DIFFERENTIAL FAULT ANALYSIS OF HUMMINGBIRD

Yaser Esmaeili Salehani and Amr Youssef
Concordia Institute for Information Systems Engineering, Concordia University
Montreal, H3G 1M8, Quebec, Canada
y_esmae@encs.concordia.ca, youssef@ciise.concordia.ca

Keywords: Hummingbird, Fault analysis, Block ciphers, Stream ciphers, Light-weight cryptography.

Abstract: Hummingbird is a lightweight encryption algorithm proposed by Engels, Fan, Gong, Hu and Smith at FC'10. Unlike other lightweight cryptographic primitives which can be classified as either block ciphers or stream ciphers, Hummingbird has a hybrid structure of block cipher and stream cipher with 16-bit block size, 256-bit key size, and 80-bit internal state. Preliminary analysis conducted by the cipher's designers show that it is resistant to most common attacks against block ciphers and stream ciphers.

In this paper, we present a differential fault analysis attack on Hummingbird. The fault model in which we analyze the cipher is the one in which the attacker is assumed to be able to fault a random word before the linear transform, after the s-boxes, of the four block ciphers which are used in the Hummingbird encryption process but cannot control the exact location of injected faults. Our attack, which recovers the 256-bit key, requires around 50 faults and 2^{66} steps.

1 INTRODUCTION

Hummingbird (Engels et al., 2010) (Fan et al., 2009) is an encryption algorithm designed for lightweight software and lightweight hardware implementations on resource-constrained devices such as RFID tags and wireless sensor nodes. Its design was inspired by the Enigma machine which led to a hybrid combination of block cipher and stream cipher structures.

The security of Hummingbird was evaluated by its designers who concluded that the the cipher is resistant to most common attacks against block ciphers and stream ciphers including birthday attacks, differential and linear cryptanalysis, structure attacks, algebraic attacks, and cube attacks. Also a chosen-IV and chosen-message attack was reported by Saarinen (Saarinen, 2011).

In this paper, we present a differential fault analysis attack on Hummingbird. The fault model in which we analyze the cipher is the one in which the attacker is assumed to inject a transient fault at a random 4 bit word before the linear transformation, after the 4×4 s-boxes, of the four block ciphers which are used in the Hummingbird encryption process but cannot control the exact location of injected faults. We also assume that the attacker is able to reset the cipher an arbitrary number of times.

In general, fault analysis attacks (Boneh et al., 1997) fall under the category of implementation dependent attacks, which include side channel attacks such as timing analysis (Kocher, 1996) and power analysis (Kocher et al., 1999). Fault attacks proved to be one of the most effective attack techniques against embedded cryptographic devices and can be used when the attacker has access to the cryptographic device even for a short time. They were first used by Boneh *et al.* (Boneh et al., 1997) to attack public key cryptosystems such as RSA (by using a faulty Chinese Remainder Theorem (CRT) computation to factor the modulus n). Then fault analysis has been applied to several symmetric cryptographic algorithms including block ciphers (Biham and Shamir, 1997) (Giraud and Thillard, 2010) (Piret and Quisquater, 2003) (Kim, 2010) and stream ciphers (Biham et al., 2005) (Hoch and Shamir, 2004).

In fault analysis attacks, fault injections are achieved by applying some kind of physical influence such as variations in the supply voltage, external clock, or temperature beyond the nominal operating range of the device, which results in a corruption of the internal memory or the computation process. Other techniques for fault injection include subjecting the device to white light, laser beams, X-rays or ion beams (Anderson and Kuhn, 1997) (Bar-El et al., 2004). The examination of the results under such faults often reveals some information about the cipher key or the secret inner state.

The main idea of our attack is inspired by recent

differential fault analysis attacks against the AES. In these attacks, the attacker collects few differential pairs relative to the last non-linear step which allows the attacker to reduce and finally guess the values computed in the last rounds and infer the last round key. However, unlike the AES case in which once the round key has been recovered, the key schedule can be inverted to obtain the initial secret key, this is not possible for Hummingbird since it does not have an explicit invertible key schedule. Instead, the above procedure has to be reiterated on each round, starting from the last one, until the whole key material is exposed. Our simulation results showed that our attack, which recovers the 256-bit key, requires around 50 fault injections and 2^{66} steps.

