Cryptanalysis of Hummingbird-1

Markku-Juhani O. Saarinen

REVERE SECURITY
4500 Westgrove Drive, Suite 335, Addison, TX 75001, USA.
mjos@reveresecurity.com

Abstract. Hummingbird-1 is a lightweight encryption and message authentication primitive published in RISC '09 and WLC '10. Hummingbird-1 utilizes a 256-bit secret key and a 64-bit IV. We report a chosen-IV, chosen-message attack that can recover the full secret key with a few million chosen messages processed under two related IVs. The attack requires at most 2^{64} off-line computational effort. The attack has been implemented and demonstrated to work against a real-life implementation of Hummingbird-1. By attacking the differentially weak E component, the overall attack complexity can be reduced by a significant factor. Our cryptanalysis is based on a differential divide-and-conquer method with some novel techniques that are uniquely applicable to ciphers of this type.

Keywords: Hummingbird cipher, constrained devices, lightweight cryptography, stream cipher cryptanalysis.

1 Introduction

The advent of small-form wireless control and communication devices, sensors and authentication tags is affecting commercial, military and domestic security engineering in ways which were almost unimaginable only 10–20 years ago.

An important selection criterion when choosing cryptographic security components for such extremely constrained devices is obviously *cost*, which directly relates to the complexity of hardware and software implementation of the component and its computational efficiency. These lightweight cryptographic solutions must also meet stringent security requirements as they are often critical links in the overall "chain of security" – user authentication with a RFID token, a private conversation using a wireless hands-free set and encryption of key presses on a wireless keyboard are some examples.

Hummingbird-1 [2, 5] is a recent cryptographic algorithm proposal for RFID tags and other constrained devices. It is covered by several pending patents and is being commercially marketed by the Revere Security [7]. Revere has invested into Hummingbird's cryptographic security assurance before its publication by contracting ISSI, a private consultancy employing some ex-NSA staff [6] and

members of U. Waterloo CACR [4]. After this work was originally done, an improved version, Hummingbird-2, has been developed.

In the present report we show that the published version of Hummingbird-1 is suspectible to a chosen-IV, chosen message attack that has an attack complexity of significantly less than 2^{64} operations and data complexity of only few megabytes, the entire 256-bit secret key can be recovered. The attack has been implemented and demonstrated to work against a validated implementation of Hummingbird-1.

This paper is structured as follows. In Section 2 we give a description of Hummingbird-1 and make a key observations about its initialization procedure. In Section 3 we build an attack, step by step, that breaks Hummingbird-1. Section 4 contains a discussion about the implementation and implications of the attack, followed by conclusions in Section 5.

2 Description of Hummingbird-1

Hummingbird-1 [2, 4, 5] is an encryption and message authentication primitive that has a 256-bit secret key, uses a 64-bit IV (nonce) and optionally produces a 64-bit authenticator for the message. Hummingbird-1 is similar to ciphers such as Helix [3] and Phelix [10] in that it is a word-based stream cipher that can also be used for authentication. We have not analyzed the security of the proposed authentication functionality and it will not be discussed in this paper.

2.1 Notation and Parameters

The 256-bit secret key K is indexed as a vector of four 64-bit subkeys $K^{(i)}$. Each one of the 64-bit subkeys further consists of 16-bit words $K_j^{(i)}$ as follows:

$$\begin{aligned} K &= (K^{(1)}, K^{(2)}, K^{(3)}, K^{(4)}) \\ K^{(1)} &= (K_1^{(1)}, K_2^{(1)}, K_3^{(1)}, K_4^{(1)}) \\ K^{(2)} &= (K_1^{(2)}, K_2^{(2)}, K_3^{(2)}, K_4^{(2)}) \\ K^{(3)} &= (K_1^{(3)}, K_2^{(3)}, K_3^{(3)}, K_4^{(3)}) \\ K^{(4)} &= (K_1^{(4)}, K_2^{(4)}, K_3^{(4)}, K_4^{(4)}). \end{aligned}$$

The 80-bit internal state of Hummingbird-1 at round t consists of four 16-bit registers $\mathrm{RS1}_t$, $\mathrm{RS2}_t$, $\mathrm{RS3}_t$, $\mathrm{RS4}_t$ and the independent shift register LFSR_t.

When considering differential attacks, we denote by Δ the additive difference between two values. In our differential analysis we will be working on

pairs of related instances of Hummingbird-1 which share the same secret key K. The state of the first and second instance at round t is written as

$$(\mathrm{RS1_t}, \mathrm{RS2_t}, \mathrm{RS3_t}, \mathrm{RS4_t}, \mathrm{LFSR_t})$$
$$\textit{and}$$
$$(\mathrm{RS1'_t}, \mathrm{RS2'_t}, \mathrm{RS3'_t}, \mathrm{RS4'_t}, \mathrm{LFSR'_t}).$$

The additive state difference $\Delta(\mathrm{RS1_t}, \mathrm{RS2_t}, \mathrm{RS3_t}, \mathrm{RS4_t}, \mathrm{LFSR_t})$ is

$$(\mathrm{RS1_t} \boxminus \mathrm{RS1'_t}, \mathrm{RS2_t} \boxminus \mathrm{RS2'_t}, \mathrm{RS3_t} \boxminus \mathrm{RS3'_t}, \mathrm{RS4_t} \boxminus \mathrm{RS4'_t}, \mathrm{LFSR_t} \boxminus \mathrm{LFSR'_t}).$$

Here $\boxminus$ denotes two's complement subtraction modulo 2^{16}. We will also write $\Delta P_i = P_i \boxminus P'_i$ and $\Delta C_i = C_i \boxminus C'_i$ to denote plaintext and ciphertext difference at message word i. Numerical values for differentials are in hexadecimal notation.

