Meet-in-the-Middle Attacks on Reduced-Round Hierocrypt-3

Ahmed Abdelkhalek, Riham AlTawy, Mohamed Tolba, and Amr M. Youssef

Concordia Institute for Information Systems Engineering
Concordia University, Montréal, Québec, Canada

Abstract. Hierocrypt-3 is an SPN-based block cipher designed by Toshiba Corporation. It operates on 128-bit state using either 128, 192 or 256-bit key. In this paper, we present two meet-in-the-middle attacks in the single-key setting on the 4-round reduced Hierocrypt-3 with 256-bit key. The first attack is based on the differential enumeration approach where we propose a truncated differential characteristic in the first 2.5 rounds and match a multiset of state differences at its output. The other attack is based on the original meet-in-the-middle attack strategy proposed by Demirci and Selçuk at FSE 2008 to attack reduced versions of both AES-192 and AES-256. For our attack based on the differential enumeration, the master key is recovered with data complexity of 2^{113} chosen plaintexts, time complexity of 2^{238} 4-round reduced Hierocrypt-3 encryptions and memory complexity of 2^{218} 128-bit blocks. The data, time and memory complexities of our second attack are 2^{32}, 2^{245} and 2^{239}, respectively. To the best of our knowledge, these are the first attacks on 4-round reduced Hierocrypt-3.

Keywords: Cryptanalysis, Hierocrypt-3, Meet-in-the-Middle attack, Differential Enumeration.

1 Introduction

Hierocrypt-3 (HC-3) [24,11,28], designed by Toshiba Corporation in 2000, is a 128-bit block cipher that was submitted to the New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project [23]. It is among the Japanese e-Government 2003 recommended ciphers list [10] and the 2013 candidate recommended ciphers list [9]. HC-3 employs a nested Substitution Permutation Network (SPN) structure as proposed in [24]. In nested SPN structure, each round has two substitution layers with distinct linear transformations and hence is equivalent to two rounds in normal SPN structure.

In the self-evaluation report done by Toshiba Corporation [28], HC-3 was concluded to be sufficiently secure against all well-known attacks at that time. Nevertheless, Barreto *et al.* presented an improved square attack against reduced round HC-3 [5]. They showed that HC-3 is vulnerable up to 3 rounds for 128-bit key, and up to 3.5 rounds for 192, 256-bit keys. These attacks are the best attacks on HC-3 so far. Then, Cheon *et al.* presented a 2-round impossible differential that can be used to attack up to 3 rounds of HC-3 [20]. Furuya and Rijmen analyzed the key scheduling of HC-3 and showed many linear relations between the round key bits [27]. Finally, after the introduction of the biclique attack in 2011 [6], Rechberger evaluated the security of HC-3, among other 128-bit block ciphers, against the biclique attacks [25].

In 1977, Diffie and Hellman were the first to propose the meet-in-the-middle (MitM) attack for the cryptanalysis of Data Encryption Standard (DES) [16]. Since then, the attack has evolved to analyze many block ciphers such as PRINCE and PRESENT [8], KTANTAN [7], LBlock [4], mCrypton [18], and Kuznyechik [3]. In addition, the MitM technique was used on hash functions to present preimage or second preimage attacks exemplified by the work done on HAS-160 [19], Whirlpool [26], Streebog [1], and Whirlwind [2]. Demirci and Selçuk were the first to apply the MitM approach on AES [13] triggering a new line of research, this attack is hereafter referred to as the *plain* MitM attack. They have shown that if the input of 4 AES rounds has just one active byte then the value of each byte of the output can be described as a parameterized function of that active byte. The number of parameters was deduced to be 25 8-bit parameters in [13] and then reduced to 24 8-bit parameters in [14]. The reduction in the number of parameters was possible by noticing that the 25th parameter is a key byte that is constant for all functions. Therefore, by considering the differences of the functions rather than the mere values, only 24 parameters can be used. The main disadvantage of the *plain* MitM attack, even with the reduced number of parameters, is the high memory requirement to store a precomputation table of all the sequences resulting from all the possible combinations of these parameters. As such, the *plain* MitM attack only works for AES-256 and then a time/memory tradeoff is used to extend the attack to AES-192.