The rest of the paper is organized as follows. A brief description of the relevant details of Hummingbird is provided in the next section. Our attack and its complexity analysis are provided in section 3. Finally, we conclude the paper in Section 4.

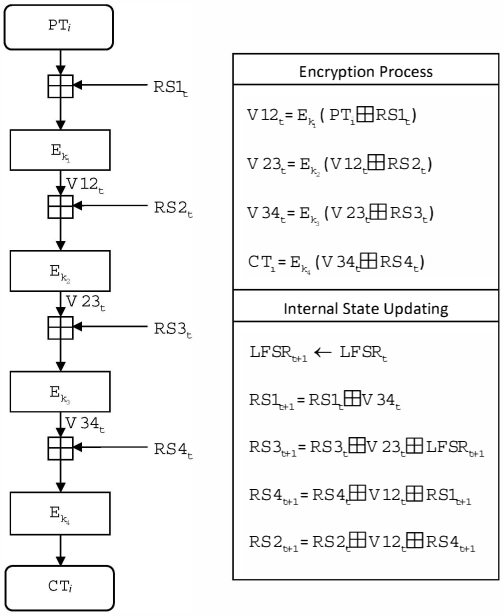

Figure 1: An overview of the Hummingbird encryption process.

2 DESCRIPTION OF HUMMINGBIRD

The following notation and functions are used throughout the paper:

- $\boxplus$: addition mod 2^{16}.
- $\oplus$: bit-wise XOR.
- $||$: Concatenation of the words.
- $<<<$: left rotation defined on 16-bit value.
- S_i: the i^{th} 4-bit s-box where $i = 1,2,3,4$.
- $SBOX$: the 16-bit nonlinear function which equals to $S_1||S_2||S_3||S_4$.
- $SBOX^{-1}$: the inverse mapping of $SBOX$.
- $L(x)$: the 16-bit linear transform equation.
- $L^{-1}(x)$: the inverse mapping of $L(x)$.
- $\triangle X_i$: the 4-bit differential input of S_i.
- $\triangle Y_i$: the 4-bit differential input of S_i.

As mentioned above, Hummingbird has a hybrid structure and consists of two main components: a stream cipher and a block cipher. The input/output block size is 16-bit and the internal state and key size are 80-bit and 256-bit, respectively.

Figure 1 depicts a top level view of the encryption process. The cipher algorithm consists of four 16-bit block ciphers, $E_{k_i}(i = 1,2,3,4)$, four 16-bit internal state registers $RSi(i = 1,2,3,4)$, and a 16-bit linear feedback shift register (LFSR). The original 256-bit key is divided into four 64-bit subkeys $k_i, i = 1,2,3,4$, which are used in the four block ciphers.

A 16-bit plaintext block PT_i is added to the first internal state register $RS1$ modulo 2^{16}. The result of the addition is then encrypted by the first block cipher E_{k_1}. This procedure is repeated in a similar manner for another three times and the output of E_{k_4} is the corresponding ciphertext, CT_i. At the same time, the states of the four internal state registers will also be updated based on their current state values, the outputs of the first three block ciphers, and the value of the LFSR. The decryption process follows a similar pattern as in the encryption mode.

In practice, Hummingbird is initialized with four 16-bit random nonces, $NONCE_i, i = 0,1,2,3$, to construct the four internal state registers $RS_i, i = 1,2,3,4$, respectively, followed by applying the encryption algorithm to the message $RS1 \boxplus RS3$. The final 16-bit ciphertext of the initialization is used to initialize the LFSR where the 13^{th} bit of the LFSR is always set to 1.

The 16-bit block cipher is a typical substitution-permutation (SP) network with 16-bit block size and 64-bit key. It consists of five rounds: four regular rounds and a final round that only includes the key mixing and the s-box substitution steps. Each round comprises of a key mixing step, a substitution layer, and a permutation layer. For the key mixing, a simple xor operation is used. Figure 2 depicts the structure of the 16-bit block cipher and the specification of the four s-boxes and linear transform mapping which are used in E_{k_i}.

The 64-bit subkey k_i is divided into four 16-bit round keys $K_j^{(i)}, j = 1,2,3,4$, which are used in the four regular rounds of SP structure, respectively.