2.2 The 16-bit permutation *E*

The 16-bit permutation component $E(x, K^{(i)})$ consists of five invocations of four S-Boxes, interleaved with a mixing of a 16-bit subkey and a linear transform L. Figure 1 illustrates the operation of the the block cipher E.

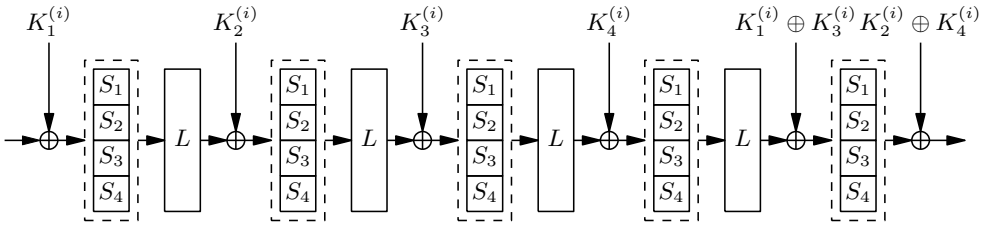

Fig. 1. The "E box" is a 16-bit permutation with a 64-bit key. L is a 16-bit linear transform $L(x) = x \oplus (x \lll 6) \oplus (x \lll 10)$.

Four permutations of values 0..15 are used as the four-bit S-boxes $S_1(x)$, $S_2(x)$, $S_3(x)$ and $S_4(x)$. We have discovered that at least two variants of the four S-Boxes exist, one set being described in [5] and an another set in ISSI's analysis [6]. The second set of S-Boxes is equivalent to S4-S7 of Serpent-1 [1] and is compatible with test vectors provided by Revere Security [9]. Tables 1 and 2 give both S-Boxes in full.

Any particular choice of S-Boxes does not affect the main cryptanalysis presented in this paper. In fact, the attack is applicable regardless of what type of E function is used as long as it is keyed with only 64 bits. Hence the particular

Table 1. Hummingbird S-Boxes as reported in [5].

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1(x)$	8	6	5	15	1	12	10	9	14	11	2	4	7	0	13	3
$S_2(x)$	0	7	14	1	5	11	8	2	3	10	13	6	15	12	4	9
$S_3(x)$	2	14	15	5	12	1	9	10	11	4	6	8	0	7	3	13
$S_4(x)$	0	7	3	4	12	1	10	15	13	14	6	11	2	8	9	5

Table 2. The actual Hummingbird S-Boxes in an implementation obtained from its authors [9].

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_1(x)$	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
$S_2(x)$	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
$S_3(x)$	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
$S_4(x)$	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

choice of the number of rounds, S-Boxes and the linear transformation has little effect to the overall security of the cipher.

We define the linear transform $L(x)$ as

$$L(x) = x \oplus (x \lll 6) \oplus (x \lll 10), \tag{1}$$

where $\lll$ is a left circular shift operator. By $S(x)$ we denote the application of the four S-boxes in parallel on the four nibbles of $x = x_0 \mid x_1 \mid x_2 \mid x_3$:

$$S(x) = S_1(x_0) \mid S_2(x_1) \mid S_3(x_2) \mid S_4(x_3). \tag{2}$$

The complete 16-bit keyed permutation $E(x, K^{(i)})$ is described by:

$$\begin{aligned} u_0 &= x \oplus K_1^{(i)} \\ u_1 &= L(S(u_0)) \oplus K_2^{(i)} \\ u_2 &= L(S(u_1)) \oplus K_3^{(i)} \\ u_3 &= L(S(u_2)) \oplus K_4^{(i)} \\ u_4 &= L(S(u_3)) \oplus K_1^{(i)} \oplus K_3^{(i)} \\ E(x, K^{(i)}) &= S(u_4) \oplus K_2^{(i)} \oplus K_4^{(i)}. \end{aligned}$$

2.3 Initialization

To set up Hummingbird-1, we first load the 64-bit IV value to the state registers:

$$(\mathrm{RS1}_{-4}, \mathrm{RS2}_{-4}, \mathrm{RS3}_{-4}, \mathrm{RS4}_{-4}) = (\mathrm{IV}_1, \mathrm{IV}_2, \mathrm{IV}_3, \mathrm{IV}_4). \tag{3}$$

