2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11

Remedying the Hummingbird Cryptographic Algorithm

Xinxin Fan and Guang Gong
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada

Honggang Hu
School of Information Science and Technology
University of Science and Technology of China
Hefei, 230027, China

***Abstract*—Hummingbird is a recently proposed lightweight cryptographic algorithm for securing RFID systems. In 2011, Saarinen reported a chosen-IV, chosen-message attack on Hummingbird in FSE'11. In this paper, we propose a lightweight remedial scheme in response to the Saarinen's attack. The scheme is quite efficient both in software and hardware since only two cyclic shifts are involved. Using this simple tweak, we can keep the compact design of Hummingbird as well as enhance the security of Hummingbird. Readers are welcome to attack the remedial Hummingbird.**

***Keywords*-RFID, lightweight cryptography, cryptanalysis, block cipher, stream cipher, cyclic shift.**

I. Introduction

With the development of pervasive computing, smart devices such as RFID tags, smart cards, and wireless sensor nodes play more and more significant role in our life. Such applications include home automation, healthcare, supply-chain management, access control, etc. Among these applications, a lot involves the processing of sensitive information such as health or biomedical data. Hence the demand for embedded applications with cryptographic function has risen. However, classical cryptographic primitives targeted for regular computers might not be suited for resource-constrained embedded devices, e.g., AES and RSA. The mass deployments of smart devices bring forward requirements for designing new cryptographic primitives providing authentication, encryption, and other security functionalities for applications in the era of pervasive computing. This new area is referred to as lightweight cryptography.

How to deal with the trade-off among security, cost, and performance is the key issue of lightweight cryptography. A host of lightweight symmetric cryptographic algorithms appeared in the literature [1], [2], [3], [7], [8], [9], [12], [14]. Among them, the first class results from compact hardware implementations of standardized block ciphers [7], [8], the second class is from slight modifications of a classical block cipher [12], and the last class features new low-cost designs [1], [2], [3], [9], [14].

In 2010, a new ultra-lightweight cryptographic algorithm named Hummingbird was proposed in [4]. Hummingbird has a hybrid structure of block cipher and stream cipher and can be efficiently implemented on both software and hardware platforms. Later, Saarinen found a chosen-IV, chosen-message attack on Hummingbird [13]. In this paper, we demonstrate how to thwart the Saarinen's attack with three cyclic shift operations. Our lightweight remedial scheme can keep the compact design of Hummingbird and enhance the security of Hummingbird at the same time. Experimental results show that our scheme is quite efficient when implemented on low-cost 8- and 16-bit microcontrollers. Compared with Hummingbird-2 proposed recently in [5], our scheme is more compact for hardware implementation.

This paper is organized as follows. In Section II, we review the specification of Hummingbird and Saarinen's attack. In Section III, we propose our remedial scheme, and discuss our motivation. Section IV describes two one-to-one mapping in Hummingbird. The cryptanalysis is included in Section V. The implementation results on microcontrollers are reported in Section VI. Finally, Section VII concludes this paper.

II. The Hummingbird Cryptographic Algorithm and Saarinen's Attack

A. The Specification and Notations

Hummingbird has 16-bit block size, 256-bit key size, and 80-bit internal state. It is an elegant combination of block cipher and stream cipher. Henceforth we need some notations as listed in Table I. For the specification of Hummingbird, please see [4].

B. Saarinen's Attack

Henceforth we use Δ to denote the difference. In [13], Saarinen found that in the initialization phase of Hummingbird, there is a difference pair as follows:

$$\begin{aligned}&\Delta(\mathrm{NONCE}_0, \mathrm{NONCE}_1, \mathrm{NONCE}_2, \mathrm{NONCE}_3)\\&\quad = (8000, 0000, 0000, 0000)\\&\quad\quad \Downarrow\\&\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)\\&\quad = (8000, 0000, 0000, 0000, 0000).\end{aligned}$$