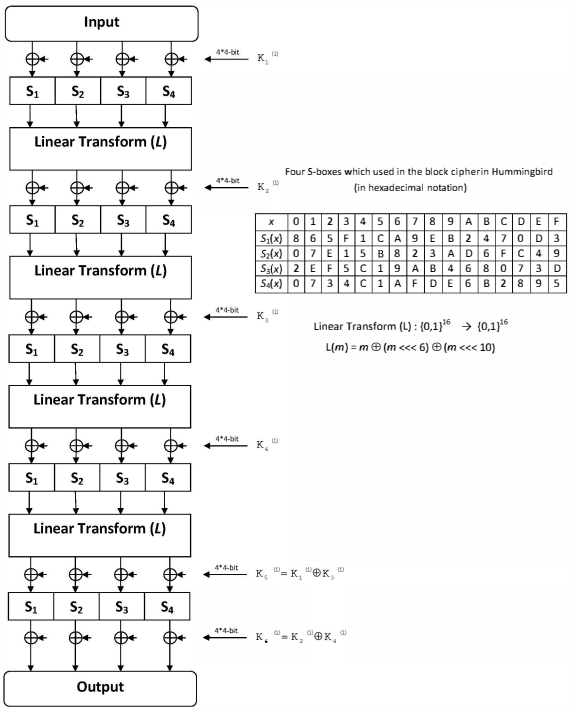

Figure 2: The structure of E_{k_i} - the 16-bit block cipher of HUMMINGBIRD in encryption mode.

Moreover, two keys $K_5^{(i)}$ and $K_6^{(i)}$ which are directly derived from the four round keys are applied before and after the last s-boxes.

Further details about the encryption, decryption and initialization processes can be found in (Engels et al., 2010).

3 THE PROPOSED ATTACK

The main idea of using differential fault analysis against Hummingbird is to recover whole secret keys based on determining round keys in four steps. In particular, we can retrieve k_4 and peel off the last round function of the encryption process (E_{k_4}) to determine $V34_t \boxplus RS4_t$. Detailed explanation of how to recover the keys of the E_{k_i} step is provided in the next subsection. By guessing all the 2^{16} values of $RS4_t$ and applying the same method for determining the keys of the 16-bit block cipher (E_{k_3}), k_3 can be revealed. Similarly, the values of k_2 and k_1 can be determined by applying the same procedure and guessing $RS3_t$ and $RS2_t$. Finally by guessing $RS2_t, RS3_t$ and $RS4_t$ we can determine 2^{48} candidates for k_1, k_2, k_3 and k_4. The correct key can then be uniquely determined using additional plaintext/ciphertext pairs.

3.1 Key recovery of E_{k_i}

In this section, we describe a differential fault analysis on the E_{k_i} function which are used in Hummingbird algorithm. The injected faults are assumed to occur in one 4-bit word before the linear transform and after the 4×4 s-boxes.

Table 1: Possible Differential outputs of the linear transformation - The $\mathcal{D}$'s list.

	Differential input = ($\triangle X_1 \triangle X_2 \triangle X_3 \triangle X_4$)			
	$\triangle X_1 \neq 0$	$\triangle X_2 \neq 0$	$\triangle X_3 \neq 0$	$\triangle X_4 \neq 0$
The corresponding differential output	1440	0144	4014	4401
	2880	0288	8028	8802
	3CC0	03CC	C03C	CC03
	4011	1401	1140	0114
	5451	1545	5154	4515
	6891	1689	9168	8916
	7CD1	17CD	D17C	CD17
	8022	2802	2280	0228
	9462	2946	6294	4629
	A8A2	2A8A	A2A8	8A2A
	BCE2	2BCE	E2BC	CE2B
	C033	3C03	33C0	033C
	D473	3D47	73D4	473D
	E8B3	3E8B	B3E8	8B3E
	FCF3	3FCF	F3FC	CF3F

Differential output = ($\triangle Y_1 \triangle Y_2 \triangle Y_3 \triangle Y_4$).
All values are in Hexadecimal.

Consider one 4-bit word differences at the input of the linear layer $L(x)$. We have 60 ($= 15 \times 4$) possible such differences corresponding to 4 different possible locations and 15 different possible values. Because of the linearity, the number of corresponding possible differences at its output is also 60 but, while the input difference affected one 4-bit word only, the output difference affects up to 12 bits because of the diffusion linear transformation layer (See Table 1.) Note that the key addition does not change the set of possible differences.

First we describe how we can recover the last round subkey $K_6^{(i)}$. We follow the same idea of the attack described in (Piret and Quisquater, 2003). Algorithm 1 shows the details.