After this, four rounds of special stepping is performed for $t = -4, -3, -2, -1$:

$$\begin{aligned}
v_{12_t} &= E((\mathrm{RS1_t} \boxplus \mathrm{RS3_t}) \boxplus \mathrm{RS1_t},\ K^{(1)}) \\
v_{23_t} &= E(v_{12_t} \boxplus \mathrm{RS2}_t,\ K^{(2)}) \\
v_{34_t} &= E(v_{23_t} \boxplus \mathrm{RS3}_t,\ K^{(3)}) \\
tv_t &= E(v_{34_t} \boxplus \mathrm{RS4}_t,\ K^{(4)}) \\
\mathrm{RS1}_{t+1} &= \mathrm{RS1}_t \boxplus tv_t \\
\mathrm{RS2}_{t+1} &= \mathrm{RS2}_t \boxplus v_{12_t} \\
\mathrm{RS3}_{t+1} &= \mathrm{RS3}_t \boxplus v_{23_t} \\
\mathrm{RS4}_{t+1} &= \mathrm{RS4}_t \boxplus v_{34_t}.
\end{aligned}$$

Here the $\boxplus$ operator denotes addition modulo 2^{16}. After the final round, we set the bit 12 (or the 13th bit as it is expressed in the specification) in the tv temporary variable and assign that as the LFSR value:

$$\mathrm{LFSR}_0 = tv_3 \ \vee\ \texttt{1000}. \tag{4}$$

Therefore the 80-bit state after the initialization phase consists of the five words

$$(\mathrm{RS1}_0\ \mathrm{RS2}_0\ \mathrm{RS3}_0\ \mathrm{RS4}_0\ \mathrm{LFSR}_0). \tag{5}$$

Observation 1 *The Hummingbird-1 initialization function has a high-bit XOR differential that holds with probability 1:*

$$\begin{gathered}
\Delta(\mathrm{IV}_1, \mathrm{IV}_2, \mathrm{IV}_3, \mathrm{IV}_4) = (\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}) \\
\Downarrow \\
\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}).
\end{gathered}$$

2.4 The encryption function

Each Hummingbird-1 encryption round accepts a 16-bit plaintext word P_i to produce a ciphertext word C_i. Figure 2 illustrates one round of Hummingbird encryption.

For $t \geq 0$ (after initialization) we have

$$
\begin{aligned}
v_{12_t} &= E(P_t \boxplus \mathrm{RS1}_t,\ K^{(1)}) \\
v_{23_t} &= E(v_{12_t} \boxplus \mathrm{RS2}_t,\ K^{(2)}) \\
v_{34_t} &= E(v_{23_t} \boxplus \mathrm{RS3}_t,\ K^{(3)}) \\
C_t &= E(v_{34_t} \boxplus \mathrm{RS4}_t,\ K^{(4)}) \\
\mathrm{LFSR}_{t+1} &= \texttt{STEP}(\mathrm{LFSR}_t) \\
\mathrm{RS1}_{t+1} &= \mathrm{RS1}_t \boxplus v_{34_t} \\
\mathrm{RS4}_{t+1} &= \mathrm{RS4}_t \boxplus v_{12_t} \boxplus \mathrm{RS1}_{t+1} \\
\mathrm{RS2}_{t+1} &= \mathrm{RS2}_t \boxplus v_{12_t} \boxplus \mathrm{RS4}_{t+1} \\
\mathrm{RS3}_{t+1} &= \mathrm{RS3}_t \boxplus v_{23_t} \boxplus \mathrm{LFSR}_{t+1}.
\end{aligned}
$$

The Hummingbird LFSR has been implemented in a slightly unusual right-cyclical fashion, which is best desribed in the C language:

```
lfsr = (lfsr >> 1) ^ (-(lfsr & 1) & 0xCA44);
```

THe LFSR operates independently from the other registers as there is no feedback from them or the plaintext to it. The particular LFSR selection or its operation does not affect on our attack in any way.

In this paper we will denote by $\mathrm{HB}(\mathrm{IV}, v) = z$ a query for encryption of vector v with the given IV value. Conversely, $\mathrm{HB}^{-1}(\mathrm{IV}, z) = v$ is a decryption query. Since Hummingbird is attacked in a "black box" fashion in this chosen-IV, chosen message attack, we don't include the unknown secret key into the notation of encryption/decryption queries.

3 Building an attack

Our attack proceeds in several stages, first attacking the initialization function and then each 64-bit subkey individually, proceeding from the "outer layer" subkeys $K^{(1)}$ and $K^{(4)}$ towards the "inner layer" subkeys $K^{(3)}$ and $K^{(2)}$. Each stage of the attack is constructed differently.

The line of attack described in this paper is just one of many. A small modification of the algorithm or adjustment of the usage model may lead to wholly different security properties.

We will first describe a very simple chosen-IV distinguisher for Hummingbird, which will be a part of subsequent stages of the attack. For any two nonces (IVs) that have a difference in the most significant bit (MSB) of the first word,

we can simply flip the MSB of the plaintext word and the ciphertext words will match.

Observation 2 *There is a Chosen-IV distinguisher for Hummingbird that works with probability $P = 65535/65536$ and has data complexity of 1 word. One can use the high-bit differential of Observation 1 and the following differential for the first round:*

$$\begin{gathered}\Delta(\mathrm{P}_0, \mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (\mathtt{8000}, \mathtt{8000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}) \\ \Updownarrow \\ \Delta(\mathrm{C}_0, \mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1) = (\mathtt{0000}, \mathtt{8000}, \mathtt{8000}, \mathtt{0000}, \mathtt{8000}, \mathtt{0000})\end{gathered}$$

The differential works both ways (chosen plaintext and chosen ciphertext). If we decipher the same word, say, $\mathtt{0000}$ under the two different nonces that are related by only having a MSB difference in the first word, there will be a high-bit difference in the first word of the corresponding plaintext. This constitutes the distinguisher.