978-0-7695-4600-1/11 $26.00 © 2011 IEEE
DOI 10.1109/TrustCom.2011.101

Table I: Notations

PT_i	the i-th plaintext block, $i = 1, 2, \ldots, n$
CT_i	the i-th ciphertext block, $i = 1, 2, \ldots, n$
K	the 256-bit secret key
$\mathbf{E}_K(\cdot)$	the encryption function of Hummingbird with 256-bit secret key K
$\mathbf{D}_K(\cdot)$	the decryption function of Hummingbird with 256-bit secret key K
k_i	the 64-bit subkey used in the i-th block cipher, $i = 1, 2, 3, 4$, such that $K = k_1 \| k_2 \| k_3 \| k_4$
$E_{k_i}(\cdot)$	a block cipher encryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $E_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \to \{0,1\}^{16}, i = 1, 2, 3, 4$
$D_{k_i}(\cdot)$	a block cipher decryption algorithm with 16-bit input, 64-bit key k_i, and 16-bit output, i.e., $D_{k_i} : \{0,1\}^{16} \times \{0,1\}^{64} \to \{0,1\}^{16}, i = 1, 2, 3, 4$
RSi	the i-th 16-bit internal state register, $i = 1, 2, 3, 4$
LFSR	a 16-stage Linear Feedback Shift Register with the characteristic polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$
$\boxplus$	modulo 2^{16} addition operator
$\boxminus$	modulo 2^{16} subtraction operator
NONCE_i	the i-th nonce which is a 16-bit random number, $i = 1, 2, 3, 4$
NONCE	the 64-bit initial vector, such that $\mathsf{NONCE} = \mathsf{NONCE}_1 \| \mathsf{NONCE}_2 \| \mathsf{NONCE}_3 \| \mathsf{NONCE}_4$

For the first round of encryption/decryption, there is a difference pair as follows:

$$\begin{aligned}
&\Delta(\mathrm{P}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)\\
&\quad = (8000, 8000, 0000, 0000, 0000)\\
&\qquad \Downarrow\\
&\Delta(\mathrm{C}_0, \mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1)\\
&\quad = (0000, 8000, 8000, 0000, 8000, 0000).
\end{aligned}$$

Furthermore, if the difference of the international state satisfies $\Delta(\mathrm{V12}_t) = 8000, \Delta(\mathrm{V23}_t) = 0000, \Delta(\mathrm{V34}_t) = 0000$, then we have

$$\begin{aligned}
&\Delta(\mathrm{RS1}_t, \mathrm{RS2}_t, \mathrm{RS3}_t, \mathrm{RS4}_t, \mathrm{LFSR}_t)\\
&\quad = (8000, 8000, 0000, 8000, 0000)\\
&\qquad \Downarrow\\
&\Delta(\mathrm{RS1}_{t+1}, \mathrm{RS2}_{t+1}, \mathrm{RS3}_{t+1}, \mathrm{RS4}_{t+1}, \mathrm{LFSR}_{t+1})\\
&\quad = (8000, 8000, 0000, 8000, 0000, 0000).
\end{aligned}$$

Based on these differential properties of Hummingbird, Saarinen found an efficient attack on Hummingbird which can recover k_1, k_2, k_3, and k_4 one by one.

III. Remedying the Hummingbird

A. The Remedial Scheme

Note that for the addition modulo 2^{16}, the most significant bit does not affect lower bits. Moreover, for the most significant bit, the addition modulo 2^{16} is equivalent to the XOR operation. This is the weakness in the original Hummingbird design, and the differential properties found by Saarinen fully take advantages of this weakness. Therefore, if we can remove this weakness from the original Hummingbird, the Saarinen's attack will not work any more. After investigating different schemes, we found that using cyclic shift provides a lightweight solution and is able to keep the original properties of Hummingbird.

For the initialization, we have changed the plaintexts to be $(RS1_t \boxplus RS3_t) \boxplus (RS1_t \lll 8)$ instead of $(RS1_t \boxplus RS3_t) \boxplus RS1_t$ in each of four round encryptions. In this way, the most significant bit of $RS1$ will affect lower bits (see Table II for more details).

