A Cryptanalysis of HummingBird-2: The Differential Sequence Analysis

Qi Chai and Guang Gong

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{q3chai, ggong}@uwaterloo.ca

Abstract. Hummingbird-2 is one recent design of lightweight block ciphers that enables compact hardware implementations, ultra-low power consumption and stringent response time as specified in ISO18000-6C.
In this paper, we present cryptanalytic results on the full version of this cipher using two pairs of related keys. We discover that the differential sequences for the last invocation of the round function can be computed by running the full cipher, due to which the search space for the key can be reduced. Base upon this observation, we propose a probabilistic attack encompassing two phases, preparation phase and key recovery phase. The preparation phase, requiring 2^{80} effort in time, reaches the internal states satisfying particular conditions with 0.5 probability. In the key recovery phase, by using the proposed differential sequence analysis (DSA) against the encryption (decryption resp.), 36-bit (another 44-bit resp.) of the 128-bit key could be recovered. Additionally, the rest 48-bit of the key can be exhaustively searched and the overall time complexity of this phase is $2^{48.14}$.
Note that the proposed attack, though exhibiting an interesting tradeoff between success probability and time complexity, is only of a theoretical interest at the moment and does not affect the practical security of the Hummingbird-2.

Keywords: lightweight cryptography, differential cryptanalysis, saturation attack

1 Introduction

Passive RFID tags and other constraint computing devices are usually characterized by extremely tight cost and power consumption requirements. The needs of cryptographic primitives on such devices have been increasing with the growing pervasiveness and mass deployment of these devices. To this end, considerable lightweight stream/block ciphers are proposed in recent years, targeting very small hardware footprint and reduced power consumption. Typical examples are listed in Table 1. Meanwhile, cryptanalysis of these lightweight primitives has received considerable attention due to a widely-accept concern – the pursue of efficiency by reducing the security margin or applying innovative but less well understood techniques lead lightweight candidates to be less durable relative to regular symmetric ciphers. This concern has been further confirmed by the successful cases of attacking KeeLoq [33], Crypto-I [32], Atmel Cipher [22, 9], PRESENT [13, 11], KTANTAN [7, 3], PRINTCipher [2, 29], reduced KLEIN [1], A2U2 [12] and so on.

A Brief History of Hummingbird Cipher: Motivated by the design of the well-known Enigma machine, the first generation of Hummingbird (call it HB-1) was proposed by the engineers in Revere

Table 1. Recent Design/Implementation of Lightweight Ciphers (ordered by gate equivalent (GE))

	Key size[bits]	Block size[bits]	Area[GE]	Throughput[Kb/s]	Logic process[μm]
PRINTCipher-48 [26]	80	48	402	6.25	0.18
KTANTAN32 [14]	80	32	462	12.5	0.13
PRINTCipher-48 [26]	80	48	503	100	0.18
KTANTAN48 [14]	80	48	571	9.4	0.13
GOST [34]	256	64	651	24.24	0.18
Piccolo-80 [38]	80	64	683	14.8	0.13
KTANTAN64 [14]	80	64	684	8.4	0.13
LED-64 [21]	64	64	688	5.1	0.18
LED-128 [21]	128	64	700	3.4	0.18
PRINTCipher-96 [26]	160	96	726	3.13	0.18
Piccolo-128 [38]	128	64	758	12.1	0.13
KATAN32 [14]	80	32	802	12.5	0.13
KATAN48 [14]	80	48	916	9.4	0.13
PRINTCipher-96 [26]	160	96	967	100	0.18
KATAN64 [14]	80	64	1,027	8.4	0.13
PRESENT [35]	80	64	1,075	11.4	0.18
KLEIN-64 [20]	64	64	1,981	N/A	0.18
KLEIN-80 [20]	80	64	2,097	N/A	0.18
HummingBird-2 [17]	**128**	**16**	**2,159**	**N/A**	**0.13**
KLEIN-96 [20]	96	64	2,213	N/A	0.18
AES [19]	128	128	3,400	12.4	0.35

Security and was further analyzed and published in [16] as an ultra-lightweight cryptographic algorithm targeting low-cost RFID tags, smart cards, and wireless sensor nodes to meet the stringent response time and power consumption requirements. Although HB-1, with an innovative hybrid structure of block cipher and stream cipher, was designed to provide 256-bit security, Saarinen, in FSE'11, showed a chosen-IV and chosen-message attack [36] that can recover the full secret key with at most 2^{64} off-line computational effort under two related IVs. Recently, Revere Security published the second generation of Hummingbird (call it Hummingbird-2 or HB-2) in [17], which inherits the design philosophy from HB-1, e.g., it has a small block size of 16-bit to adapt the needs of encrypting short messages in RFID applications and it retains the hybrid structure as a security compensation for the small block size. High level differences between HB-1 and HB-2 are: (1) key size has been reduced to 128 bits to satisfy the actual need for constrained devices; (2) size of the internal state has been increased from 80 bits to 128 bits; (3) the nonlinear keyed transformation in HB-2 has four invocations of the S-boxes, compared to five in HB-1, to further increase the throughput.

In addition, it is claimed in the same paper that HB-2 can withstand differential, linear and algebraic attacks and the four 4-bit S-Boxes in HB-2 belong to the optimal classes as discussed in [31]. Its resistance to the side-channel cube attack is recently investigated in [18], where the author applied cube attack to recover 48 bits of the secret key providing the attacker could access the internal states of HB-2 during an early stage in the initialization. However, this attack is marginal since it only threats HB-2 before the finishing of its initialization.

Our Contribution: By refining/improving our preliminary results in [10], we present, in this paper, the cryptanalytic result on the full version of this cipher using two pairs of related keys. Our attack makes use of the internal state of such a cipher and our philosophy is general: (1) the outputs of the encryption/decryption may leak information of the subkeys (under the differential cryptanalysis) as long as the internal states of the cipher satisfy particular conditions; (2) due to the birthday paradox, such a condition always happens with 1/2 probability providing $2^{L/2}$ attempts are made, where L (in bit) is the size of the internal state. To be specific, we propose the following attack encompassing two phases, a probabilistic preparation phase and a key recovery phase.

- To realize the two particular conditions regarding the internal states, the preparation phase spends 2^{80} effort in time to achieve the succeed probability 0.5 (due to the birthday paradox). If succeeds, one could proceed to the key recovery phase.
- The key recovery phase is basically an instance of a novel cryptanalytic technique – we call it differential sequence analysis (DSA) – which can be seen as a hybrid of the conventional differential cryptanalysis and saturation attack. After exhibiting DSA's definitions and properties, we present its applications in the attacking scenarios, i.e.,
  - by using the encryption of HB-2, DSA recovers 36-bit (out of 128-bit) of the key, if condition (A) (regarding HB-2's secret key, input and internal state) holds.
  - by using the decryption of HB-2, DSA recovers another recovers another 44-bit of the key, if condition (B) (regarding HB-2's secret key, input and internal state) holds.
  - the rest 48-bit of the key can be exhaustively searched and the overall time complexity is $2^{48.14}$.

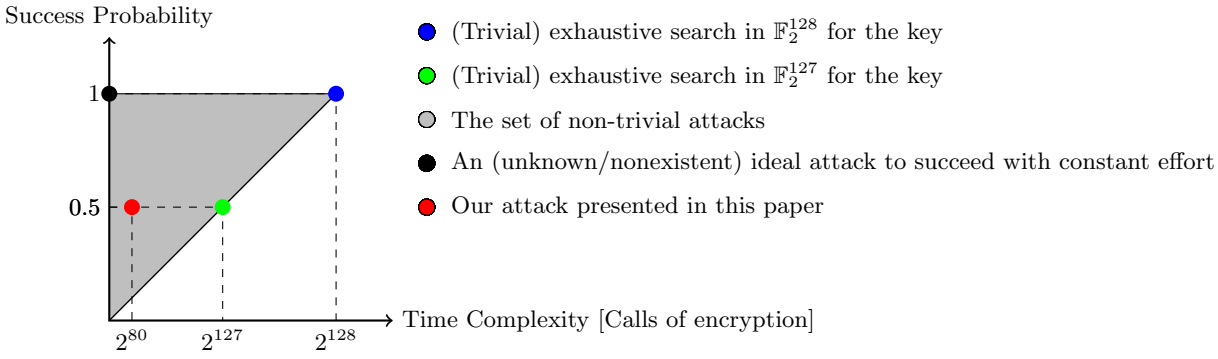

Fig. 1. Tradeoff between Success Probability and Time Complexity when attacking HB-2 (In fact, since one encryption only provides 16-bit entropy of the key, the exhaustive search needs a bit more than 2^{128} calls of the encryption function following the "key testing" procedure as desired in [4], i.e., $2^{128}+2^{112}+2^{96}+2^{80}+2^{64}+2^{48}+2^{32}+2^{16} \approx 2^{128.000022}$ and 8 plaintext-ciphertext pairs to uniquely determine the key with probability 1.)

Note that our results in this paper exhibit an interesting tradeoff between success probability and time complexity for HB-2, as shown in Fig. 1, which is analog to the collision attack in the hash function due to the birthday paradox. Stated in another way, to be successful with probability 0.5, our attack is faster than the exhaustive search by a factor of 2^{50}. Unfortunately, to succeed with probability 1, our

preparation phase requiring more effort in time than the exhaustive search, which makes the proposed method only of theoretical interests at the moment.

Organization: In Section 2, the specification of HB-2 is presented. Section 3 describes the principle of our attack at a high level. In Section 4, we devise the DSA technique, discuss its properties and how to use it to break parts of HB-2. In Section 5, we show how to achieve the desired conditions. We conclude the paper in Section 6.

Notations: Throughout the rest of this paper, we make use of the following notation for illustration.