3.1 An iterative differential

Observation 3 *There is a one-round iterated differential that works if a collision occurs inside the cipher as follows:*

$$\begin{gathered}\Delta v_{12_t} = \mathtt{8000}\ ,\ \Delta v_{23_t} = \mathtt{0000}\ ,\ \Delta v_{34_t} = \mathtt{0000} \\ \Delta(\mathrm{RS1_t}, \mathrm{RS2_t}, \mathrm{RS3_t}, \mathrm{RS4_t}, \mathrm{LFSR_t}) = (\mathtt{8000}, \mathtt{8000}, \mathtt{0000}, \mathtt{8000}, \mathtt{0000}) \\ \Updownarrow \\ \Delta(\mathrm{RS1_{t+1}}, \cdots \mathrm{RS4_{t+1}}, \mathrm{LFSR_{t+1}}) = (\mathtt{8000}, \mathtt{8000}, \mathtt{0000}, \mathtt{8000}, \mathtt{0000}).\end{gathered}$$

The initial condition for $t = 5$ can be satisfied using the initialization and first-round encryption differentials given in Observations 1 and 2.

To verify Observation 3, one may find it useful to trace the high-bit differentials (and their internal cancellation) in Figure 2 with a highlighting pen. We note that each one of the conditions $\Delta v_{12} = \mathtt{8000}$, $\Delta v_{23} = \mathtt{0000}$, $\Delta v_{34} = \mathtt{0000}$ implies the other two if the input (or output) state differential holds.

From the algorithm description we see that the internal value v_{34} satisfies

$$\Delta v_{34} = \Delta E^{-1}(C_i, K^{(4)}) \boxminus \Delta \mathrm{RS4}_t. \tag{6}$$

For the condition $\Delta v_{34} = \mathtt{0000}$ to be satisfied and the iterative differential to work it suffices to find a pair of ciphertext words $C_i = a$ and $C_i' = b$ such that

$$E^{-1}(a, K^{(4)}) \boxminus E^{-1}(b, K^{(4)}) = \mathtt{8000}. \tag{7}$$

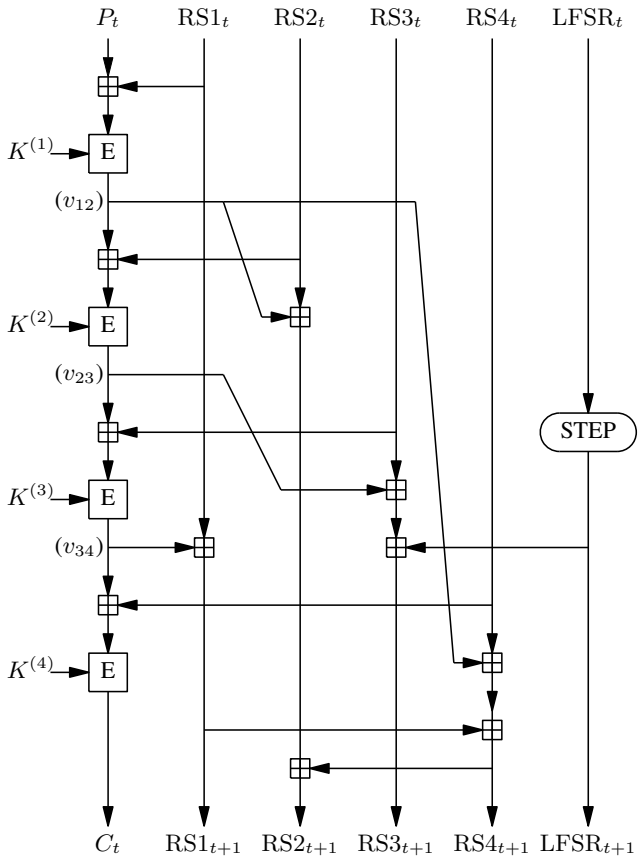

Fig. 2. Encrypting a single 16-bit word P_t to produce a ciphertext word C_t with Hummingbird. After initialization $t \geq 0$.

The first stage of our overall attack is based on chosen-ciphertext queries of the type

$$P = \mathrm{HB}^{-1}\big((\mathtt{0000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}), (x, a, a, \ldots, a)\big) \tag{8}$$

$$P' = \mathrm{HB}^{-1}\big((\mathtt{8000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}), (x, b, b, \ldots, b)\big). \tag{9}$$

If a and b are related in as in Equation 7, the iterative differential of Observation 3 will hold for all $t \geq 1$ in Equations 8 and 9 above. The initial x word is arbitrary; the differential will work as long as $C_0 = C'_0$. This will result in $\Delta P_0 = \mathtt{8000}$.

For our attack *any* pair (a, b) satisfying Equation 7 will suffice. It is easy to see that there are 2^{16} such pairs. By the birthday paradox, by decrypting about $\sqrt{2^{16}} = 2^8$ vectors of the form given in Equations 8 and 9, we should have found one such pair. How to distinguish it from the other pairs ?