At ASIACRYPT 2010, Dunkelman, Keller, and Shamir [17] proposed a couple of new ideas to address the high memory requirement of the *plain* MitM attack. First, they showed that the precomputation table does not need to have the whole sequence, just its associated multiset, i.e., the unordered sequence with multiplicity rather than the ordered one. The introduction of the multiset concept reduced the size of the precomputation table by a factor of 4. However, the more significant reduction in the size of the precomputation table was due to the second and main idea which they called the differential enumeration. Differential enumeration reduced the number of parameters that describe the sequence or rather the multiset from 24 bytes to 16 bytes. This is achievable by relying on a low probability truncated differential characteristic where the generated sequence or multiset at its output can only take a restricted number of values. Consequently, the memory requirement has been reduced from 2^{192} to 2^{128}. However, the use of this truncated differential characteristic increases the data complexity as now we have to search through a large amount of input data pairs to find one pair that conforms to the used truncated differential characteristic.

Later on, at Eurocrypt 2013, Derbez, Fouque and Jean showed that it is possible, by borrowing ideas from the rebound attack [22], to enumerate the whole set of sequences more efficiently [15]. In particular, they showed that using their technique, which they called efficient enumeration, the whole set can take only 2^{80} values instead of 2^{128}. This means that the number of parameters is further reduced to 10 parameters only. The consequences of using the efficient enumeration technique were numerous. Firstly, the attack became feasible on AES-128 and in fact, their attack is considered the most efficient attack on 7-round reduced AES-128. Secondly, the use of a 5-round truncated differential characteristic is feasible which mounts to attacking 9-round reduced AES-256. Thirdly, the memory complexity is no longer the bottleneck of the attack.

Finally, Li, Jia and Wang [21] employed a key-dependent sieve to further reduce the memory complexity of Derbez's attack. This technique helped them present an attack on 9-round reduced AES-192 using a 5-round truncated differential characteristic and an attack on 8-round reduced PRINCE.

In this work, we present two MitM attacks on HC-3; the first attack uses the idea of efficient differential enumeration while the second one utilizes the *plain* MitM attack in a data/memory tradeoff approach. Contrary to AES, HC-3 alternates between two distinct linear transformations; the first linear transformation, similar to the MixColumns transformation in AES, operates on 4 bytes while the other linear transformation acts on the whole 16-byte state. Nevertheless, we manage to construct a 2.5-round truncated differential characteristic that we use to mount an attack on 4-round reduced HC-3. In the second attack, we show that if the input has one active byte in a specific position then after 2 rounds of HC-3, certain bytes of the output can be described by a function of that active byte parameterized by 30 8-bit parameters. We use this 2-round distinguisher to attack 4-round reduced HC-3 as well with much less data complexity but higher memory and time complexities.

The rest of the paper is organized as follows. In Section 2, we describe the HC-3 block cipher and provide the notation used throughout the paper. Section 3 discusses the attack based on the efficient differential enumeration technique where we describe the chosen truncated differential characteristic and the adopted attack procedure. Afterwards in Section 4, we provide a brief description of our *plain* MitM attack on HC-3. Finally, we conclude the paper in Section 5.

2 Specification of Hierocrypt-3

HC-3 is an iterated nested SPN block cipher that operates on a 128-bit state with either 128, 192 or 256-bit key. In the nested SPN structure, the low level SPN structure is recursively used in the SPN of the higher level. That is, one large SBox is composed of two substitution layers with smaller SBoxes and one linear transformation in the middle of them. Therefore, one round in nested

SPN structure has two substitution layers and hence corresponds to two rounds in normal SPN structure. The number of rounds in HC-3 varies with the cipher key size. For 128-bit keys, HC-3 has 6 rounds, for 192-bit keys there are 7 rounds, and for 256-bit keys, 8 rounds. In all cases, the last round is slightly different than the other rounds.

At the higher SPN level, an HC-3 round, as shown in Figure 1, consists of three transformations:

– X: A subkey mixing layer consisting of xoring the 128-bit input with 16-byte subkey.
– XS: A layer of 4 parallel 32×32-bit keyed substitution boxes.
– H: A linear transformation consisting of xoring 8-bit subdata $x_{i(8)}(\in GF(2)^8; i = 0, 1, \cdots 15)$, which is represented by a matrix MDS_H. The matrix MDS_H and its inverse MDS_H^{-1} are given in Appendix **??**.