- An hexadecimal number is indicated by a prefix "0x", e.g., 0x10 = 16.
- Unless otherwise stated, "+" denotes the addition in $\mathbb{F}_2$, which can also be vector-wise, e.g., $(a, b) + (c, d) = (a + c, b + d)$, where $a, b, c, d \in \mathbb{F}_2^m$.
- "$\boxplus$" or "$\boxminus$" operator denote addition or subtraction modulo 2^{16}.
- The high-bit XOR differential is defined as H =0x8000, a nice property of which is, given $x, x', y \in \mathbb{F}_2^{16}$ and $x + x' = H$, the following holds with probability 1,

$$(x \boxplus y) + (x' \boxplus y) = H, \quad (x \boxminus y) + (x' \boxminus y) = H, \quad (y \boxminus x) + (y \boxminus x') = H.$$

  That is to say, as pointed out in [36], the differential H behaves the same under "+" and "$\boxplus/\boxminus$".

2 Specification of Hummingbird-2

Hummingbird-2 is a 16-bit block cipher with a 128-bit secret key $K = (K_1, ..., K_8) \in (\mathbb{F}_2^{16}, ..., \mathbb{F}_2^{16}) = \mathbb{F}_2^{128}$ and a 64-bit public initialization vector $IV = (IV_1, ..., IV_4) \in (\mathbb{F}_2^{16}, ..., \mathbb{F}_2^{16}) = \mathbb{F}_2^{64}$. As opposed to conventional block ciphers, it has an 128-bit internal state $R = (R_1, ..., R_8) \in (\mathbb{F}_2^{16}, ..., \mathbb{F}_2^{16}) = \mathbb{F}_2^{128}$, which participates in each encryption/decryption and is updated after that.

Building Block: $WD16 : \{0,1\}^{16} \mapsto \{0,1\}^{16}$ is the fundamental block or round function of HB-2 encryption, which is defined as

$$WD16(x, K_a, K_b, K_c, K_d) = f(f(f(f(x + K_a) + K_b) + K_c) + K_d),$$

where x is the varying input, e.g., plaintext, intermediate state, K_a, K_b, K_c, K_d are four 16-bit secret keys and the nonlinear function f is specified as

$$\begin{aligned} S(x) &= S_1(x_1)||S_2(x_2)||S_3(x_3)||S_4(x_4), x = (x_1, x_2, x_3, x_4) \\ L(x) &= x + (x <<< 6) + (x <<< 10) \\ f(x) &= L(S(x)). \end{aligned}$$

Note that the four S-boxes, i.e., $S_1(x_i)$ to $S_4(x_i)$, are given in Table 2.

Besides, the inverse of $WD16$ is employed in the decryption, which is defined as

$$WD16^{-1}(y, K_d, K_c, K_b, K_a) = f^{-1}(f^{-1}(f^{-1}(f^{-1}(y) + K_d) + K_c) + K_b) + K_a,$$

Table 2. S-boxes in HummingBird-2

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

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

where $y = WD16(x, K_a, K_b, K_c, K_d)$ and f^{-1} is the inverse of f. The four S-boxes used in f^{-1} are also listed in Table 2.

Initialization: Hummingbird-2 is initialized before use. Let $(R_1^{(r)}, ...R_8^{(r)}) \in \{0,1\}^{128}$ denote the internal state at the rth iteration in the initialization. The initialization can thus be formulated as, for $r = 0, 1, 2, 3$,

$$t_1 = WD16(R_1^{(r)} \boxplus < r >, K_1, K_2, K_3, K_4) \tag{1}$$
$$t_2 = WD16(R_2^{(r)} \boxplus t_1, K_5, K_6, K_7, K_8) \tag{2}$$
$$t_3 = WD16(R_3^{(r)} \boxplus t_2, K_1, K_2, K_3, K_4) \tag{3}$$
$$t_4 = WD16(R_4^{(r)} \boxplus t_3, K_5, K_6, K_7, K_8) \tag{4}$$
$$R_1^{(r+1)} = (R_1^{(r)} \boxplus t_4) \lll 3 \tag{5}$$
$$R_2^{(r+1)} = (R_2^{(r)} \boxplus t_1) \lll 1 \tag{6}$$
$$R_3^{(r+1)} = (R_3^{(r)} \boxplus t_2) \lll 8 \tag{7}$$
$$R_4^{(r+1)} = (R_4^{(r)} \boxplus t_3) \lll 1 \tag{8}$$
$$R_5^{(r+1)} = R_5^{(r)} + R_1^{(r+1)} \tag{9}$$
$$R_6^{(r+1)} = R_6^{(r)} + R_2^{(r+1)} \tag{10}$$
$$R_7^{(r+1)} = R_7^{(r)} + R_3^{(r+1)} \tag{11}$$
$$R_8^{(r+1)} = R_8^{(r)} + R_4^{(r+1)}, \tag{12}$$

where $< r >$ represents a counter and $(R_1^{(0)}, ..., R_8^{(0)}) = (IV_1, IV_2, IV_3, IV_4, IV_1, IV_2, IV_3, IV_4)$.

Encryption: After the initialization, each encryption, by invoking the round function for four times, transforms a single plaintext word $P_i \in \mathbb{F}_2^{16}, i = 1, 2, ...$, to a corresponding ciphertext word C_i, i.e.,

$$t_1 = WD16(R_1^{(i)} \boxplus P_i, K_1, K_2, K_3, K_4) \tag{13}$$
$$t_2 = WD16(R_2^{(i)} \boxplus t_1, K_5 + R_5^{(i)}, K_6 + R_6^{(i)}, K_7 + R_7^{(i)}, K_8 + R_8^{(i)}) \tag{14}$$
$$t_3 = WD16(R_3^{(i)} \boxplus t_2, K_1 + R_5^{(i)}, K_2 + R_6^{(i)}, K_3 + R_7^{(i)}, K_4 + R_8^{(i)}) \tag{15}$$
$$C_i = WD16(R_4^{(i)} \boxplus t_3, K_5, K_6, K_7, K_8) \boxplus R_1^{(i)}, \tag{16}$$

where $(R_1^{(i)}, ..., R_8^{(i)}) \in \mathbb{F}_2^{128}$ is the internal state during the ith encryption and it is updated, at the end of the encryption, as follows:

$$R_1^{(i+1)} = R_1^{(i)} \boxplus t_3 \tag{17}$$
$$R_2^{(i+1)} = R_2^{(i)} \boxplus t_1 \tag{18}$$
$$R_3^{(i+1)} = R_3^{(i)} \boxplus t_2 \tag{19}$$
$$R_4^{(i+1)} = R_4^{(i)} \boxplus t_1 \boxplus R_1^{(i+1)} \tag{20}$$
$$R_5^{(i+1)} = R_5^{(i)} + R_1^{(i+1)} \tag{21}$$
$$R_6^{(i+1)} = R_6^{(i)} + R_2^{(i+1)} \tag{22}$$
$$R_7^{(i+1)} = R_7^{(i)} + R_3^{(i+1)} \tag{23}$$
$$R_8^{(i+1)} = R_8^{(i)} + R_4^{(i+1)} \tag{24}$$

A shorthand of Eq. (13)-(24) is $C_i = E(P_i, K) = E(P_i, (K_1, ..., K_8))$.

Decryption: Decryption of a single word $C_i \in \mathbb{F}_2^{16}, i = 1, 2, ...$, followed by the same initialization, is

$$u_3 = WD16^{-1}(C_i \boxminus R_1^{(i)}, K_8, K_7, K_6, K_5) \tag{25}$$
$$u_2 = WD16^{-1}(u_3 \boxminus R_4^{(i)}, K_4 + R_8^{(i)}, K_3 + R_7^{(i)}, K_2 + R_6^{(i)}, K_1 + R_5^{(i)}) \tag{26}$$
$$u_1 = WD16^{-1}(u_2 \boxminus R_3^{(i)}, K_8 + R_8^{(i)}, K_7 + R_7^{(i)}, K_6 + R_6^{(i)}, K_5 + R_5^{(i)}) \tag{27}$$
$$P_i = WD16^{-1}(u_1 \boxminus R_2^{(i)}, K_4, K_3, K_2, K_1) \boxminus R_1^{(i)}. \tag{28}$$

After this, the internal states are updated as in the encryption, i.e., using Eq. (17)-(24), where $t_3 = u_3 \boxminus R_4^{(i)}$, $t_2 = u_2 \boxminus R_3^{(i)}$ and $t_1 = u_1 \boxminus R_2^{(i)}$.

3 Overview of Our Cryptanalytic Method on the Full HB-2

Adversary Model: We consider a scenario that two paralleled executions of encryptions are $C_i = E(P_i, K)$ and $C'_{i'} = E(P'_{i'}, K')$, where the internal states are $(R_1^{(i)}, ..., R_8^{(i)})$ and $(R'^{(i')}_1, ..., R'^{(i')}_8)$ respectively, the intermediate values are (t_1, t_2, t_3) and (t'_1, t'_2, t'_3) respectively, and K and K' are related. (Similar for the decryption). The attacker follows the chosen plaintext/ciphertext model such that the attacker is free to choose plaintext $P_i \in \mathbb{F}_2^{16}$ and $P'_{i'} \in \mathbb{F}_2^{16}$, launch encryption without knowing the related keys, and observe the corresponding $C_i \in \mathbb{F}_2^{16}$ and $C'_{i'} \in \mathbb{F}_2^{16}$; or chooses $C_i \in \mathbb{F}_2^{16}$ and $C'_{i'} \in \mathbb{F}_2^{16}$, launches decryption without knowing the related keys, and observes the corresponding $P_i \in \mathbb{F}_2^{16}$ and $P'_{i'} \in F_2^{16}$.