Table II: Initialization

Nonce Initialization
$RS1_{-4} = \mathsf{NONCE}_0$
$RS2_{-4} = \mathsf{NONCE}_1$
$RS3_{-4} = \mathsf{NONCE}_2$
$RS4_{-4} = \mathsf{NONCE}_3$
Four Rounds of Encryption
for t = -4 to -1 do
$V12_t = E_{k_1}((RS1_t \boxplus RS3_t) \boxplus (RS1_t \lll 8))$
$V23_t = E_{k_2}(V12_t \boxplus RS2_t)$
$V34_t = E_{k_3}(V23_t \boxplus RS3_t)$
$TV_t = E_{k_4}(V34_t \boxplus RS4_t)$
$RS1_{t+1} = RS1_t \boxplus TV_t$
$RS2_{t+1} = RS2_t \boxplus V12_t$
$RS3_{t+1} = RS3_t \boxplus V23_t$
$RS4_{t+1} = RS4_t \boxplus V34_t$
end for
LFSR Initialization
$\mathsf{LFSR}_0 = TV_{-1} \mid \mathsf{0x1000}$

Similarly, for encryption and decryption, the update equation of $RS1_{t+1}$ has been changed from $RS1_t \boxplus V34_t$ to $(RS1_t \lll 8) \boxplus V34_t$. In this case, the most significant bit of $RS1_t$ will affect lower bits of $RS1_{t+1}$. Moreover, after one more encryption round, it will affect all bits of intermediate variables (see Table III for more details).

B. Motivation

The above remedial scheme is motivated by the design of four AES finalists: Serpent, Twofish, RC6, and MARS. It is not difficult to find that several cyclic shifts are

Table III: Encryption/Decryption and Internal State Updating

Encryption	Decryption
$V12_t = E_{k_1}(PT_i \boxplus RS1_t)$	$V34_t = D_{k_4}(CT_i) \boxminus RS4_t$
$V23_t = E_{k_2}(V12_t \boxplus RS2_t)$	$V23_t = D_{k_3}(V12_t) \boxminus RS3_t$
$V34_t = E_{k_3}(V23_t \boxplus RS3_t)$	$V12_t = D_{k_2}(V23_t) \boxminus RS2_t$
$CT_i = E_{k_4}(V34_t \boxplus RS4_t)$	$PT_i = D_{k_1}(V34_t) \boxminus RS1_t$
Internal State Updating	
$\text{LFSR}_{t+1} \leftarrow \text{LFSR}_t$ $RS1_{t+1} = (RS1_t \lll 8) \boxplus V34_t$ $RS3_{t+1} = RS3_t \boxplus V23_t \boxplus \text{LFSR}_{t+1}$ $RS4_{t+1} = RS4_t \boxplus V12_t \boxplus RS1_{t+1}$ $RS2_{t+1} = RS2_t \boxplus V12_t \boxplus RS4_{t+1}$	

employed in the four AES finalists. Looking into the original Hummingbird cipher, we notice that no cyclic shift has been applied to the plaintext in the initialization process as well as the internal state updating, which explains why the differential pairs found by Saarinen exist. Although more cyclic shifts might be added into the original design of Hummingbird, we believe that two cyclic shifts are enough to thwart the Saarinen's attack, as shown in Section III-A.

IV. Two One-To-One Mappings

To conduct correct encryptions and decryptions, two important one-to-one mappings are used in the original Hummingbird cipher. One is in the initialization, and the other is in the encryption. In this section, we demonstrate that the proposed remedial scheme still keeps two one-to-one mappings. As a result, the remedial Hummingbird cipher can encrypt (decrypt) plaintext (ciphertext) correctly.

Theorem 1: For fixed key, the mapping

$$\text{NONCE} \rightarrow (RS1_0, RS2_0, RS3_0, RS4_0)$$

is one-to-one.

Proof: We only need to prove the mapping

$$(RS1_t, RS2_t, RS3_t, RS4_t) \\ \rightarrow (RS1_{t+1}, RS2_{t+1}, RS3_{t+1}, RS4_{t+1})$$

is one-to-one for $t = -4, -3, -2, -1$.

Suppose that there exist two states $(\widetilde{RS1}_t, \widetilde{RS2}_t, \widetilde{RS3}_t, \widetilde{RS4}_t)$ and $(\widehat{RS1}_t, \widehat{RS2}_t, \widehat{RS3}_t, \widehat{RS4}_t)$ such that $(\widetilde{RS1}_{t+1}, \widetilde{RS2}_{t+1}, \widetilde{RS3}_{t+1}, \widetilde{RS4}_{t+1}) = (\widehat{RS1}_{t+1}, \widehat{RS2}_{t+1}, \widehat{RS3}_{t+1}, \widehat{RS4}_{t+1})$. We need to prove that $(\widetilde{RS1}_t, \widetilde{RS2}_t, \widetilde{RS3}_t, \widetilde{RS4}_t) = (\widehat{RS1}_t, \widehat{RS2}_t, \widehat{RS3}_t, \widehat{RS4}_t)$. Since $\widetilde{V23}_t = E_{k_2}(\widetilde{V12}_t \boxplus \widetilde{RS2}_t) = E_{k_2}(\widetilde{RS2}_{t+1}) = E_{k_2}(\widehat{RS2}_{t+1}) = \widehat{V23}_t$, we have $\widetilde{RS3}_t = \widehat{RS3}_t$. Similarly, we get $\widetilde{RS4}_t = \widehat{RS4}_t$, and $\widetilde{RS1}_t = \widehat{RS1}_t$. Furthermore, because