From the algorithm definition we can see that if the iterative differential holds, then $\Delta v_{12} = \texttt{8000}$, $\Delta \mathrm{RS1}_t = \texttt{8000}$ and the plaintext words satisfy for all $t > 0$

$$\Delta P_t = E^{-1}(v_{12_t}, K^{(1)}) \boxminus \big(E^{-1}(v_{12_t} \boxplus \texttt{8000}, K^{(1)}) \boxplus \texttt{8000}\big). \tag{10}$$

To analyze this condition, we may consider a random bijective function F on n-bit values and the behavior of the differential

$$\Delta F(x) = F(x) \boxminus F(x \boxplus c) \tag{11}$$

where c is some nonzero constant and x takes on all values $0 \leq x \leq 2^n$. It is easy to show that the behavior of $\Delta F(x)$ resembles that of a random function in that its range can be expected to be $2^n(1 - e^{-1}) \approx 0.6321 \times 2^n$ rather than 2^n. For ease of exposition we will be considering the absolute delta value

$$\mathrm{abs}(\Delta x) = x - x' \text{ if } x > x' \text{ and } x' - x \text{ otherwise.} \tag{12}$$

ΔP_i in Equation 10 has similarly limited range if the iterative differential holds. If the differential does not hold, ΔP_i may have any value. We use this feature to test for the right pair; if the iterative differential holds for some ciphertext words x and y, the range of $\mathrm{abs}(\Delta P_i)$ values will be close to $2^{15}(1 - e^{-1}) \approx 20713$ rather than $2^{15} = 32768$. The procedure is given by Algorithm 1. The complexity of Algorithm 1 is less than 2^{30} operations and data complexity is equivalent to decrypting eight megabytes of data. The choice of looping through 2^9 values of i and using 2^{12} words of data in Algorithm 1 may not be optimal, but will be sufficient for actually finding a correct pair with a reasonable probability.

In practice the algorithm finds a right pair in a few seconds. The current implementation also rechecks the pair with longer decryptions and performs a retry if the count of the absolute range is larger than 25000.

3.2 Attacking $K^{(1)}$

Our first target is to attack the 64-bit subkey $K^{(1)}$. With the (a, b) ciphertext word pair obtained with Algorithm 1, and further chosen-message queries, we will extract the entire range $\mathbf{S_1}$ of the function δ_1 defined by

$$\delta_1(x) = \mathrm{abs}\big(E^{-1}(x, K^{(1)}) \boxminus E^{-1}(x \boxplus \texttt{8000}, K^{(1)})\big). \tag{13}$$

The expected size of $\mathbf{S_1}$ is $2^{15}(1 - \frac{1}{e}) \approx 20713$ elements. To compute $\mathbf{S}_1$, we decrypt two at least megaword-long vectors consisting of the a and b words:

$$P = \mathrm{HB}^{-1}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (\texttt{0000}, a, a, \ldots, a)\big) \tag{14}$$

$$P' = \mathrm{HB}^{-1}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (\texttt{0000}, b, b, \ldots, b)\big). \tag{15}$$

Algorithm 1 Probabilistically find a pair (a, b) satisfying Equation 7 as discussed in Section 3.1.

for $i = 1, 2, \ldots, 2^9$ **do**
  $v = (\texttt{0000}, i, i, \ldots, i)$, a vector of 2^{12} words.
  $x[i][1..2^{12}] = \mathrm{HB}^{-1}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), v\big)$.
  $y[i][1..2^{12}] = \mathrm{HB}^{-1}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), v\big)$.
end for
$a = 0, b = 0, m = 2^{15}$.
for $i = 0, 1, \ldots, 2^9$ **do**
  for $j = 0, 1, \ldots, 2^9$ **do**
    Count the number of different words n in the set defined by $\mathrm{abs}(x[i][k] \boxminus y[j][k])$.
    if $n < m$ **then**
      $a = i, b = j, m = n$.
    end if
  end for
end for

Since the iterative differential of Observation 3 holds for all rounds $t > 1$ but the internal state is otherwise evolving and can be modelled as random, each difference in corresponding plaintext words can be simply inserted into the set $\mathbf{S}_1$:

$$\mathrm{abs}(P_i \boxminus P'_i \oplus \texttt{8000}) \in \mathbf{S}_1 \;\text{ when } i > 0. \tag{16}$$

Note that the completeness of $\mathbf{S}_1$ is highly dependent on the length of the ciphertext vectors; one million words will yield a complete set with high certainty, but one hundred thousand words with very low certainty.

Armed with the set $\mathbf{S}_1$, we can perform an off-line attack on the first subkey. To test a subkey candidate $K^{(1)}$ it suffices to loop through values x doing the membership test $\delta(x) \in \mathbf{S_i}$, as indicated by Equation 13. For a false key candidate the membership test will fail with probability of roughly 63.2%. Most key candidates can be discarded after two trials. Since each membership test (for x) is independent, the certainty that a correct key has not been found after n successful trials $(1 - \frac{1}{e})^n$. $n = 97$ trials gives a 2^{-64} uncertainty. Our implementation performs all $n = 2^{15}$ trials, as the performance penalty is negligible due to the early exit strategy.