Attack In A Nutshell: Block ciphers are usually based on iterating a cryptographically weak function sufficient number of times without disturbing, e.g., modifying, the outputs of intermediate rounds except whitening them with round-keys. Our attack on the full HB-2 exploits the fact that the internal states, which, instead of enhancing the overall cryptanalytic strength, give the attacker an

opportunity to create an input differential for the last invocation of $WD16$ ($WD16^{-1}$ resp.) in the encryption (decryption resp.) and to retrieve the corresponding distribution of the output differences (call the collection of them a *differential sequence*), which is information-rich in (a subset of) $(K_5, ..., K_8)$ ($(K_1, ..., K_4)$ resp.). Henceforth, after obtaining such a *template sequence*, the attacker, in an off-line environment, could search for the key bits associated, which usually constitute a subset of entire key bits. In all, our full attack can be divided into two phases: **preparation phase** as described in Section 5 and **key recovery phase** as described in Section 4.

Key Recovery Phase: In the key recovery phase, to remove the undesired interference introduced by the varying internal states when consecutive words are encrypted/decrypted, our attack here targets a specific encryption/decryption after the *preparation*, i.e., ith encryption/decryption for one HB-2 instance and i'th encryption/decryption for the other one. This is because given the key, IV, and the plaintext chain fed are fixed, the ith internal state and the i'th internal state are fixed as well. Henceforth, we omit the superscript/subscript i and i' of HB-2 variables for convenience when describing operations in the key recovery phase.

Providing the preparation phase succeeds, the attacker accomplishes the following utilizing the properties of the differential sequence analysis:

- Step 1. 36 bits of $(K_5, ..., K_8) \in \mathbb{F}_2^{64}$ are recovered using the differential sequence obtained from the last invocation of $WD16$ in the encryption if a particular condition meets, as shown in Fig. 2.
- Step 2. 28 bits of $(K_4, ..., K_1) \in \mathbb{F}_2^{64}$ are recovered using the differential sequence obtained from the last invocation of $WD16$ in the decryption if another particular condition meets.
- Step 3. the rest 64-bit key are exhaustively searched using either encryption or decryption.

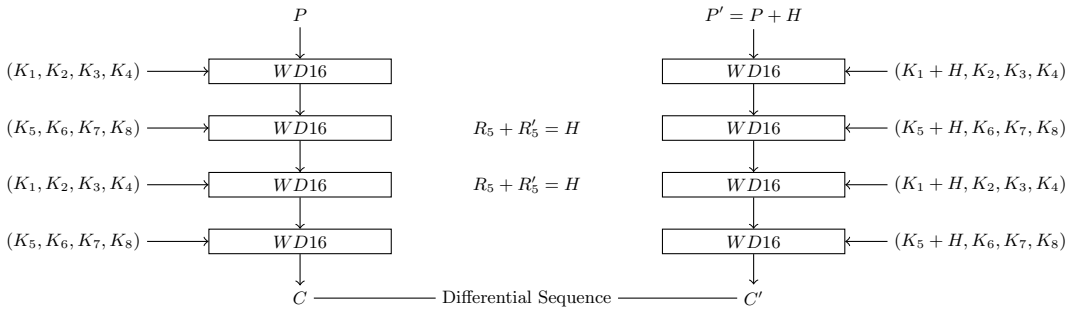

Fig. 2. Constructing Differential Sequence from Encryption with Condition (A)

To be specific, the condition needed to launch Step 1 in *key recovery phase* is:

$$\begin{aligned} \text{Condition (A)}: \quad & \Delta K = (K_1, ..., K_8) + (K'_1, ..., K'_8) = (H, 0, 0, 0, H, 0, 0, 0) \\ & \Delta P = P + P' = H \\ & \Delta R = (R_1, ..., R_8) + (R'_1, ..., R'_8) = (0, 0, 0, 0, H, 0, 0, 0). \end{aligned}$$

The condition needed to launch Step 2 in *key recovery phase* is:

$$\begin{aligned}\text{Condition (B)}: \quad & \Delta K = (K_1, ..., K_8) + (K'_1, ..., K'_8) = (0,0,0,H,0,0,0,H) \\ & \Delta C = C + C' = H \\ & \Delta R = (R_1, ..., R_8) + (R'_1, ..., R'_8) = (0,0,0,0,0,0,0,H).\end{aligned}$$

To reach ΔP or ΔC in the two conditions above, the adversary model already allows the plaintext/ciphertext to be freely chosen; to reach ΔK, two pair of related-keys have to be used in our attack; and to reach ΔR, an extra phase, called preparation phase, has to be introduced.

Preparation Phase: As one may expected, preparation phase of our attack copes with the realization of ΔRs one at a time. To this end, one obvious way is to mount side-channel injection attack as shown in Appendix A, which gives the attacker no time/memory penalty, i.e., the overall time/memory complexity of the attack is dominated by that of the key recovery phase.

However, side-channel injection attack is not considered much in this work. Instead, we realize both conditions in a probabilistic manner, i.e.,

- $(R_1^{(i)}, ..., R_8^{(i)})$ and $(R_1'^{(i')}, ..., R_8'^{(i')})$ can be "randomized" by feeding both HB-2 instances with either different IVs and/or chains of random plaintext words. According to the birthday paradox, there is at least 0.5 chance that the randomized $(R_1^{(i)}, ..., R_8^{(i)}) \in \mathbb{F}_2^{128}$ and the randomized $(R_1'^{(i')}, ..., R_8'^{(i')}) \in \mathbb{F}_2^{128}$ satisfies ΔR in condition (A) (condition (B) resp.) providing 2^{64} attempts are made.
- Note that, in the previous step, even if ΔR happens, the attacker is usually unaware. To determine, we improve the mechanism above in light of another characteristic of HB-2, i.e., if condition (A) (condition (B) resp.) holds at the current round, it also holds for the next round. Hence, the differential sequences produced at the current round by $((R_1^{(i)}, ..., R_8^{(i)}), (R_1'^{(i')}, ..., R_8'^{(i')}))$ is exactly the same as that produced at the next round by $((R_1^{(i+1)}, ..., R_8^{(i+1)}), (R_1'^{(i'+1)}, ..., R_8'^{(i'+1)}))$.
- If the above step succeeds, the attacker proceeds to the key recovery phase to attack.

In what follows, we detail each of the above phases and steps.

4 Differentials Sequence Analysis (DSA)

In this section, we present a novel technique called differential sequence analysis (DSA) rooted in the differential cryptanalysis and the saturation attack. To be specific, we exhibiting its definitions, properties, and applications to attack one round of the HB-2, that constitutes the *key recovery phase* in our whole attack.

4.1 Differential Cryptanalysis and Saturation Attack

Differential cryptanalysis is a method analyzing the effect of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs, which is based on a crucial observation that for any particular input differential, not all the output differential are possible, and the possible ones may

not appear uniformly. In the original version of differential cryptanalysis [37], a unique differential is exploited to recover the subkey used in the last round of a block cipher. This idea has been extended in several ways: Biham and Shamir themselves further considered in [37] to use a trail of differentials to attack; Lai in [27] connected differential cryptanalysis with derivative of polynomials and presented a fine definition of higher order differentials; Knudsen [25] considered to use part of the input and output that have differential characteristics for the analysis; Biham, Biryukov and Shamir proposed in [5] to use differentials that happens with probability 0 as distinguishers; and recently, Blondeau and Gérard demonstrated the multiple differential cryptanalysis in [6], where a set of input/output differentials are considered together.

Saturation attack [23, 28, 8] exploits the fact that the output set is saturated, i.e., the outputs forms the whole space of $\mathbb{F}_2^m$, if the input set for the m-bit core injective function is saturated. Since the saturation of the outputs is observable, this technique usually serves as a distinguisher for the attacker.

At a high level, our differential sequence analysis in this paper can be understood as a hybrid of the conventional differential cryptanalysis and saturation attack, i.e., the set of the output differentials (instead of the outputs themselves) with respect to a particular/fixed input differential and a saturated set of inputs is considered. From another angle, due to the use of output differentials caused by a saturated set of inputs, our attack is also a special case of multiple differential cryptanalysis [6].

4.2 (First-order) Differential Sequence

Assume we have a keyed permutation $h(w, K)$ mapping $w \in \mathbb{F}_2^m$ to $h(w, K) \in \mathbb{F}_2^m$ with respect to the secret key K, where m is a positive integer. Given a fixed $\theta \in \mathbb{F}_2^m$, the first-order differential is known as

$$\Delta_{\theta,K}(w) = h(w, K) + h(w + \theta, K).$$

The *(first-order) differentials sequence* of h at θ is basically one row in the differential distribution table of h with respect to the input differential θ. To discuss its properties, we define it in a more formal way.

Definition 1. *The first-order differential sequence (DS) of h at θ is a non-binary sequence of 2^m entries, i.e.,*

$$\Delta_{\theta,K} = (z_0, z_1, ..., z_{2^m-1}),$$

where z_i denotes the multiplicity (that is, number of occurrences) of i in the set $\{w \in \mathbb{F}_2^m | \Delta_{\theta,K}(w)\}$, i.e.,

$$z_i = |\{w \in \mathbb{F}_2^m |\ \Delta_{\theta,K}(w) = i\}|.$$

Note that this definition can be extended to higher orders. In this paper, we constrained ourself to the first-order case.

For example, the differential sequence is $\{0, 0, 0, 2, 0, 0, 2, 0, 0, 4, 2, 0, 4, 0, 0, 2\}$ providing $\{w = \mathbb{F}_2^4 | \Delta_{\theta,K}(w), \} = \{12, 10, 3, 9, 6, 9, 15, 12, 12, 10, 3, 9, 6, 9, 15, 12\}$ and $\theta =$ 0x08. The length of the differential sequence is the sum of all its multiplicities (16 in this example).

4.3 Properties of the Differential Sequence

The saturated set of inputs brings quite a lot interesting properties to the conventional differential cryptanalysis. We list the core properties to attack HB-2 here.

Property 1. For a fixed $\theta \in \mathbb{F}_2^m$, $\Delta_{\theta,K}$ is constructed by evaluating and counting $(h(w,K)+h(w+\theta,K))$ for every w in $\mathbb{F}_2^m$ regardless of the order of $w \in \mathbb{F}_2^m$ been accessed.