$$V12_t = E_{k_1}\left((RS1_t \boxplus RS3_t) \boxplus (RS1_t \lll 8)\right),$$

it follows that $\widetilde{V12}_t = \widehat{V12}_t$. Hence $\widetilde{RS2}_t = \widehat{RS2}_t$. ■

Theorem 2: For fixed NONCE and key, the mapping $(PT_1, PT_2, PT_3, PT_4) \rightarrow (RS1_4, RS2_4, RS3_4, RS4_4)$ is one-to-one.

Proof: Let (PT_1, PT_2, PT_3, PT_4) and $(PT_1^{'}, PT_2^{'}, PT_3^{'}, PT_4^{'})$ be two different messages. Let $(RS1_4, RS2_4, RS3_4, RS4_4, \text{LFSR}_4)$ be the state after the encryption of (PT_1, PT_2, PT_3, PT_4), and $(RS1_4^{'}, RS2_4^{'}, RS3_4^{'}, RS4_4^{'}, \text{LFSR}_4^{'})$ be the state after the encryption of $(PT_1^{'}, PT_2^{'}, PT_3^{'}, PT_4^{'})$. Because NONCE and key are fixed, we have $\text{LFSR}_t = \text{LFSR}_t^{'}$ for $t \geq 0$.

Suppose that $(RS1_4, RS2_4, RS3_4, RS4_4) = (RS1_4^{'}, RS2_4^{'}, RS3_4^{'}, RS4_4^{'})$. We will show that $(PT_1, PT_2, PT_3, PT_4) = (PT_1^{'}, PT_2^{'}, PT_3^{'}, PT_4^{'})$ in the following. By the state transition, we have $V34_t = E_{k_3}(RS3_{t+1} \boxminus \text{LFSR}_{t+1})$, and $V23_t = E_{k_2}(RS2_{t+1} \boxminus RS4_{t+1})$. It follows that

$$\begin{aligned}
&(RS1_4, RS2_4, RS3_4, RS4_4, \text{LFSR}_4) \\
&\quad = (RS1_4^{'}, RS2_4^{'}, RS3_4^{'}, RS4_4^{'}, \text{LFSR}_4^{'}) \\
&\qquad\qquad \Rightarrow \\
&\quad RS1_3 = RS1_3^{'}, RS3_3 = RS3_3^{'}, \\
&\quad RS2_3 \boxminus RS4_3 = RS2_3^{'} \boxminus RS4_3^{'} \\
&\qquad\qquad \Rightarrow \\
&\quad RS1_2 = RS1_2^{'}, RS3_2 = RS3_2^{'} \\
&\qquad\qquad \Rightarrow \\
&\quad RS1_1 = RS1_1^{'}.
\end{aligned}$$

For fixed NONCE and key, it holds that $(RS1_0, RS2_0, RS3_0, RS4_0) = (RS1_0^{'}, RS2_0^{'}, RS3_0^{'}, RS4_0^{'})$. By the state transition and $RS1_1 = RS1_1^{'}$, we have

$$\begin{aligned}
V34_0 = V34_0^{'} &\Rightarrow V23_0 \boxplus RS3_0 = V23_0^{'} \boxplus RS3_0^{'} \\
&\Rightarrow V23_0 = V23_0^{'} \\
&\Rightarrow V12_0 \boxplus RS2_0 = V12_0^{'} \boxplus RS2_0^{'} \\
&\Rightarrow V12_0 = V12_0^{'} \\
&\Rightarrow PT_1 \boxplus RS1_0 = PT_1^{'} \boxplus RS1_0^{'} \\
&\Rightarrow PT_1 = PT_1^{'}.
\end{aligned}$$

Hence

$$(RS1_1, RS2_1, RS3_1, RS4_1) = (RS1_1^{'}, RS2_1^{'}, RS3_1^{'}, RS4_1^{'}).$$

Similarly, we have $PT_1 = PT_1^{'}$, $PT_2 = PT_2^{'}$, and $PT_3 = PT_3^{'}$. ■

V. Security Analysis of the Remedial Hummingbird

In this section, we consider the security of our remedial scheme. In addition to attacks investigated in [4], we consider three more attacks, namely, the Saarinen's attack, Boomerang-type attack, and impossible differential attack. For those attacks that have already been considered in [4], the analysis is almost the same. So we omit them here.