Algorithm 1:

1. Compute the 60 possible differences at the output of linear transform ($L(x)$), i.e. the 60 values of $L(x)$, where $x = x_1||x_2||x_3||x_4$ and only one of the x_i's has a Hamming weight not equal to zero. Store the obtained values in a list $\mathcal{D}$.
2. Consider a plaintext P, its corresponding ciphertext C and faulty ciphertext C'.

Table 2: Summary of the Fault Attack on Hummingbird.

Fault location		The recovered data	The recovered keys
E_{k_4}	round 4	$K_2^{(4)} \oplus K_4^{(4)}$	
	round 3	$K_1^{(4)} \oplus K_3^{(4)}$	
	round 2	$K_4^{(4)}$	$K_2^{(4)}$,$K_4^{(4)}$
	round 1	$K_3^{(4)}$	$K_1^{(4)}$,$K_3^{(4)}$
Guess all of 2^{16} possibilities of $RS4_t$		$\longmapsto$	Get $V34_t$
E_{k_3}	round 4	$K_2^{(3)} \oplus K_4^{(3)}$	
	round 3	$K_1^{(3)} \oplus K_3^{(3)}$	
	round 2	$K_4^{(3)}$	$K_2^{(3)}$,$K_4^{(3)}$
	round 1	$K_3^{(3)}$	$K_1^{(3)}$,$K_3^{(3)}$
Guess all of 2^{16} possibilities of $RS3_t$		$\longmapsto$	Get $V23_t$
E_{k_2}	round 4	$K_2^{(2)} \oplus K_4^{(2)}$	
	round 3	$K_1^{(2)} \oplus K_3^{(2)}$	
	round 2	$K_4^{(2)}$	$K_2^{(2)}$,$K_4^{(2)}$
	round 1	$K_2^{(4)}$	$K_1^{(2)}$,$K_3^{(2)}$
Guess all of 2^{16} possibilities of $RS2_t$		$\longmapsto$	Get $V12_t$
E_{k_1}	round 4	$K_2^{(1)} \oplus K_4^{(1)}$	
	round 3	$K_1^{(1)} \oplus K_3^{(1)}$	
	round 2	$K_4^{(1)}$	$K_2^{(1)}$,$K_4^{(1)}$
	round 1	$K_3^{(1)}$	$K_1^{(1)}$,$K_3^{(1)}$

3. Guess the value of the round key $K_6^{(i)}$.
4. Compute the difference $SBOX^{-1}(C \oplus K_6^{(i)}) \oplus SBOX^{-1}(C' \oplus K_6^{(i)})$. Check whether it is in $\mathcal{D}$. If yes, add the round key to the list $\mathcal{L}$ of possible candidates.
5. Consider a new plaintext P (with corresponding C and C') and go back to step 2. This time, the round key guesses only go through the list $\mathcal{L}$ of possible candidates. If the difference computed at step 4 is not in $\mathcal{D}$, remove the candidate from $\mathcal{L}$). Repeat until there remains only one candidate in $\mathcal{L}$.

The complexity of the Algorithm 1 is around 2^{16} since we have to search all of the target key at the beginning of step 3 of the algorithm. After $K_6^{(i)}$ is uniquely determined, the last round is peeled off, and the attack is repeated on the reduced cipher.

We can find the subkey $K_5^{(i)}$ by repeating the same algorithm which is used to recover $K_6^{(i)}$ but using the difference equation $SBOX^{-1}(L^{-1}(C_{-1} \oplus K_5^{(i)})) \oplus SBOX^{-1}(L^{-1}(C'_{-1} \oplus K_5^{(i)}))$ where $C_{-1} = SBOX^{-1}(C \oplus K_6^{(i)})$ and $C'_{-1} = SBOX^{-1}(C' \oplus K_6^{(i)})$. $K_3^{(i)}$ and $K_4^{(i)}$ are revealed in the same way. Finally, we determine the remaining subkeys of $K_1^{(i)}$ and $K_2^{(i)}$ by computing $K_3^{(i)} \oplus K_5^{(i)}$ and $K_4^{(i)} \oplus K_6^{(i)}$, respectively.