This property follows immediately from Definition 1 and is useful in the sense that even though $h(w,K)$ is an intermediate round in a cipher (thus, w is an intermediate value), we are able to capture $\Delta_{\theta,K}$ given that θ can be fixed in a particular way and w traverses the whole space of $\mathbb{F}_2^m$. Stated in another way, we have the property below.

Property 2. Let $perm(w)$ be a permutation of w in $\mathbb{F}_2^m$, i.e., $perm(w) : \mathbb{F}_2^m \mapsto \mathbb{F}_2^m$. For a fixed $\theta \in \mathbb{F}_2^m$ and every $w \in \mathbb{F}_2^m$, $\Delta_{\theta,K}$ can be obtained either by evaluating and counting $(h(w,K) + h(w+\theta,K))$, or by evaluating and counting $(h(perm(w),K) + h(perm(w)+\theta,K))$.

In what follows, we use

$$[h(w,K) + h(w+\theta,K) | w \in \mathbb{F}_2^m] = [h(perm(w),K) + h(perm(w)+\theta,K) | w \in \mathbb{F}_2^m]$$

as a symbolic expression for Property 2, where [...] actually defines a multiset and θ is always a fixed value in $\mathbb{F}_2^m$ for the rest of the paper. Unsurprisingly, an extension of Property 2 can be derived below.

Property 3. Let $perm_i$, $i = 1, ..., n$, be permutations in $\mathbb{F}_2^m$. We have

$$\begin{aligned} &[h(w,K) + h(w+\theta,K) | w \in \mathbb{F}_2^m] \\ =\ &[h(perm_n(...(perm_1(w))),K) + h(perm_n(...(perm_1(w)))+\theta,K) | w \in \mathbb{F}_2^m]. \end{aligned}$$

Proof. $perm_n(...(perm_1(w)))$ can be written as $perm(w)$ in $\mathbb{F}_2^m$. □

As aforementioned, the obtained differential sequence is primarily used to search for the key bits associated. Henceforth, we are especially interested in the correspondences between the differential sequence and the K in the underlying function $h(w,K)$, e.g., is the mapping from K to the differential sequence injective or not? To this end, we start with a special case of Property 2.

Property 4. Providing $K = K_a \bigcup K_b$, $K_a \bigcap K_b = \emptyset$ and $h(w,K) = h(w+K_a, K_b)$, we have

$$[h(w,K) + h(w+\theta,K) | w \in \mathbb{F}_2^m] = [h((w+K_a),K_b) + h((w+K_a)+\theta,K_b) | w \in \mathbb{F}_2^m].$$

Proof. By applying Property 2 and set $perm(w) = w + K_a$, this property follows immediately. □

From the property above, it is clear that all $K_a \in \mathbb{F}_2^{|K_a|}$ produces the same sequence while different K_bs may produce different sequences. Therefore, this property in fact implies that the obtained differential sequence of h at θ can be used to search for (a subset of) the key nonlinearly associated. To further explore the analytic relationship between the partial key nonlinearly associated and the different sequences, we need to investigate the properties of sub-differential sequences.

Property 5. Let Γ be a subset of $\mathbb{F}_2^m$ and *perm* is a permutation in Γ, we have

$$[h(w,K)+h(w+\theta,K)|w\in\Gamma]=[h(perm(w),K)+h(perm(w)+\theta,K)|w\in\Gamma].$$

Proof. This property follows from Definition 1, if and only if *perm* is a permutation in Γ, i.e., $perm(w):\Gamma\mapsto\Gamma$. We call $[h(w,K)+h(w+\theta,K)|w\in\Gamma]$ or $[h(perm(w),K)+h(perm(w)+\theta,K)|w\in\Gamma]$ a *sub-differential sequence* of $\Delta_{\theta,K}$. □

Due to this, we can actually view a differential sequence obtained in $\mathbb{F}_2^m$ as a summation of (corresponding entries of) several sub-differential sequences obtained in the disjoint subspaces of $\mathbb{F}_2^m$. This intuition can be written as below.

Property 6. Let Γ_i, $i=1,...,q$, be q disjoint partitions of $\mathbb{F}_2^m$, i.e.,

$$\Gamma_i\cap\Gamma_j=\emptyset,\qquad 1\le i\neq j\le q \tag{29}$$
$$\cup_{i=1}^{q}\Gamma_i=\mathbb{F}_2^m \tag{30}$$

and let the differential sequence obtained by $[h(w,K)+h(w+\theta,K)|w\in\Gamma_i]$ be $\Delta_{\theta,K}^{\{\Gamma_i\}}$, we thus have,

$$\Delta_{\theta,K}=\sum_{i=1}^{q}\Delta_{\theta,K}^{\{\Gamma_i\}}.$$

Following this reasoning, Property 4 can also be extended as below, which tells us that a differential sequence in Γ only corresponds to the key nonlinearly (in Γ) associated.

Property 7. Providing $K=K_a\bigcup K_b$, $K_a\bigcap K_b=\emptyset$ and $h(w,K)=h(w+K_a,K_b)$, we have, if Γ is a subset of $\mathbb{F}_2^m$ and $(w+K_a)$ is a permutation in Γ with respect to K_a,

$$[h(w,K)+h(w+\theta,K)|w\in\Gamma]=[h((w+K_a),K_b)+h((w+K_a)+\theta,K_b)|w\in\Gamma].$$

Therefore, if each of the sub-differential sequence stays the same with respect to the keys belonging to a particular set, denoted as Φ_0, the overall differential sequence remain the same under Φ_0. We formalize this correspondence as below.

Property 8. Let $\Phi_0=\cap_{i=1}^{q}\{k|w+k:\Gamma_i\mapsto\Gamma_i,k,w\in\mathbb{F}_2^m\}$, $K=K_a\bigcup K_b$, $K_a\bigcap K_b=\emptyset$ and $h(w,K)=h(w+K_a,K_b)$, we have

$$\Delta_{\theta,K}=\Delta_{\theta,\kappa}, \tag{31}$$

where $\kappa=\kappa_a\bigcup\kappa_b$, $\kappa_a\bigcap\kappa_b=\emptyset$, $K_a,\kappa_a\in\Phi_0$ and $\kappa_b=K_b$.

Proof. Let $\Delta_{\theta,K}^{\{\Gamma_i\}}$ be the sub-differential sequence obtained by Property 7. Thanks to Property 6, we have $\Delta_{\theta,K}=\sum_{i=1}^{q}\Delta_{\theta,K}^{\{\Gamma_i\}}$ and $\Delta_{\theta,\kappa}=\sum_{i=1}^{q}\Delta_{\theta,\kappa}^{\{\Gamma_i\}}$. Thanks to Property 7, for each i , $\Delta_{\theta,K}^{\{\Gamma_i\}}=\Delta_{\theta,\kappa}^{\{\Gamma_i\}}$ providing $K_a,\kappa_a\in\Phi_0$ and $\kappa_b=K_b$.

As opposed, providing $\kappa_b\neq K_b$ while $K_a,\kappa_a\in\Phi_0$, it is quite likely that $\Delta_{\theta,K}\neq\Delta_{\theta,\kappa}$ since $\Delta_{\theta,K}^{\{\Gamma_i\}}\neq\Delta_{\theta,\kappa}^{\{\Gamma_i\}}$ for each i.

□

4.4 Differential Sequence Analysis against HB-2

In this subsection, we attack the last invocation of $WD16$ ($WD16^{-1}$ resp.) in the encryption (decryption resp.) of HB-2 by exploiting the DSA as presented. To be specific, our Theorem 1 and Theorem 3 give answers to the question "how to obtain the differential sequences" while our Theorem 2 and Theorem 4 exhibit "how to use the differentials sequences". Since the HB-2 has a 16-bit block size, we have $m = 16$ for the rest.

Attacking $WD16$ in Encryption: To show our idea in a concise way, we assume that R_1 and R'_1 are known (in fact, as they are identified by our algorithms in the preparation phase). In addition, let h in Definition 1 be the last invocation of $WD16$, i.e., Eq. (16), in the encryption. We thus have the following theorems.

Theorem 1. *When condition (A) meets, the differential sequence of the last $WD16$ in the encryption at $\theta = H$ can be extracted from executing the entire encryption.*

Proof: First of all, when condition (A) holds, we have,

$$\begin{aligned}
t'_1 &= WD16(R'_1 \boxplus P', K'_1, K'_2, K'_3, K'_4) \\
&= WD16(R_1 \boxplus (P + H), (K_1 + H), K_2, K_3, K_4) = t_1 \\
t'_2 &= WD16(R'_2 \boxplus t'_1, K'_5 + R'_5, K'_6 + R'_6, K'_7 + R'_7, K'_8 + R'_8) \\
&= WD16(R_2 \boxplus t_1, (K_5 + H) + (R_5 + H), K_6 + R_6, K_7 + R_7, \\
&\quad K_8 + R_8) = t_2 \\
t'_3 &= WD16(R'_3 \boxplus t'_2, K'_1 + R'_5, K'_2 + R'_6, K'_3 + R'_7, K'_4 + R'_8) \\
&= WD16(R_3 \boxplus t_2, (K_1 + H) + (R_5 + H), K_2 + R_6, K_3 + R_7, \\
&\quad K_4 + R_8) = t_3
\end{aligned}$$

Next, $\Delta_{H,(K_5,K_6,K_7,K_8)} = [z_0, z_1, ..., z_{2^{16}-1}]$ can be extracted, where

$$\begin{aligned}
z_i &= |\{t_3 \in \mathbb{F}_2^{16}|\ (WD16(R_4 \boxplus t_3, K_5, K_6, K_7, K_8) + WD16(R'_4 \boxplus t'_3, K'_5, K'_6, K'_7, K'_8)) = i\}| \\
&= |\{t_3 \in \mathbb{F}_2^{16}|\ (WD16(R_4 \boxplus t_3, K_5, K_6, K_7, K_8) + WD16(R_4 \boxplus t_3, (K_5 + H), K_6, K_7, K_8)) = i\}| \\
&= |\{t_3 \in \mathbb{F}_2^{16}|\ (WD16(R_4 \boxplus t_3, K_5, K_6, K_7, K_8) + WD16((R_4 + H) \boxplus t_3, K_5, K_6, K_7, K_8)) = i\}| \\
&= |\{P \in \mathbb{F}_2^{16}, P' = P + H|\ (C \boxminus R_1) + (C' \boxminus R_1) = i\}|.
\end{aligned}$$