Table IV: Three Transformations in a Block Cipher Structure

Transformation	Input Difference	Output Difference
Zero Transformation	$x \in \{0, l_i, m_i, r_i\}$	0
Identity Transformation	$x \in \{0, l_i, m_i, r_i\}$	x
Nonlinear Bijective Transformation	0	0
	l_i	m_j
	m_i	m_j
	r_i	r_j

A. Saarinen's Attack

We randomly selected one NONCE, and constructed another NONCE$'$ by complementing one bit of NONCE. Then we carried out the process of initialization, and obtained two international states using NONCE and NONCE$'$. We computed the Hamming weight between these two international states. We haven't found any Hamming weight smaller than 15 by computer search. We randomly selected two different NONCE and NONCE$'$, and did the same test as above. We haven't found any Hamming weight smaller than 15 by computer search.

We randomly selected one NONCE, and constructed another NONCE$'$ by complementing one bit of NONCE. Then we randomly selected the plaintext. After several rounds of encryption, we record these two international states. We haven't found any low Hamming weight between these two international states. Moreover, we keep the process of initialization the same as that in original Hummingbird, and did the same test as above. We haven't found any low Hamming weight between these two international states, neither.

We collected some IV and ciphertext for which Saarinen's attack works. Then we did three kinds of tests on them.

1) We kept the initialization the same as that in original Hummingbird, but revised the encryption. In this case, Saarinen's attack did not work.
2) We revised the initialization, but kept the encryption the same as that in original Hummingbird. In this case, Saarinen's attack did not work.
3) We revised both the initialization and the encryption. In this case, Saarinen's attack did not work.

B. Boomerang-Type Attack

If the block cipher E can be split into two consecutive stages E_0 and E_1, i.e., $E = E_0 \circ E_1$, and both E_0 and E_1 have differential path, then we may launch differential attack on E [15]. Such attacks are called Boomerang-type attacks. Because of the carry propagation resulting from four rotors, it seems that such attacks can not be applicable to the remedial Hummingbird.

C. Impossible Differential Attack

Impossible differential cryptanalysis exploits differences that are impossible at some intermediate state of the cryptographic algorithm which may leak certain information about the key.

There are four types of differences.

1) **Zero difference.** The difference is zero, and denoted by 0.
2) **Nonzero fixed difference.** The difference is nonzero and fixed, denoted by l.
3) **Nonzero varied difference.** The difference can be any value except zero, and denoted by m.
4) **Varied difference.** The difference can be any value, and denoted by r.

The relationship between input difference and out difference in a block cipher structure is listed in Table IV.

Now we consider the case of the Hamming weight of $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)$ is one. We investigate how the difference propagate. First we consider the case of $\Delta(\mathrm{PT}_0) \neq 0$. Then we consider the case that the Hamming weight of $\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)$ is one.

Theorem 3: Suppose that $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (l_0, 0, 0, 0, 0, 0)$ or $(m_0, 0, 0, 0, 0, 0)$. Then

$$\Delta(\mathrm{CT}_0, \mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1) = (m_1, m_{13}, 2m_{11} \boxplus m_{13}, m_{12}, m_{11} \boxplus m_{13}, 0).$$

Proof: By the encryption in Table III and the difference transform in Table IV, we have

$$\Delta(V12_0) = m_{11}, \Delta(V23_0) = m_{12},$$
$$\Delta(V34_0) = m_{13}, \Delta(CT_0) = m_1.$$

By the internal state updating in Table III, we have

$$\Delta(LFSR_1) = 0,\ \Delta(RS1_1) = m_{13},\ \Delta(RS3_1) = m_{12},$$
$$\Delta(RS4_1) = m_{11} \boxplus m_{13},\ \Delta(RS2_1) = 2m_{11} \boxplus m_{13}.$$

