_{i+1}.\)
   \(\text{ii. Choose } m_i = X_i \oplus L_i.\)
   \(\text{iii. Increase counter } c \text{ by one.}\)
3. \(\text{IF counter } c = 0 \text{ THEN no fixed point found for this } R_i.\)
4. \(\text{ELSE There are } c \text{ fixed points for this } R_i.\)

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 \(\text{Michael}((L_i, R_i)), m_i) = (L_i, R_i).\) 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.

<table>
<thead>
<tr>
<th>(X_i)</th>
<th>(m_i)</th>
<th>(L_i)</th>
<th>(R_i)</th>
<th>(X_i)</th>
<th>(m_i)</th>
<th>(L_i)</th>
<th>(R_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5207d86f</td>
<td>1e2c2710</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6c06529a</td>
<td>886b395</td>
<td>19480c10f</td>
<td>0</td>
<td>6e66938</td>
<td>d44d5dd2</td>
<td>d2ab34ea</td>
<td>9</td>
</tr>
<tr>
<td>4e91dea2</td>
<td>7161872</td>
<td>4987c6d0</td>
<td>1</td>
<td>8381416d</td>
<td>5fc4b0d</td>
<td>dc4d0a60</td>
<td>9</td>
</tr>
<tr>
<td>54e8bce4</td>
<td>69bd6b8c</td>
<td>3d52d7ba</td>
<td>1</td>
<td>299915c</td>
<td>ba9f2472</td>
<td>4896b52e</td>
<td>9</td>
</tr>
<tr>
<td>84c99b9d</td>
<td>bba8b1a</td>
<td>3f651087</td>
<td>2</td>
<td>80bd5b57</td>
<td>8ebe3dfe</td>
<td>10388a8c</td>
<td>e</td>
</tr>
<tr>
<td>5e8f604</td>
<td>a026b006</td>
<td>fca17602</td>
<td>3</td>
<td>49992f30</td>
<td>41951984</td>
<td>4d1ce0b4</td>
<td>e</td>
</tr>
<tr>
<td>5e9936a</td>
<td>16439092</td>
<td>4addbaa8</td>
<td>3</td>
<td>35848ac</td>
<td>4be1c3b</td>
<td>48a65497</td>
<td>d</td>
</tr>
<tr>
<td>a99e358e</td>
<td>1c308e95</td>
<td>55076b19</td>
<td>3</td>
<td>5557549c</td>
<td>64902b1</td>
<td>31e8626d</td>
<td>d</td>
</tr>
<tr>
<td>b5d2b3a7</td>
<td>f14a38f0</td>
<td>8b911357</td>
<td>3</td>
<td>7e0332e2</td>
<td>1538792</td>
<td>69355570</td>
<td>d</td>
</tr>
<tr>
<td>a8d11208</td>
<td>77718177c</td>
<td>6fa90514</td>
<td>4</td>
<td>3918e91</td>
<td>9ca7035d</td>
<td>47bf3dee</td>
<td>e</td>
</tr>
<tr>
<td>578f1083c</td>
<td>960dcede</td>
<td>e1c8cfe2</td>
<td>5</td>
<td>5eecc6a1</td>
<td>b6f114c5</td>
<td>33bda264</td>
<td>e</td>
</tr>
<tr>
<td>6ac32cee</td>
<td>60884be2</td>
<td>a4b652d</td>
<td>5</td>
<td>a9a9eef</td>
<td>d7e790</td>
<td>0d7d792f</td>
<td>e</td>
</tr>
<tr>
<td>b2dcf2ad</td>
<td>68ac3e02</td>
<td>6470145f</td>
<td>5</td>
<td>bc4846a</td>
<td>2abd84ca</td>
<td>96f20090</td>
<td>e</td>
</tr>
<tr>
<td>fc231200</td>
<td>d0fd4eb9</td>
<td>2c5e5cb9</td>
<td>5</td>
<td>fadce69</td>
<td>976b78c1</td>
<td>d7a6b8a7</td>
<td>f</td>
</tr>
<tr>
<td>e5cdd7a7</td>
<td>673b6f6c6</td>
<td>a4b622bc</td>
<td>6</td>
<td>afb1715f</td>
<td>74cca5e1</td>
<td>d57d41be</td>
<td>10</td>
</tr>
<tr>
<td>e5b473b1</td>
<td>83e9090c</td>
<td>665e34d</td>
<td>7</td>
<td>d8689e66</td>
<td>2f6d168a</td>
<td>2abe88ec</td>
<td>10</td>
</tr>
</tbody>
</table>

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 ($[2^{32} - 2^{10}, 2^{32}]$) 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 $[2^{10}, 2^{32} - 2^{10}]$ 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 $2^{32} \cdot p$ (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.

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 $\text{length}(M_1) \mod 4 = \text{length}(M_2) \mod 4$. 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 $\text{Michael}(K, M_1) = \text{Michael}(K, M_2)$.

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


Appendix

A Function InvMichael [11]

Algorithm A.1: InvMichael((v0, v1), (m0, ..., mn−1))

Input : Michael value (v0, v1)
Input : Padded message (m0, ..., mn−1)
Output : Key (k0, k1)

(L, R) ← (v0, v1)
for i ← n − 1 downto 0
    do (L, R) ← B−1(L, R)(Algorithm A.2)
    L ← L ⊕ mi
return (L, R)

Algorithm A.2: B−1(L, R)

Input : (L, R)
Output : (L, R)

L ← (L − R) mod 232
R ← R ⊕ (L >>> 2)
L ← (L − R) mod 232
R ← R ⊕ (L <<< 3)
L ← (L − R) mod 232
R ← R ⊕ XSWAP(L)
L ← (L − R) mod 232
R ← R ⊕ (L <<< 17)
return (L, R)

B Fixed Points in [0, 210] and [232 − 210, 232]

We found 984 fixed points for Ri in [0, 210], and found 992 fixed points for Ri in [232−210, 232]. Therefore, the percentage of having fixed points for Ri in [0, 210] and [232−210, 232] 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]:

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

where $n$ is the number of trials, $x$ stands for number of fixed points within $n$ trials, $\alpha$ is called confidence coefficient and $(1-\alpha)100\%$ 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

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

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 $R_i$. 
Table 2. Last 42 Fixed Points of Michael

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$m_1$</th>
<th>$L_1$</th>
<th>$R_1$</th>
<th>$X_1$</th>
<th>$m_1$</th>
<th>$L_1$</th>
<th>$R_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
</tr>
<tr>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
</tr>
<tr>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
<td>2353</td>
<td>a4c14dd</td>
<td>66d0f2ca</td>
<td>3823a1</td>
</tr>
</tbody>
</table>

Table 3. Some Other Fixed Points for $R_1$ in $[2^{10}, 2^{32} − 2^{10}]$