The second last equality comes from the fact

$$(R_4 \boxplus t_3) + (K_5 + H) = ((R_4 + H) \boxplus t_3) + K_5.$$

Note that condition (A) is essentially a necessary condition for the following condition:

$$\begin{aligned}
\Delta K &= (K_1, ..., K_8) + (K'_1, ..., K'_8) = (0, 0, 0, 0, 0, 0, 0, 0) \\
\Delta P &= P_1 + P'_{i'} = 0 \\
\Delta R &= (R_1, ..., R_8) + (R'_1, ..., R'_8) = (0, 0, 0, H, 0, 0, 0, 0),
\end{aligned}$$

such that both of them produce the same differential sequence of $WD16$. However, we use condition (A) through the rest of the paper because it has an additional property that keeps the attacker informed once ΔR happens (see Section 5.2). □

This theorem suggests that, after querying the encryption with every $P \in \mathbb{F}_2^{16}$ and obtaining the resultant output differentials, the attacker could have a *template sequence* $\Delta_{H,(K_5,K_6,K_7,K_8)}$ to search for parts of (K_5, K_6, K_7, K_8). The next theorem discloses the correspondence between $\Delta_{H,(K_5,K_6,K_7,K_8)}$ and (K_5, K_6, K_7, K_8).

Theorem 2. *Let $\Delta_{H,(K_5,K_6,K_7,K_8)}$ be obtained from Theorem 1. For $\kappa_5 \in \mathbb{F}_2^{16}$ and $\kappa_6 \in \mathbb{F}_2^{16}$, we have*

$$\Delta_{H,(K_5,K_6,K_7,K_8)} = \Delta_{H,(\kappa_5,\kappa_6,K_7,K_8)},$$

where K_6 and κ_6 belong to the same set $\Phi_i = \Phi_0 + i$, $0 \le i \le 15$ of cardinality 2^{12}.

Proof: To prove, we discuss the correspondence between K_5, K_6, K_7, K_8 and the template sequence in a respective way.

Correspondence Between K_5 and DS: For the time being, let us consider $h(w, K) = f(f(w+K_5)+K_6)$ (a simplified $WD16$), where $f : \mathbb{F}_2^{16} \mapsto \mathbb{F}_2^{16}$ (as described in Section 2) is an injective function, we thus have, by letting $w = R_4 \boxplus t_3$ and $\theta = H$,

$$\begin{aligned}
&[h(w, K) + h(w + \theta, K) | w \in \mathbb{F}_2^{16}] \\
&= [f(f(w + K_5) + K_6) + f(f(w + K_5 + \theta) + K_6) | w \in \mathbb{F}_2^{16}] \\
&= [f(f(perm_1(w)) + K_6) + f(f(perm_1(w) + \theta) + K_6) | w \in \mathbb{F}_2^{16}]
\end{aligned}$$

It is clear from the context that $perm_1(w) = w + K_5$ is a permutation in $\mathbb{F}_2^m$, and, due to Property 4, $\Delta_{H,(K_5,K_6,K_7,K_8)}$ does not dependent on K_5.

Correspondence Between K_6 and DS: We define the following auxiliary variables for convenience:

- λ_i, $i = 1, ..., q$, are q possible output differences of f, given the input difference is θ.
- $\Gamma_i = \{f(w) | f(w) + f(w + \theta) = \lambda_i, w \in \mathbb{F}_2^{16}\}$, $i = 1, ..., q$, are q disjoint partitions of $\mathbb{F}_2^{16}$ such that: (1) Eq. (29) holds, otherwise there is a $w \in (\Gamma_i \cap \Gamma_j)$, $1 \le i \ne j \le q$, such that $f(w) + f(w + \theta)$ produces output differences λ_i and λ_j, $\lambda_i \ne \lambda_j$, which is impossible; (2) Eq. (30) holds, otherwise there is a $w \in (\mathbb{F}_2^{16} - \cup_{i=1}^{q} \Gamma_i)$, that produces an output difference $\notin \{\lambda_1, ..., \lambda_q\}$, which contradicts our definition.
- $\Phi_0 = \cap_{i=1}^{q} \{k | f(w) + k : \Gamma_i \mapsto \Gamma_i, k \in \mathbb{F}_2^{16}\}$. Intuitively, Φ_0 encompasses all possible keys, which make $f(w) + k$ a permutation in Γ_i, $i = 1, ..., q$.

Furthermore, let us consider two cases: (1) $K_6 \in \Phi_0$; and (2) $K_6 \in$ a coset of Φ_0, i.e., $\Phi_i = \Phi_0 + i$, $0 < i \le 15$.

For case (1), the above equations can be further written as, by setting $perm_2(w) = f(perm_1(w)) + K_6$,

$$\begin{aligned}
&[f(f(perm_1(w)) + K_6) + f(f(perm_1(w) + \theta) + K_6) | w \in \mathbb{F}_2^{16}] \\
&= [f(f(perm_1(w)) + K_6) + f(f(perm_1(w)) + \lambda_i + K_6) | w \in \Gamma_i] && \text{for } i = 1, ..., q \\
&= [f(f(perm_1(w)) + K_6) + f(f(perm_1(w)) + K_6 + \lambda_i) | w \in \Gamma_i] && \text{for } i = 1, ..., q \\
&= [f(perm_2(w)) + f(perm_2(w) + \lambda_i) | w \in \Gamma_i] && \text{for } i = 1, ..., q \\
&= [f(w) + f(w + \lambda_i) | w \in \Gamma_i] && \text{for } i = 1, ..., q
\end{aligned}$$

The above equation holds because of Property 8, e.g., for $K_6 \in \Phi_0$, every $[f(w)+f(w+\lambda_i)|w \in \Gamma_i]$ produces the same sub-differential sequence. Therefore, the overall differential sequence stays the same for every $K_6 \in \Phi_0$. Stating in another way, providing K_6 and κ_6 are both in Φ_0, $\Delta_{H,(K_5,K_6)} = \Delta_{H,(\kappa_5,\kappa_6)}$.

The above derivation is further confirmed through extensive experiments, where we found $(\lambda_1, ..., \lambda_6) = (0x30cc, 0x6198, 0x9264, 0xa2a8, 0xc330, 0xf3fc)$, $(\Gamma_1, ...\Gamma_6)$, and Φ_0 of cardinality 2^{12}.

Moreover, case (2) can be proved. This is because, by letting $K_6 = \triangleright K_6 + \triangleleft K_6$ such that $\triangleright K_6 \in \Phi_0$,

$$\begin{aligned}
&[f(f(perm_1(w)) + K_6) + f(f(perm_1(w)) + K_6 + \lambda_i)|w \in \Gamma_i] && \text{for } i = 1, ..., q\\
&= [f(f(perm_1(w)) + \triangleright K_6 + \triangleleft K_6) + f(f(perm_1(w)) + \triangleright K_6 + \triangleleft K_6 + \lambda_i)|w \in \Gamma_i] && \text{for } i = 1, ..., q\\
&= [f(perm_3(w) + \triangleleft K_6) + f(perm_3(w) + \triangleleft K_6 + \lambda_i)|w \in \Gamma_i] && \text{for } i = 1, ..., q\\
&= [f(w + \triangleleft K_6) + f(w + \triangleleft K_6 + \lambda_i)|w \in \Gamma_i] && \text{for } i = 1, ..., q
\end{aligned}$$

The second last equation holds because of our proof of case (1) (by letting $perm_3(w) = f(perm_1(w)) + \triangleright K_6$).

In addition, it is clear that:

- the sub-differential sequence $[f(w + \triangleleft K_6) + f(w + \triangleleft K_6 + \lambda_i)|w \in \Gamma_i]$, is different from $[f(w + f(w + \lambda_i)|w \in \Gamma_i]$ as long as $\triangleleft K_6 \neq 0$. So is the overall differential sequence with overwhelming probability.
- for $K_6 = \triangleright K_6 + \triangleleft K_6$, $\kappa_6 = \triangleright \kappa_6 + \triangleleft \kappa_6$, $K_6 \neq \kappa_6$, the sub-differential sequence $[f(w + \triangleleft K_6) + f(w + \triangleleft K_6 + \lambda_i)|w \in \Gamma_i]$, is the same as $[f(w + \triangleleft \kappa_6) + f(w + \triangleleft \kappa_6 + \lambda_i)|w \in \Gamma_i]$ as long as $\triangleleft K_6 = \triangleleft \kappa_6$. This is due to the possibility that $\triangleright K_6, \triangleright \kappa_6 \in \Phi_0$, $\triangleright K_6 \neq \triangleright \kappa_6$ could yield $\triangleleft K_6 = \triangleleft \kappa_6$.

From the accusation above and our extensive experiments, it can be concluded that the key space of $K_6 \in \mathbb{F}_2^{16}$ has been divided into 16 cosets, i.e., $\Phi_0, ..., \Phi_{15}$, and each is of cardinality 2^{12}.

Correspondence Between (K_7, K_8) and DS: We carry on all the notations above for K_7 except setting $h(w, K) = f(f(f(f(w + K_5) + K_6) + K_7) + K_8)$. We found that, for K_7, Φ_0 is always a empty set because too many λ_i divides $\mathbb{F}_2^{16}$ into numerous tiny subspaces Γ_i, for which there is no K_7 could make $f(w) + K_7$ a permutation in every Γ_i, $i = 1, ..., q$. Same phenomenon happens to K_8. In all, each choice of (K_7, K_8) produces a different differential sequence, which is further confirmed empirically. □

Attacking $WD16^{-1}$ in Decryption: Similar attack can be performed against the decryption. By assuming R_1 and R'_1 are known and letting h in Definition 1 be the last invocation of $WD16^{-1}$, i.e., Eq. (28), we have the following results for our attack.