■

Theorem 4: Suppose that $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (l_0, 0, 0, 0, 0, 0)$ or $(m_0, 0, 0, 0, 0, 0)$ and $\Delta(\mathrm{PT}_1) = 0$. Then

$$\Delta(\mathrm{CT}_1, \mathrm{RS1}_2, \mathrm{RS2}_2, \mathrm{RS3}_2, \mathrm{RS4}_2, \mathrm{LFSR}_2) = (r_2, r_{21}, r_{22}, r_{23}, r_{24}, 0).$$

Proof: By Theorem 3, we have

$$\Delta(\mathrm{CT}_0, \mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1) = (m_{14}, m_{13}, 2m_{11} \boxplus m_{13}, m_{12}, m_{11} \boxplus m_{13}, 0).$$

By the encryption in Table III and the difference transform in Table IV, we have

$$\Delta(V12_1) = m_{21}, \Delta(V23_1) = r_{22}^{'},$$
$$\Delta(V34_1) = r_{23}^{'}, \Delta(CT_1) = r_2.$$

By the internal state updating in Table III, we have

$$\Delta(LFSR_2) = 0,\ \Delta(RS1_2) = r_{21},\ \Delta(RS3_2) = r_{23},$$
$$\Delta(RS4_2) = r_{24},\ \Delta(RS2_2) = r_{22}.$$

■

Theorem 5: Suppose that $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (0, l_0, 0, 0, 0, 0)$ or $(0, m_0, 0, 0, 0, 0)$ and $\Delta(\mathrm{PT}_1) = l_1$ or m_1. Then

$$\Delta(\mathrm{CT}_1, \mathrm{RS1}_2, \mathrm{RS2}_2, \mathrm{RS3}_2, \mathrm{RS4}_2, \mathrm{LFSR}_2) = (r_2, r_{21}, r_{22}, r_{23}, r_{24}, 0).$$

Proof: The proof is the same as that of Theorem 4. So we omit it. ■

By Theorems 3, 4, and 5, in the case of $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (l_0, 0, 0, 0, 0, 0)$ or $(m_0, 0, 0, 0, 0, 0)$, after two rounds of encryption, the output difference of $(\mathrm{CT}, \mathrm{RS1}, \mathrm{RS2}, \mathrm{RS3}, \mathrm{RS4})$ could be any value. Hence it is hard to find any valuable impossible differential pairs after two rounds of encryption.

Now we consider the case that the Hamming weight of $\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)$ is one. In this case, we only need to consider $\Delta(\mathrm{RS1}_0) \neq 0$ because other cases are the same.

Theorem 6: Suppose that $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (0, l_0, 0, 0, 0, 0)$ or $(0, m_0, 0, 0, 0, 0)$. Then

$$\Delta(\mathrm{CT}_0, \mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1) = (m_1, m_{13}, 2m_{11} \boxplus m_{13}, m_{12}, m_{11} \boxplus m_{13}, 0).$$

Proof: By the encryption in Table III and the difference transform in Table IV, we have

$$\Delta(V12_0) = m_{11}, \Delta(V23_0) = m_{12},$$
$$\Delta(V34_0) = m_{13}, \Delta(CT_0) = m_1.$$

By the internal state updating in Table III, we have

$$\Delta(LFSR_1) = 0,\ \Delta(RS1_1) = m_{13},\ \Delta(RS3_1) = m_{12},$$
$$\Delta(RS4_1) = m_{11} \boxplus m_{13},\ \Delta(RS2_1) = 2m_{11} \boxplus m_{13}.$$

■

Theorem 7: Suppose that $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (0, l_0, 0, 0, 0, 0)$ or $(0, m_0, 0, 0, 0, 0)$, and $\Delta(\mathrm{PT}_1) = 0$, l_1 or m_1. Then

$$\Delta(\mathrm{CT}_1, \mathrm{RS1}_2, \mathrm{RS2}_2, \mathrm{RS3}_2, \mathrm{RS4}_2, \mathrm{LFSR}_2) = (r_2, r_{21}, r_{22}, r_{23}, r_{24}, 0).$$

Proof: The proof is the same as that of Theorem 4. So we omit it. ■

By Theorems 6 and 7, in the case of $\Delta(\mathrm{PT}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (0, l_0, 0, 0, 0, 0)$ or $(0, m_0, 0, 0, 0, 0)$, after two rounds of encryption, the output difference of $(\mathrm{CT}, \mathrm{RS1}, \mathrm{RS2}, \mathrm{RS3}, \mathrm{RS4})$ could be any value. Hence it is hard to find any valuable impossible differential pairs after two rounds of encryption. Moreover, it seems hard to find two different NONCE such that the Hamming weight of $\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0)$ is one.