3.3 Attacking $K^{(4)}$

The next subkey to be attacked after $K^{(1)}$ is the last to be used during encryption, $K^{(4)}$. There are several ways to do this efficiently. We will describe the one we implemented.

We use our knowledge of $K^{(1)}$ and the differential of Observation 3 to find more ciphertext pairs (C_i, C'_i) that have a $\Delta = \texttt{8000}$ input difference to the last

invocation of E. This implies that these ciphertext pairs satisfy the equation

$$E^{-1}(C_i, K^{(4)}) = E^{-1}(C'_i, K^{(4)}) \boxplus \texttt{8000}. \tag{17}$$

If at least four such ciphertext word pairs are available, we may do a conclusive exhaustive search over the entire 64-bit subkey $K^{(4)}$ by using Equation 17 as a test.

We will first obtain a known value for $\mathrm{RS1}_1$. We use the known (a, b) pair from Section 3.1 and Algorithm 1 and decrypt a set of two-word vectors for few running values of initial ciphertext word x:

$$P = \mathrm{HB}^{-1}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (\texttt{x}, \texttt{a})\big) \tag{18}$$

$$P' = \mathrm{HB}^{-1}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (\texttt{x}, \texttt{b})\big). \tag{19}$$

For each decryption $\Delta P_0 = \texttt{8000}$ as indicated by Observation 2. The second plaintext word will satisfy

$$E(P_1 \boxplus \mathrm{RS1}_1, K^{(1)}) = E(P'_1 \boxplus \mathrm{RS1}_1 \boxplus \texttt{8000}, K^{(1)}) \boxplus \texttt{8000} \tag{20}$$

since $\Delta\mathrm{RS1}_1 = \texttt{8000}$. There usually is only one or at most few possible values of $\mathrm{RS1}_1$ that satisfy Equation 20. Such an unique value is found for some x by simply searching through all possible 2^{16} values of $\mathrm{RS1}_1$ using the knowledge of the subkey $K^{(1)}$ (that was obtained in the previous section). This gives us information about the internal state of the cipher after one encryption round.

Let $y = P_0 = P'_0 \boxplus \texttt{8000}$ for some pair of related decryptions described in Equations 18 and 19 such that an unique value for $\mathrm{RS1}_1$ can be established. To create pairs suitable for testing by Equation 17 we again turn into a chosen-plaintext attack and encrypt few vectors for a chosen running value of v_{12_1}:

$$\begin{aligned} C =& \mathrm{HB}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (y,\ E^{-1}(v_{12_1}, K^{(1)}) \boxminus \mathrm{RS1}_1)\big) \\ C' =& \mathrm{HB}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), \\ & (y \boxplus \texttt{8000}, E^{-1}(v_{12_1} \boxplus \texttt{8000}, K^{(1)}) \boxminus \mathrm{RS1}_1 \boxplus \texttt{8000})\big). \end{aligned}$$

The ciphertext words C_1 and C'_1 can be used for exhaustive search of the 64-bit subkey $K^{(4)}$ using Equation 17.

3.4 Attacking $K^{(3)}$

Thus far we have recovered 128 bits of the secret key K, $K^{(1)}$ and $K^{(4)}$ using MSB differentials only. The next in turn is $K^{(3)}$, which appears to require a slightly more complicated attack also involving second highest bit.

We will be using the two new differentials in addition to the ones given in Observation 1 for initialization rounds $t = -4, \ldots, -1$ and Observation 2 for $t = 0$. For $t = 1$ the differential is:

$$\begin{gathered}\Delta v_{12_1} = \texttt{C000}\ ,\ \Delta v_{23_1} = d\ ,\ \Delta v_{34_1} = \texttt{8000} \\ \Delta(\mathrm{RS1}_1, \mathrm{RS2}_1, \mathrm{RS3}_1, \mathrm{RS4}_1, \mathrm{LFSR}_1) = (\texttt{8000}, \texttt{8000}, \texttt{0000}, \texttt{8000}, \texttt{0000}) \\ \Downarrow \\ \Delta(\mathrm{RS1}_2, \mathrm{RS2}_2, \mathrm{RS3}_2, \mathrm{RS4}_2, \mathrm{LFSR}_2) = (\texttt{0000}, \texttt{8000}, d, \texttt{4000}, \texttt{0000}).\end{gathered}$$

To make this differential work, we will use the known value for $\mathrm{RS1}_1$ obtained in Section 3.3. Loop through the values $y = v_{12_1} = 0, 1, \ldots, 2^{16} - 1$ and for each one of those make the following two-word encryption queries until $C_1 = C_1'$ condition is reached:

$$\begin{aligned}C =& \mathrm{HB}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (x, E^{-1}(y, K^{(1)}) \boxminus \mathrm{RS1}_1)\big) \\ C' =& \mathrm{HB}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), \\ & (x \boxplus \texttt{8000}, E^{-1}(y \boxplus \texttt{C000}, K^{(1)}) \boxminus \mathrm{RS1}_1 \boxminus \texttt{8000})\big)\end{aligned}$$