Theorem 3. *With the condition (B), the differential sequence of the last $WD16^{-1}$ in the decryption at $\theta = H$ can be extracted from executing the entire decryption.*

Proof: First of all, when condition (B) holds, we have,

$$\begin{aligned}
u_3 &= WD16^{-1}(C \boxminus R_1, K_8, K_7, K_6, K_5)\\
&= WD16^{-1}((C + H) \boxminus R'_1, (K_8 + H), K'_7, K'_6, K'_5) = u'_3\\
u_2 &= WD16^{-1}(u_3 \boxminus R_4, K_4 + R_8, K_3 + R_7, K_2 + R_6, K_1 + R_5)\\
&= WD16^{-1}(u'_3 \boxminus R'_4, (K_4 + H) + (R_8 + H), K'_3 + R'_7, K'_2 + R'_6, K'_1 + R'_5) = u'_2\\
u_1 &= WD16^{-1}(u_2 \boxminus R_3, K_8 + R_8, K_7 + R_7, K_6 + R_6, K_5 + R_5)\\
&= WD16^{-1}(u'_2 \boxminus R'_3, (K_8 + H) + (R_8 + H), K'_7 + R'_7, K'_6 + R'_6, K'_5 + R'_5) = u'_1
\end{aligned}$$

Next, $\Delta_{H,(K_4,K_3,K_2,K_1)} = [z_0, z_1, ..., z_{2^{16}-1}]$ can be extracted, where,

$$\begin{aligned} z_i &= |\{u_1 \in \mathbb{F}_2^{16}|\ (WD16^{-1}(u_1 \boxminus R_2, K_4, K_3, K_2, K_1) + WD16^{-1}(u_1' \boxminus R_2', K_4', K_3', K_2', K_1')) = i\}| \\ &= |\{u_1 \in \mathbb{F}_2^{16}|\ (WD16^{-1}(u_1 \boxminus R_2, K_4, K_3, K_2, K_1) + WD16^{-1}(u_1' \boxminus R_2', K_4 + H, K_3, K_2, K_1)) = i\}| \\ &= |\{C \in \mathbb{F}_2^{16}, C' = C + H|\ (P \boxplus R_1) + (P' \boxplus R_1) = i\}|. \end{aligned}$$

□

A similar theorem describes the correspondence between $\Delta_{H,(K_4,K_3,K_2,K_1)}$ and (K_4, K_3, K_2, K_1).

Theorem 4. *Let $\Delta_{H,(K_4,K_3,K_2,K_1)}$ be obtained from Theorem 3. For $\kappa_1 \in \mathbb{F}_2^{16}$ and $\kappa_4 \in \mathbb{F}_2^{16}$,*

$$\Delta_{H,(K_4,K_3,K_2,K_1)} = \Delta_{H,(\kappa_4,K_3,K_2,\kappa_1)},$$

where K_4 and κ_4 belong to the same set $\Phi_i = \Phi_0 + i$, $0 \le i \le 2^{12}-1$, and $\Phi_0 = \{0x0000, 0x0010, ..., 0x00f0\}$.

Proof: Similar as Theorem 2, except that we could easily observe from the experimental data that $\Phi_0 = \{0x0000, 0x0010, 0x0020, ..., 0x00f0\}$. □

4.5 Local Search in DSA

After the template sequence is extracted, the attacker could, in an off-line environment, launches $h(w, K) = WD16(.)$ ($h(w, K) = WD16^{-1}(.)$ resp.) to search for parts of (K_5, K_6, K_7, K_8) ($(K_1, K_2, K_3, K_4$ resp.), which is called the *local search* in DSA. Through the local search, the attacker recovers 36-bit (44-bit resp.) information regarding the key.

A naive way to search locally is producing a complete local differential sequence from $[h(w, K) + h(w + H, K)|w \in \mathbb{F}_2^{16}]$ with a random K, comparing each entry of which with the corresponding entry of the template sequence. The cost per key trial is 2^{16} executions of $h(w, K)$s and 2^{16} comparisons.

The efficiency of this method can be substantially improved if the early-abort strategy [30] is adapted, i.e., given the ith entry in the local differential sequence is greater than the ith entry in the template sequence, one could assert that the trial key is incorrect and terminate the search in advance. We present this improved local search algorithm as below.

1: let TDS be the template sequence obtained
2: initiate the local differential sequence LDS as a list of 2^{16} "0"s
3: **for** w from 0 to $2^{16} - 1$ **do**
4: randomly choose K
5: $diff \leftarrow h(w + K) + h(w + H + K)$
6: $LDS[diff] \leftarrow LDS[diff] + 1$
7: **if** $LDS[diff] > TDS[diff]$ **then**
8: return NULL
9: **end if**
10: **end for**
11: **if** $LDS[w] = TDS[w]$ for $w = 0, 1, ..., 2^{16} - 1$ **then**
12: return K
13: **end if**

The theoretical derivation of the time complexity of the above algorithm could be quite cumbersome. Instead, we recorded the number of the for-loops that are actually executed, denoted as l, during the search. Through repeated testings, we found that, in average, $1.640 < l < 1.660$ for-loops are spent per key trial for both local searches using $WD16(.)$ and $WD16^{-1}(.)$. Thus, we conclude the cost per key trial of our local search algorithm is 1.65 executions of (a pair of) $h(w, K)$s .

4.6 Differential Sequence Analysis (DSA) Against HB-2 and Its Time Complexity

We are ready to list out the steps performed by the attacker during the key recovery phase, as below.

1. When condition (A) holds, the attacker extracts the template sequence $\Delta_{H,(K_5,K_6,K_7,K_8)}$ using $((C \boxminus R_1) + (C' \boxminus R_1))$, where C and C' can be obtained by querying the encryption with P and $P' = P + H$, and R_1 and R'_1 are obtained in the preparation phase. Then, the attacker locally searches 36-bit of (K_5, K_6, K_7, K_8) using the proposed local search algorithm.
2. Similarly, utilizing the decryption, when condition (B) holds, the attacker extracts another template sequence $\Delta_{H,(K_4,K_3,K_2,K_1)}$ using $(P \boxplus R_1) + (P' \boxplus R_1)$, and guesses to determine 44-bit of (K_4, K_3, K_2, K_1) using the proposed local search algorithm.
3. After that, the attacker searches the remaining 48-bit of the key using 2^{48} trial encryptions[1].

The overall complexity of the above steps is

$$\underbrace{2^{36} \times 1.65}_{\text{determine 36-bit of } (K_5,...,K_8)} + \underbrace{2^{44} \times 1.65}_{\text{determine 44-bit of } (K_1,...,K_4)} + \underbrace{2^{48}}_{\text{determine the rest}} \approx 2^{48.14},$$

where negligible memory is required by each steps.

5 A Probabilistic Realization of Conditions (A) and (B)

The attacks in the last section solely depends on the occurrences of conditions (A) and (B), to reach ΔRs in which sounds unpractical at the first glance as the initialization of HB-2 makes the internal states unpredictable. In this section, we show a probabilistic approach to realize these conditions – when the internal states of two HB-2 instances are respectively random, there is a certain chance that the attacker could get the desired differentials in the internal states. To this end, we study how to randomize the internal states of HB-2 at first, and, how to determine whether the desired ΔRs happen.

5.1 Randomize the Internal States

There are two ways for the adversary to affect the internal states of HB-2:

– Providing the key is fixed, it is suffice, from Eq. (1)-(12), that $(IV_1, ..., IV_4) \mapsto (R_1, ..., R_4)$ is an injective mapping and so is $(IV_1, ..., IV_4) \mapsto (R_5, ..., R_8)$. Therefore, the attacker could easily generate 2^{64} (out of 2^{128}) different internal states by choosing different IVs and launching the initialization.

[1] In fact, $2^{48} + 2^{32} + 2^{16} = 2^{48.00002201}$ trial encryptions and three plaintext-ciphertext pairs are required.

– For a fixed key and a particular IV, the attacker could choose plaintext P_1 to feed HB-2 at first. If a state transition graph is drawn, we can see that the starting state, i.e., $R^{(1)}$, transits to 2^{16} neighboring states while each $P_1 \in \mathbb{F}_2^{16}$ is encrypted. Next, if another encryption is performed, e.g., encrypting P_2, each of these "neighboring states" again transits to another 2^{16} states providing P_2 takes every value in $\mathbb{F}_2^{16}$. By continuing this process, we would have all 2^{128} states covered in this graph. Therefore, to produce a set of random internal states, i.e., $\{R^{(1)}, R^{(2)}, ...\}$, we could, as shown in Fig. 3, feed the encryptions with a plaintext chain where P_i is selected uniformly at random in $\mathbb{F}_2^{16}$ for $i = 1, 2, ...$ Similarly, a ciphertext chain could be fed to the decryption oracle to generate a set of random internal states as well. Note that feeding HB-2 encryption with a chain of N random inputs is equivalent to perform an N-step 2^{16}-dimensional random walk in its state transition graph. Therefore, $|\{R^{(1)}, R^{(2)}, ...\}| \approx N$ if$N \ll 2^{128}$ [15].

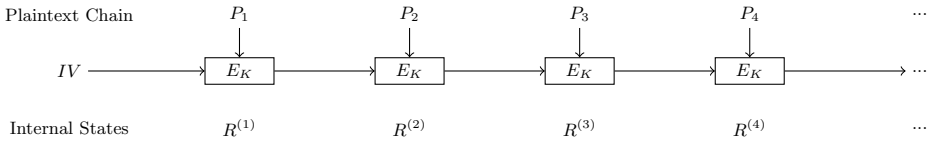

Fig. 3. Feeding HB-2 Encryption with a Plaintext Chain

Therefore, the algorithm below provides, to the later steps, the randomized internal states of two running HB-2 instances through an effort-saving way – one instance initializes a random IV and encrypts one random plaintext, while the other one, besides initializes a random IV, encrypts N random plaintexts consecutively. Since $\{R^{(1)}, R^{(2)}, ..., R^{(N)}\}$ is a set of random variables as analyzed, $\{R^{(1)} + R'^{(1)}, R^{(2)} + R'^{(1)}, ..., R^{(N)} + R'^{(1)}\}$ must also be random variables.