By the analysis above, we may conclude that it is hard to lunch any impossible differential attack on the remedial Hummingbird.

VI. Implementation of Remedial Hummingbird

In this section, we report software implementation results of the remedial Hummingbird cipher on two low-cost microcontrollers Atmega128L and MSP430. In order to compare the performance of the remedial and original Hummingbird cipher, we use the same platforms as those in [4].

A. Low-Cost Microcontrollers

We choose a 8-bit microcontroller Atmega128L and a 16-bit microcontroller MSP430F1611 as the target platforms, which are the processors equipped in wireless sensor nodes MICAz and TELOSB/TMote Sky. The Atmega128L from Atmel has a RISC architecture and comes with 128 KBytes flash, 4 KBytes EEPROM and 8 KBytes SRAM. Moreover, the Atmega128L can run from 0 to 8 MHz and the power supplies can go from 2.7 to 5.5 V.

The MSP430F1611 from Texas Instrument features a traditional von-Neumann architecture with 48 KBytes flash and 10 KBytes RAM. It can run from 0 to 8 MHz and the power supplies can go from 1.8 to 3.6 V.

B. Development Tools

For the 8-bit microcontroller Atmega128L, we use the integrated development environment AVR Studio 4.17 from Atmel as an editor and a simulator. Additionally, we use the open-source WinAVR-20090313 toolkit to compile and link the source code. For the 16-bit microcontroller MSP430F1611, we utilize CrossWorks for MSP430 Version 2 from Rowley Associates [10] to implement and simulate the Hummingbird cipher.

Table V: Memory Consumption and Cycle Count Comparison (Size Optimized Implementation)

Cipher	8-bit/16-bit Microcontroller	Flash Size [bytes]	Hex Code Size [Kbytes]	Init. [cycles]	Enc. [cycles/ block]	Dec. [cycles/ block]
Hummingbird [4]	ATmega128L	1,308	3.68	14,735	3,664	3,868
	MSP430F1611	1,064	2.95	9,667	2,414	2,650
Remedial Hummingbird	ATmega128L	1,315	3.68	14,746	3,671	3,875
	MSP430F1611	1,072	2.95	9,674	2,420	2,656

Table VI: Memory Consumption and Cycle Count Comparison (Speed Optimized Implementation)

Cipher	8-bit/16-bit Microcontroller	Flash Size [bytes]	Hex Code Size [Kbytes]	Init. [cycles]	Enc. [cycles/ block]	Dec. [cycles/ block]
Hummingbird [4]	ATmega128L	10,918	30.5	8,182	1,399	1,635
	MSP430F1611	1,360	3.76	4,824	1,220	1,461
Remedial Hummingbird	ATmega128L	10,926	30.5	8,194	1,405	1,642
	MSP430F1611	1,367	3.76	4,837	1,226	1,466

C. Size Optimized Implementation

For the size optimized implementation, the S-box is implemented as a byte array with 16 elements and the S-box look-up of a 16-bit block is performed sequentially with 4 bits being processed each time. Note that for the remedial Hummingbird we need to rotate a 16-bit block by 8 positions to the left. Fortunately, this operation is almost free on Atmega128L and MSP430F1611, since the instruction set of both microcontrollers does allow a one clock cycle nibble (or byte) swap within a register. Table V compares the memory consumption and cycle count between the original and remedial Hummingbird cipher on 8- and 16-bit microcontrollers for the size optimized implementation. It is not difficult to find that the code size of the remedial Hummingbird is only about 0.6% larger than that of the original design on the 8- and 16-bit microcontrollers. Moreover, the throughput of the remedial version is around 0.2% slower than that of the original Hummingbird on the target 8- and 16-bit platforms. Hence, for the size optimized implementation, our remedial scheme is quite lightweight.