From the $C_1 = C_1'$ condition we will know that $\Delta v_{34_1} = \texttt{8000}$ as it cancels out the differential $\Delta\mathrm{RS4}_1 = \texttt{8000}$ before invocation of the last E function. When the condition is met by some x, $d = \Delta v_{23_1} = \Delta\mathrm{RS3}_2$ will be a quantity that satisfies

$$E^{-1}(v_{34_1}, K^{(4)}) \boxminus E^{-1}(v_{34_1} \boxplus \texttt{8000}, K^{(4)}) = d. \tag{21}$$

Now we will extend the chosen-plaintext attack by one more round. We will use the differential:

$$\begin{gathered}\Delta v_{12_2} = \texttt{8000}\ ,\ \Delta v_{23_2} = \texttt{0000}\ ,\ \Delta v_{34_2} = \texttt{8000} \\ \Delta(\mathrm{RS1}_2, \mathrm{RS2}_2, \mathrm{RS3}_2, \mathrm{RS4}_2, \mathrm{LFSR}_2) = (\texttt{0000}, \texttt{8000}, d, \texttt{4000}, \texttt{0000}) \\ \Downarrow \\ \Delta(\mathrm{RS1}_3, \mathrm{RS2}_3, \mathrm{RS3}_3, \mathrm{RS4}_3, \mathrm{LFSR}_3) = (\texttt{8000}, \texttt{4000}, d, \texttt{4000}, \texttt{0000}).\end{gathered}$$

We now proceed to deriving the contents of $\mathrm{RS1}_2$. We choose the first two plaintext words P_0, P_0', P_1, P_1' as before. For some z and $y = 0, 1, \ldots, 2^{16} - 1$ the third words will be chosen as

$$P_2 = E^{-1}(z, K^{(1)}) \boxminus y \tag{22}$$

$$P_2' = E^{-1}(z \boxplus \texttt{8000}, K^{(1)}) \boxminus y \tag{23}$$

until the corresponding ciphertext

$$C = \mathrm{HB}\big((\texttt{0000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (P_0, P_1, P_2)\big)$$
$$C' = \mathrm{HB}\big((\texttt{8000}, \texttt{0000}, \texttt{0000}, \texttt{0000}), (P'_0, P'_1, P'_2)\big)$$

satisfies the previous conditions and the additional condition

$$E^{-1}(C_2, K^{(4)}) \boxminus E^{-1}(C'_2, K^{(4)}) = \texttt{4000}. \tag{24}$$

This will imply that the second differential works and the conditions $\Delta v_{12_2} = \texttt{8000}$, $\Delta v_{23_2} = \texttt{0000}$, and $\Delta v_{34_2} = \texttt{8000}$ hold. Furthermore we will have the contents of register $\mathrm{RS1}_2 = y$ and $v_{12_2} = z$. Note that if the guess for $\mathrm{RS1}_2 = y$ is correct, then Equation 24 will hold for any z in Equations 22 and 23.

We now have sufficient information about the internal state of Hummingbird to mount a "quartet" attack on $K^{(3)}$. Additional quantities of the internal state can be derived as follows:

$$v_{34_1} = \mathrm{RS1}_2 \boxminus \mathrm{RS1}_1 \tag{25}$$
$$\mathrm{RS4}_1 = E^{-1}(C_1, K^{(4)}) \boxminus v_{34_1} \tag{26}$$
$$\mathrm{RS4}_2 = \mathrm{RS4}_1 \boxplus E(P_1 \boxplus \mathrm{RS1}_2, K^{(1)}) \boxplus \mathrm{RS1}_2 \tag{27}$$
$$v_{34_2} = E^{-1}(C_2, K^{(4)}) \boxminus \mathrm{RS4}_2. \tag{28}$$

We can now perform an exhaustive search for $K^{(3)}$ that satisfies

$$E^{-1}(v_{34_1}, K^{(3)}) \boxminus E^{-1}(v_{34_1} \boxplus \texttt{8000}, K^{(3)}) = d \quad \text{and} \tag{29}$$
$$E^{-1}(v_{34_2}, K^{(3)}) \boxminus E^{-1}(v_{34_2} \boxplus \texttt{8000}, K^{(3)}) = d \tag{30}$$

for some value d. We call this a "quartet test" as it involves four (inverse) E invocations. To get more quartets (you will need at least four), increase z in Equations 22 and 23 and perform more chosen-plaintext queries.

3.5 Attacking $K^{(2)}$

After the recovery of $K^{(1)}$, $K^{(3)}$ and $K^{(4)}$, there is only 64 bits of unknown keying material left to discover. A simple known-plaintext exhaustive search for $K^{(2)}$ will suffice to recover this last missing piece.

4 Discussion

Hummingbird has some superficial similarities to the Helix [3] and Phelix [10] ciphers – these are stream ciphers where message data is used to modify the internal state of the cipher and an authentication code is produced. An analysis by Muller also used a lack of high-bit propagation in a distinguishing attack [8].

4.1 Implementing the Attack

Our attack on Hummingbird-1 was implemented using the C language on Linux platform. Due to the divide-and-conquer technique that we are using, we may efficiently demonstrate the attack with keys that have limited entropy in each one of the subkeys. Our demonstration code attacks a variant that has four 24-bit subkeys, bringing the total effective key size to 96 bits. Note that this is not a reduced cipher; the subkey entropy has simply been reduced.