1: Let $R^{(i)} \Leftarrow E(P_i, K)$ be the internal state $R^{(i)}$ after encrypting $P_1, ..., P_i$
2: Randomly choose IV' and P'_1, $R'^{(1)} \Leftarrow E(P'_1, K)$
3: Randomly choose IV
4: **for** i from 1 to N **do**
5: Randomly choose P_i, $R^{(i)} \Leftarrow E(P_i, K)$
6: **if** $R'^{(1)} + R^{(i)} = \Delta R$ **then**
7: return "ΔR happens"
8: **end if**
9: **end for**

Note that, currently, the given algorithm is only a skeleton for our attack, which is discussed in more detail in the next subsections and the full-fledged version is given at last. Nevertheless, one can already sense an interesting property from this skeleton algorithm.

Property 9. In the algorithm above, a certain ΔR happens with 0.5 probability when $N = 2^{64}$.

Proof. This property holds due to the birthday paradox. □

5.2 Determine while Guessing

To inform the attacker during the attempting, as long as condition (A) (condition (B) resp.) happens, we use one unusual differential characteristic in the encryption (decryption resp.), as first pointed out by HB-2's designers, such that the differentials in the internal states, secret keys and the inputs can be maintained and entered into the next round, i.e., for a positive integer i,

$$(\Delta P_i, \Delta K, \Delta R^{(i)}) = (\Delta P_{i+1}, \Delta K, \Delta R^{(i+1)}).$$

Therefore, the following theorem holds.

Theorem 5. *Let $\Delta^{(i')}_{H,(K_5,K_6,K_7,K_8)}$ ($\Delta^{(i')}_{H,(K_4,K_3,K_2,K_1)}$ resp.) be the differential sequence produced by the two encryption instances (two decryption instances resp.) with internal states $R^{(1)}$ and $R'^{(i')}$ and let $\Delta^{(i'+1)}_{H,(K_5,K_6,K_7,K_8)}$ ($\Delta^{(i'+1)}_{H,(K_4,K_3,K_2,K_1)}$ resp.) be the differential sequence produced by the two encryption instances (two decryption instances resp.) with internal states $R^{(2)}$ and $R'^{(i'+1)}$ (call $\Delta^{(i')}_{H,K}$ and $\Delta^{(i'+1)}_{H,K}$ neighboring template sequences). Therefore,*

- *If condition (A) happens during encryption, the adversary observes two identical neighboring template sequences, i.e.,*
$$\Delta^{(i')}_{H,(K_5,K_6,K_7,K_8)} = \Delta^{(i'+1)}_{H,(K_5,K_6,K_7,K_8)};$$
otherwise, the above equation holds with negligible probability.
- *If condition (B) happens during decryption, the adversary observes two identical neighboring template sequences, i.e.,*
$$\Delta^{(i')}_{H,(K_4,K_3,K_2,K_1)} = \Delta^{(i'+1)}_{H,(K_4,K_3,K_2,K_1)};$$
otherwise, the above equation hold with negligible probability.

Proof: It follows from Definition 1 and Property 1. □

Therefore, the above theorem can serve as an algorithm to determine the occurrences of condition (A) or condition (B), i.e., it returns either $(Success, \Delta^{(i')}_{H,K}, R^{(1)}_1, R^{(2)}_1)$ or $(False, NULL, NULL, NULL)$ to the key recovery phase. Unfortunately, in this algorithm, the correct template sequences can only be extracted with the correct $R^{(1)}_1$ and $R^{(2)}_1$ due to Theorem 1 and Theorem 3. For instance, using the encryption, the two neighboring sequences are

$$\Delta^{(i')} = (z^{(i')}_0, z^{(i')}_1, ..., z^{(i')}_{65535}) \tag{32}$$
$$\Delta^{(i'+1)} = (z^{(i'+1)}_0, z^{(i'+1)}_1, ..., z^{(i'+1)}_{65535}) \tag{33}$$

where

$$z^{(i')}_j = |\{P_1 \in \mathbb{F}_2^{16}, P'_{i'} = P_1 + H|\ (C_1 \boxminus R^{(1)}_1) + (C'_{i'} \boxminus R'^{(i')}_1) = j\}| \qquad \text{and}$$
$$z^{(i'+1)}_j = |\{P_2 \in \mathbb{F}_2^{16}, P'_{i'+1} = P_2 + H|\ (C_2 \boxminus R^{(2)}_1) + (C'_{i'+1} \boxminus R'^{(i'+1)}_1) = j\}|$$

Henceforth, it is true that by guessing $R_1^{(1)}$ and $R_1^{(2)}$, the theorem/algorihtm above would cost 2^{32} encryptions/decryptions per execution.

To improve its efficiency, we make use of the following fact: as the modulo addition is only first-order correlation-immune, the two identical neighboring sequences obfuscated by modulo additions of different R_1s may have an apparent correlation, while two distinct neighboring sequences may not. This intuition is further verified by our extensive experiments. In parallel with Eq. (32) and Eq. (33), let us define the *raw neighboring sequences* as:

$$\underline{\Delta^{(i')}} = (\underline{z_0^{(i')}}, \underline{z_1^{(i')}}, ..., \underline{z_{65535}^{(i')}})$$
$$\underline{\Delta^{(i'+1)}} = (\underline{z_0^{(i'+1)}}, \underline{z_1^{(i'+1)}}, ..., \underline{z_{65535}^{(i'+1)}})$$

where

$$\underline{z_j^{(i')}} = |\{P_1 \in \mathbb{F}_2^{16}, P'_{i'} = P_1 + H |\ C_1 + C'_{i'} = j\}| \qquad \text{and}$$
$$\underline{z_j^{(i'+1)}} = |\{P_2 \in \mathbb{F}_2^{16}, P'_{i'+1} = P_2 + H |\ C_2 + C'_{i'+1} = j\}|.$$

We found that, for the identical neighboring sequences, the corresponding two raw neighboring sequences always have more than 30000 (out of 65536) identical entries, i.e.,

$$Corr(\underline{\Delta^{(i')}}, \underline{\Delta^{(i'+1)}}) = |\{\underline{z_j^{(i')}} = \underline{z_j^{(i'+1)}}, j = 0, 1, ..., 65535\}| > 30000, \text{ iff } \Delta^{(i')} = \Delta^{(i'+1)},$$

where $Corr(.,.)$ is the non-normalized correlation.

On the contrary, for the distinct neighboring sequences, the corresponding two raw neighboring sequences always have less than 19000 (out of 65536) identical entries, i.e.,

$$Corr(\underline{\Delta^{(i')}}, \underline{\Delta^{(i'+1)}}) = |\{\underline{z_j^{(i')}} = \underline{z_j^{(i'+1)}}, j = 0, 1, ..., 65535\}| < 19000, \text{ iff } \Delta^{(i')} \neq \Delta^{(i'+1)}.$$

By treating the correlation of the raw neighboring sequences as a criterion, Theorem 5 is now able to return whether $\Delta^{(i')}$ equals $\Delta^{(i'+1)}$ with 2^{16} time complexity. Once the identical neighboring sequences are identified, the adversary is able to guess to recover $R_1^{(1)}$ and $R_1^{(2)}$ with 2^{32} effort in time.

5.3 Preparation Phase and Its Time Complexity

We recap the whole process in the preparation phase for the encryption as shown below, which is an extension of the skeleton algorithm we shown before. Note that the preparation using the decryption is similar and omitted here.

randomly choose IV' and P'_1, $R'^{(1)} \Leftarrow E(P'_1, K)$
randomly choose IV
randomly choose a constant P'_2
for i from 1 to $N = 2^{64}$ **do**
  randomly choose P_i, $R^{(i)} \Leftarrow E(P_i, K)$
  generate $\Delta^{(i)}$ using $R'^{(1)}$ and $R^{(i)}$
  $R'^{(2)} \Leftarrow E(P'_2, K)$
  $R^{(i+1)} \Leftarrow E(P_{i+1}, K)$ where $P_{i+1} = P'_2 + H$
  generate $\Delta^{(i+1)}$ using $R'^{(2)}$ and $R^{(i+1)}$
  if $Corr(\Delta^{(i)}, \Delta^{(i+1)}) > 30000$ **then**
    guess to determine $R_1^{(1)}$ and $R_1^{(2)}$
    recover $\Delta_{H,K}^{(i')}$ from the raw neighboring sequences
    return $(Success, \Delta_{H,K}^{(i')}, R_1^{(1)}, R_1^{(2)})$, keep current states and enter the key recovery phase
  end if
  decrypt using C'_2 and C_{i+1} to roll back HB-2's states to $R'^{(1)}$ and $R^{(i)}$
end for
return $(False, NULL, NULL, NULL)$

Using the encryption (decryption resp.) only, the attacker has 0.5 probability to reach condition (A) (condition (B) resp.) with $2^{64} \times 2^{16} = 2^{80}$ time complexity. After that, he is able to guess to determine $R_1^{(1)}$ and $R_1^{(2)}$ with additional 2^{32} effort in time. In all, the time complexity of the preparation phase is

$$\underbrace{2^{64} \times 2^{16}}_{\text{test whether the condition happens}} + \underbrace{2^{32}}_{\text{guess to determine } R_1\text{s}} + \underbrace{2^{16}}_{\text{recover the template seuqnece}} \approx 2^{80}.$$

It is worthy to mention that to succeed with probability 1, the preparation phase requires $2^{128+16} = 2^{144}$ effort in time, which is slower than the exhaustive search.