D. Speed Optimized Implementation

For the speed optimized implementation, we combine two 4×4 S-boxes to form a larger 8×8 S-box, which accepts a byte as an input and cost an extra 512 bytes of data memory. Again, the overhead of the rotation operations in the remedial Hummingbird is negligible on both microcontrollers Atmega128L and MSP430F1611. Table VI compares the memory consumption and cycle count between the original and remedial Hummingbird cipher on 8- and 16-bit microcontrollers for the speed optimized implementation. One can find that the code size of the remedial Hummingbird is only about 0.07% and 0.5% larger than that of the original design on the 8- and 16-bit microcontrollers, respectively. In addition, the throughput of the remedial version is around 0.4% slower than that of the original Hummingbird on the target 8- and 16-bit platforms. Therefore, for the speed optimized implementation, the overhead introduced by our remedial scheme is negligible.

VII. Concluding Remarks

In this paper, we propose a simple and lightweight tweak for the original Hummingbird cipher. Our scheme is quite efficient both in software and hardware since it only involves three cyclic shifts. Using this simple approach, we are able to keep the compact design as well as enhance the security of the original Hummingbird cipher. The cryptanalysis shows that the remedial Hummingbird is resistant to all known attacks. In particular, our remedial scheme does not affect the performance of the original Hummingbird cipher.

Acknowledgement

This work was supported in part by NSERC Discovery and DAS Grant, and the National Natural Science Foundation of China (60802029, 60833008).

References

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher", *The 9th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007*, LNCS 4727, P. Paillier and I. Verbauwhede (eds.), Berlin, Germany: Springer-Verlag, pp. 450-466, 2007.

[2] C. De Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN-A Family of Small and Efficient Hardware-Oriented Block Ciphers", *The 11th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2009*, LNCS 5747, C. Clavier and K. Gaj (eds.), Berlin, Germany: Springer-Verlag, pp. 272-288, 2009.

[3] C. De Cannière and B. Preneel, "Trivium-A Stream Cipher Construction Inspired by Block Cipher Design Principles", ECRYPT Stream Cipher, Available at http://www.ecrypt.eu.org/stream/papersdir/2006/021.pdf, 2005.

[4] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices, The 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC'2010), Financial Cryptography 2010 Workshops, LNCS 6054, pp. 3-18, 2010.

[5] D. Engels, M. O. Saarinen, and E. M. Smith, The Hummingbird-2 Lightweight Authenticated Encryption Algorithm, RFID Security 2011

[6] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations", *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 522-533, 2007.

[7] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm", *The 6th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2004*, LNCS 3156, M. Joye and J.-J. Quisquater (eds.), Berlin, Germany: Springer-Verlag, pp. 357-370, 2004.

[8] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. D. Hämäläinen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core", *The 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools - DSD 2006*, pp. 577-583, IEEE Computer Society, 2006.

[9] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, and S. Chee, "HIGHT: A New Block Cipher Suitable for Low-Resource Device", *The 8th International Workshop on Cryptographic Hardware and Embedded Systems-CHES 2006*, LNCS 4249, L. Goubin and M. Matsui (eds.), Berlin, Germany: Springer-Verlag, pp. 46-59, 2006.

[10] Rowley Associates. CrossWorks for MSP430. Available at http://www.rowley.co.uk/msp430/index.htm.

[11] X. Lai, "Higher Order Derivatives and Differential Cryptanalysis", *Proceedings of Symposium on Communication, Coding and Cryptography*, in honor of James L. Massey on the occasion of his 60'th birthday, 1994.

[12] G. Leander, C. Paar, A. Poschmann, and K. Schramm, "New Lightweight DES Variants", *The 14th Annual Fast Software Encryption Workshop - FSE 2007*, LNCS 4593, A. Biryukov (ed.), Berlin, Germany: Springer-Verlag, pp. 196-210, 2007.

[13] M. O. Saarinen, Cryptanalysis of Hummingbird-1, FSE 2011.

[14] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, "SEA: A Scalable Encryption Algorithm for Small Embedded Applications", *The 7th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications-CARDIS 2006*, LNCS 3928, J. Domingo-Ferrer, J. Posegga, and D. Schreckling (eds.), Berlin, Germany: Springer-Verlag, pp. 222-236, 2006.

[15] D. Wagner, L. Knudsen, Ed., "The boomerang attack," in Proc. Fast Software Encryption (FSE'99), LCNS Series, 1999, vol. 1636, pp. 156-170.