The demonstration code first performs a self-test of its Hummingbird-1 implementation against test vectors supplied by Revere Security It then chooses a random key and lets the attack code perform black-box chosen-IV encryption or decryption queries. Typical execution time is 15-20 seconds before the correct 96-bit key is found on an Intel Core 2 Duo clocked at 3.16 GHz.

It seems reasonable to assume that the the E function described in Section 2.2 offers less than 2^{64} security since its diffusion properties are far from perfect. To illustrate this, we note that in Figure 1 it is easy to see that the 16-bit subkey $K_4^{(i)}$ affects two invocations of the S-Box layer and a single bit linear diffusion layer – therefore a single bit change in this subkey won't even necessary affect all ciphertext bits. Since the security of Hummingbird-1 is reduced to the security of the E function by the techniques described in this paper, we feel confident in estimating that Hummingbird-1 offers significantly less than 64 bits of security.

Throughout this paper the any constant pair of IVs can be used as long as

$$\Delta(\mathrm{RS1}_0, \mathrm{RS2}_0, \mathrm{RS3}_0, \mathrm{RS4}_0, \mathrm{LFSR}_0) = (\mathtt{8000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000}).$$

This initial condition follows from $\Delta IV = (\mathtt{8000}, \mathtt{0000}, \mathtt{0000}, \mathtt{0000})$ by Observation 1, but if the flaw in the initialization function is fixed, we may find such pairs by the birthday paradox. If the initialization function would be completely random, finding such a pair would require about $\sqrt{2^{80}} = 2^{40}$ queries. Testing for the condition can be done with Observation 2.

4.2 Lessons Learned

Due to its extremely light-weight application target scenario, the security margins used in the design of Hummingbird-1 are very small. In addition to the unfortunate bug in the initialization function (Observation 1), the security of Hummingbird-1 seems to suffer from the fact its state size is very small and that chosen input can directly affect almost all of its internal state bits (apart from the LFSR "counter") in an adaptive attack. We suggest that the number of state bits which run independently from input data should be increased in future encryption algorithm designs of the Hummingbird type.

5 Conclusions

We have described a key-recovery attack against the 256-bit authenticated encryption primitive Hummingbird-1. The attack is based on a divide-and-conquer and differential techniques and has complexity upper bounded by 2^{64} operations. Significant improvements to this bound are possible by attacking the E function. The attack requires processing of few megabytes of chosen messages under two related nonces (IVs).

The attack proceeds in four stages, attacking each one of the 64-bit subkeys individually. The attacks are mainly based on differentials in in the high bits of words. It is noteworthy that the described attacks work regardless of the design of the main nonlinear component, the E keyed permutation. The present line of attack are made effective by a clear design flaw in the Hummingbird-1 initialization function, but similar attacks can be envisioned for many possible straightforward fixes.

We conclude that the published version of Hummingbird-1 may not offer adequate security for some cryptographic applications. The Revere Security team is actively developing an improved version that will remedy the security issues reported in this paper.

References

1. R. ANDERSON, E. BIHAM AND L. KNUDSEN. "Serpent: A Proposal for the Advanced Encryption Standard." `http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf`, 1999.
2. X. FAN, H. HU, G. GONG, E. M. SMITH AND D. ENGELS. "Lightweight Implementation of Hummingbird Cryptographic Algorithm on 4-Bit Microcontroller." The 1st International Workshop on RFID Security and Cryptography 2009 (RISC'09), pp. 838–844, 2009.
3. N. FERGUSON, D. WHITING, B. SCHNEIER, J. KELSEY, S. LUCKS, AND T. KOHNO "Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive." FSE 2003, LNCS 2887, Springer, pp. 330–346, 2003.
4. D. ENGELS, X. FAN, G. GONG, H. HU AND E. M. SMITH. "Ultra-Lightweight Cryptography for Low-Cost RFID Tags: Hummingbird Algorithm and Protocol." Centre for Applied Cryptographic Research (CACR) Technical Reports, CACR-2009-29. `http://www.cacr.math.uwaterloo.ca/techreports/2009/cacr2009-29.pdf`
5. D. ENGELS, X. FAN, G. GONG, H. HU AND E. M. SMITH. "Hummingbird: Ultra-Lightweight Cryptography for Resource-Constrained Devices." 1st International Workshop on Lightweight Cryptography for Resource-Constrained Devices (WLC'2010). Tenerife, Canary Islands, Spain, January 2010
6. R. FRAZER (ED.) "An Analysis of the Hummingbird Cryptographic Algorithm." Commercial security analysis report by Information Security Systems Inc. Dated 26 April 2009. `http://www.reveresecurity.com/pdfs/ISSI_Hummingbird.pdf`
7. REVERE SECURITY. Web page and infomation on the Hummingbird cipher. Fetched 03-Nov-2010. `http://www.reveresecurity.com/`

8. F. MULLER. "Differential Attacks against the Helix Stream Cipher." FSE 2004, LNCS 3017, Springer, oo. 94–108, 2004.
9. E. M. SMITH. Personal communication, July 7, 2010.
10. D. WHITING, B. SCHNEIER, S. LUCKS, AND F. MULLER. "Phelix – Fast Encryption and Authentication in a Single Cryptographic Primitive." ECRYPT Stream Cipher Project Report 2005/027. `http://www.schneier.com/paper-phelix.html`, 2005.