6 Concluding Remarks

In this paper, we present a novel cryptanalytic technique called differential sequence analysis (DSA), which is especially effective if the differential sequence reflecting parts of a cipher associated with parts of the key can be obtained. In addition, we demonstrate the application of this technique, that constitutes the key recovery of the lightweight block cipher Hummingbird-2 with $2^{48.14}$ time complexity, given particular conditions hold in its internal states, secret keys and the inputs. Furthermore, we investigate how to reach these conditions in our preparation phase with 0.5 chance and 2^{80} effort in time. To the best of our knowledge, this is the first cryptanalytic result of the full Hummingbird-2.

The attack presented against Hummingbird-2 is a special case of the general DSA, to build the theoretic framework of which is part of our future work. In addition, it will be evaluated in the recent future: (1) whether the generalized DSA provides even better results against Hummingbird-2 and other potentially vulnerable ciphers, especially the ones with small block size and with internal states, e.g., stateful block ciphers [24]; (2) the possibility that the generalized DSA can work with other cryptanalysis technologies, e.g., meet-in-the-middle.

References

1. J.P. Aumasson, M. Naya-Plasencia and M.J.O. Saarinen, Practical attack on 8 rounds of the lightweight block cipher KLEIN, to appear in Proceedings of *INDOCRYPT'11*, pp.1–13, 2011.
2. M. Abdelraheem, G. Leander and E. Zenner, Differential cryptanalysis of round-reduced PRINTcipher: computing roots of permutations, *Fast Software Encryption, FSE'11*, LNCS 6733, pp. 1–17, 2011.
3. M. Ågren, Some instant-and practical-time related-key attacks on KTANTAN32/48/64, to appear in Proceedings of *Selected Areas in Cryptography, SAC'11*, pp. 1–17, 2011.
4. A. Bogdanov and C. Rechberger, A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN, *Selected Areas in Cryptography*, LNCS 6544, pp. 229–240, 2011.
5. E. Biham, A. Biryukov and A. Shamir, Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials, *Journal Of Cryptology*, vol. 18, no. 4, pp. 291–311, 1999.
6. C. Blondeau and B. Gérard, Multiple differential cryptanalysis: theory and practice, *Fast Software Encryption, FSE'11*, LNCS 6733, pp. 35-54, 2011.
7. A. Bogdanov and C. Rechberger, A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN, *Selected Areas in Cryptography, SAC'10*, LNCS 6544, pp. 229–240, 2010.
8. A. Biryukov and A. Shamir, Structural cryptanalysis of SASAS, *Journal of cryptology*, vol. 23, no. 4, pp. 505–518, 2010.
9. A. Biryukov, I. Kizhvatov and B. Zhang, Cryptanalysis of the Atmel cipher in SecureMemory, CryptoMemory and CryptoRF, *Applied Cryptography and Network Security, ACNS'11*, LNCS 6715, pp. 91–109, 2011.
10. Q. Chai, Design and analysis of security schemes for low-cost RFID systems, *PhD Thesis, University of Waterloo*, 2012.
11. J. Cho, Linear cryptanalysis of reduced-round PRESENT, *The Cryptographers' Track at the RSA Conference, CT-RSA'10*, LNCS 5985, pp. 302–317, 2010.
12. Q. Chai, X. Fan and G.Gong, An ultra-efficient key recovery attack on the lightweight stream cipher A2U2, *Cryptology ePrint Archive: Report 2011/247*, pp. 1–4, 2011.
13. B. Collard and F.X. Standaert, A statistical saturation attack against the block cipher PRESENT, *The Cryptographers' Track at the RSA Conference, CT-RSA'09*, LNCS 5473, pp. 195–210, 2009.
14. C. De Canniere, O. Dunkelman and M. Knežević, KATAN and KTANTAN – a family of small and efficient hardware-oriented block ciphers, *Cryptographic Hardware and Embedded Systems, CHES'09*, LNCS 5747, pp. 272–288, 2009.
15. A. Dvoretzky and P. Erdos, Some problems on random walk in space, *In Proceedings of Second Berkeley Symposium on Mathematical Statistics and Probability*, vol. 353, pp. 353–367, 1951.
16. D. Engels, X. Fan, G. Gong, H. Hu and E. Smith, Hummingbird: ultra-lightweight cryptography for resource-constrained devices, *Financial Cryptography and Data Security, FC'10*, pp. 3–18, 2010.
17. D. Engels, M.J.O. Saarinen and E. Smith, The Hummingbird-2 lightweight authenticated encryption algorithm, to appear in Proceedings of *Workshop on RFID Security, RFIDSec'11*, pp. 1-14, 2011.
18. X. Fan and G. Gong, On the security of Hummingbird-2 against side-channel Cube attacks, *Western European Workshop on Research in Cryptology, WEWoRC'11*, pp. 100–104, 2011.
19. M. Feldhofer, J. Wolkerstorfer and V. Rijmen, AES implementation on a grain of sand, *Information Security, IEE Proceedings*, vol. 152, no. 1, pp. 13-20, 2005.
20. Z. Gong, S. Nikova and Y.W. Law, Klein: a new family of lightweight block ciphers, to appear in Proceedings of *Workshop on RFID Security, RFIDSec'11*, pp.1–18, 2011.
21. J. Guo, T. Peyrin, A. Poschmann and M.Robshaw, The LED block cipher, *Cryptographic Hardware and Embedded Systems, CHES'11*, LNCS 6917, pp. 326–341, 2011.
22. F.D. Garcia, P. van Rossum, R. Verdult and R.W. Schreur, Dismantling SecureMemory, CryptoMemory and CryptoRF. *In Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS'10*, pp. 250-259, 2010.
23. K. Hwang, W. Lee, S. Lee, S. Lee and J. Lim, Saturation attacks on reduced round Skipjack, *Fast Software Encryption, FSE'02*, pp. 15–23, 2002.

24. A. Kiayias and M. Yung, cryptographic hardness based on the decoding of Reed-Solomon codes, *Automata, Languages and Programming*, pp. 783–783, 2002.
25. L. Knudsen, Truncated and higher order differentials, *Fast Software Encryption. FSE'95*, LNCS 1008, pp. 196–211, 1995.
26. L. Knudsen, G. Leander, A. Poschmann and M. Robshaw, PRINTcipher: a block cipher for IC-printing, *Cryptographic Hardware and Embedded Systems, CHES'10*, LNCS 6225, pp.16–32, 2011.
27. X. Lai, Higher order derivatives and differential cryptanalysis, *Kluwer International Series in Engineering and Computer Science*, pp. 227–227, 1994.
28. S. Lucks, The saturation attack: a bait for Twofish, *Fast Software Encryption, FSE'02*, pp. 187–205, 2002.
29. G. Leander, M.A. Abdelraheem, H. AlKhzaimi and E. Zenner, A cryptanalysis of PRINTcipher: the invariant subspace attack. *Advances in Cryptology, CRYPTO'11*, LNCS 6841, pp. 206-221, 2011.
30. J. Lu, J. Kim, N. Keller and O. Dunkelman, Improving the efficiency of impossible differential cryptanalysis of reduced Camellia and MISTY1, *The Cryptographers' Track at the RSA Conference, CT-RSA'08*, LNCS 4964, pp. 370–386, 2008.
31. G. Leander and A. Poschmann On the classification of 4 bit s-boxes, *Arithmetic of Finite Fields*, LNCS 4547, pp. 159-176, 2007.
32. K. Nohl, D. Evans, S. Starbug and H. Plötz, Reverse-engineering a cryptographic RFID tag, *In Proceedings of the 17th conference on Security symposium, USENIX'08*, pp. 185–193, 2008.
33. N. Courtois, G. Bard and D. Wagner, Algebraic and slide attacks on KeeLoq, *Fast Software Encryption, FSE'08*, LNCS 5086, pp. 97-115, 2008.
34. A. Poschmann, S. Ling and H. Wang, 256 bit standardized crypto for 650 GE–GOST revisited, *Cryptographic Hardware and Embedded Systems, CHES'10*, LNCS 6225, pp. 219–233, 2011.
35. C. Rolfes, A. Poschmann, G. Leander and C. Paar, Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents. *In Proceedings of the 8th IFIP WG 8.8/11.2 International Conference on Smart Card Research and Advanced Applications, CARDIS'08*, LNCS 5189, pp. 89–103, 2008.
36. M.J.O. Saarinen, Cryptanalysis of Hummingbird-1, *Fast Software Encryption, FSE'11*, LNCS 6733, pp. 328–341, 2011.
37. A. Shamir and E. Biham, Differential cryptanalysis of DES-like cryptosystems, *Advances in Cryptology, CRYPTO'90*, LNCS 537, pp. 2-21, 1990.
38. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita and T. Shirai, Piccolo: an ultra-lightweight blockcipher, *Cryptographic Hardware and Embedded Systems, CHES'11*, LNCS 6917, pp. 342–357, 2011.

A Side-channel Injection Attack to Realize ΔR

As mentioned in the Proof of Theorem 1, condition (A) gives us the same differential sequence as the following condition does,

$$\begin{aligned}
&\Delta K = (K_1, ..., K_8) + (K'_1, ..., K'_8) = (0,0,0,0,0,0,0,0)\\
&\Delta P = P_1 + P'_{i'} = 0\\
&\Delta R = (R_1, ..., R_8) + (R'_1, ..., R'_8) = (0,0,0,H,0,0,0,0),
\end{aligned}$$

Therefore, to create the difference between R_4 and R'_4 (or between $f^{-1}(R_2 \boxminus u_1)$ and $f^{-1}(R'_2 \boxminus u'_1)$), one obvious way is to start with two instances initialized with the same IVs and keys and then mount side-channel injection attack, where the attacker simply injects H to the victim register, e.g., R_4 or $f^{-1}(R_2 \boxminus u_1)$, of one instance any time before the execution of the last round of encryption/decryption. Note that the *preparation through injection* gives the attacker no time/memory penalty, i.e., the overall time/memory complexity of the attack is dominated by that of the *key recovery phase*.