The above attack was simulated for 10,000 times using different random values for the 16-bit input and 64-bit subkey of k_i for E_{k_i}. Based on our simulations, our attack requires an average of 12.51 fault injections to recover the whole subkeys, $k_i, i = 1,2,3,4$. The recorded minimum and the largest number of required faults were 8 and 22, respectively.

Table 2 summarizes the whole steps of the attack to recover the 256-bit secret key.

After the keys are recovered, we can then find the other internal registers by applying the initialization process since the values of $NONCE_i (i = 0,1,2,3)$ are public.

3.2 The Complexity of our Attack

To recover all of the 256-bit secret key we have to apply the above fault attack for four times and guess

all candidate values of the three internal registers: $RS2_t, RS3_t$ and $RS4_t$. From our experimental results, we need around 12.5 faulty values to uniquely determine each subround key. Thus, our attack requires 50 faulty ciphertext, 2^{48} guessing of 16-bit values and calling Algorithm 1 (with complexity 2^{16}) in each step to reveal the subkeys. Since we have to use four times of the Algorithm 1 in each step of E_{k_i}, the total complexity of our attack is about $4 \times 2^{16} \times 2^{48} = 2^{66}$.

4 CONCLUSIONS

In this paper, we presented a fault attack against a newly introduced ultra lightweight encryption algorithm, Hummingbird. Each 64-bit round key can be found, on average, using 12.51 faulty encryptions. If we assume that the 256-bit secret key and 80-bit internal state have random distribution, then the whole cipher can be broken after around 50 faults. To fully recover the key, we have to guess 2^{48} values of three 16-bit internal state registers which brings the whole complexity of our attack to 2^{66}.

REFERENCES

Anderson, R. and Kuhn, M. (1997). Low cost attacks on tamper resistant devices. In *Proc. of the 5th international workshop on security protocols, LNCS-1361, pp. 125-136*. Springer-Verlag.

Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., and Whelan, C. (2004). The sorcerers apprentice guide to fault attacks. In *Proc. of DSN 2004, pp. 330342*.

Biham, E., Granboulan, L., and Nguyen, P. Q. (2005). Impossible fault analysis of rc4 and differential fault analysis of rc4. In *Proc. of FSE 2005, LNCS 3557, pp. 359-367*. Springer-Verlag.

Biham, E. and Shamir, A. (1997). Differential fault analysis of secret key cryptosystem. In *Proc. of Crypto 1997, LNCS 1294, pp. 513-528*. Springer-Verlag.

Boneh, D., Demillo, R., and Lipton, R. (1997). On the importance of checking cryptographic prtocols for faults. In *Proc. of Eurocrypt 1997, LNCS 1233, pp. 37-51*. Springer-Verlag.

Engels, D., Fan, X., Gong, G., Hu, H., and Smith, E. M. (2010). Hummingbird: Ultra-lightweight cryptography for resource-constrained devices. In *Proc. of Financial Cryptography 2010, LNCS 6054, pp. 3-18*. Springer-Verlag.

Fan, X., Hu, H., Gong, G., Smith, E. M., and Engels, D. (2009). Lightweight implementation of hummingbird cryptographic algorithm on 4-bit microcontroller. In *The 1st International Workshop on RFID Security and Cryptography 2009 (RISC09), pp. 838-844*.

Giraud, C. and Thillard, A. (2010). Piret and quisquater's dfa on aes revisited. Available at http://eprint.iacr.org/2010/440.

Hoch, J. and Shamir, A. (2004). Fault analysis of stream ciphers. In *Proc. of CHES 2004, LNCS 3156, pp. 240253*. Springer-Verlag.

Kim, C. H. (2010). Differential fault analysis against aes-192 and aes-256 with minimal faults. In *Proc. of FDTC 2010, pp.3-9,IEEE Computer Society*. IEEE.

Kocher, P. (1996). Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proc. of Crypto 1996,LNCS-1109, pp. 104-113*. Springer-Verlag.

Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Proc. of Crypto 1999, LNCS-1666, pp. 388-397*. Springer-Verlag.

Piret, G. and Quisquater, J. J. (2003). A differential fault attack technique against spn structures, with application to the aes and khazad. In *Proc. of CHES 2003, LNCS 2779, pp. 77-88*. Springer-Verlag.

Saarinen, M. (2011). Cryptanalysis of hummingbird-1. In *18th International Workshop on Fast Software Encryption (FSE 2011), to appear*. Also, available at: http://eprint.iacr.org/2010/612.pdf.