At the lower SPN level, each 32×32-bit XS-box consists of:

– S: A nonlinear layer composed of simultaneous application of 4 8×8-bit SBoxes.
– L: A bytewise linear transformation defined by a 4×4 matrix called mds_l. L has a branch number of 5, i.e., when the input (resp. output) has 1 active byte then the output (resp. input) must have 4 active bytes.
– X: A subkey mixing layer consisting of xoring the 32-bit input with 4-byte subkey.
– S: Another nonlinear layer composed of simultaneous application of 4 8×8-bit SBoxes. Hence, each round consists of two SBox layers and in our attacks below, the counting is done per SBox layer.

In the last round, the H transformation is replaced by a post-whitening key addition. Hence, the full encryption function of the 256-bit key version of HC-3 where the ciphertext C is evaluated from the plaintext P can be described as:

$$C = X[K_1^{(9)}] \circ ((S \circ X[K_2^{(8)}] \circ L \circ S) \circ X[K_1^{(8)}]) \circ \cdots$$
$$\circ (H \circ (S \circ X[K_2^{(1)}] \circ L \circ S) \circ X[K_1^{(1)}])(P)$$

For the convenience of describing our attacks, we use a different representation of HC-3, similar to the alternative expression used in [20]. As illustrated in Figure 2, the state is represented as a 4×4 matrix where each element in the matrix represents a state byte. In this representation, mds_l operates column-wise, similar to the MixColumns operation in AES, and MDS_H acts on the whole matrix. In some cases, we are also interested in swapping the order of the linear transformations, either H or L, and the xor with the key X. These operations are linear and hence they can be interchanged, by first xoring the input with an equivalent key and then applying the linear transformation. The equivalent subkey at SBox layer i is denoted by U_i where $U_i = L^{-1}(K_i)$ when i is odd or $U_i = H^{-1}(K_i)$ when i is even. Additionally, we rely on the property of the SBox stated in proposition 1 below :

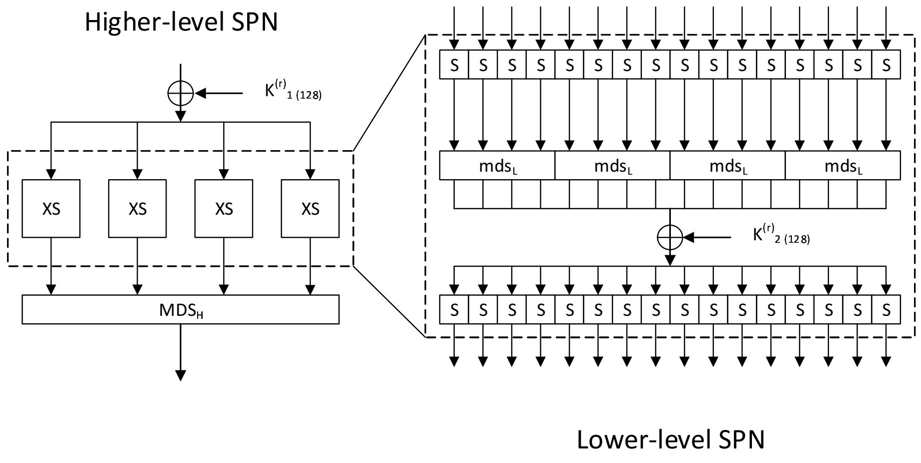


Fig. 1. One round of Hierocrypt-3.

Proposition 1 (Differential Property of S) *Given Δ_i and Δ_o two non-zero differences in $\mathbb{F}_{256}$, the equation: $S(x) + S(x + \Delta_i) = \Delta_o$ has one solution on average. This property also applies to S^{-1}.*

Key schedule. We now give a brief description of the 256-bit key schedule in HC-3.

The key state $Z_1 \| Z_2 \| Z_3 \| Z_4$ is 256-bit and undergoes 10 rounds to generate the 17 keys used in 8 rounds (2/1 keys per round/SBox layer) in addition to the final key. The first key state is denoted by $Z_1^{(-1)} \| Z_2^{(-1)} \| Z_3^{(-1)} \| Z_4^{(-1)}$ and instantiated with the master key. The first round is special as it omits a linear function and does not generate any round key. The other rounds can be split into 2 groups which we denote by $G1$ and $G2$. We focus our description on $G1$ as it is used to generate the round keys from round 1 up to round 5 which covers our attacked rounds. The key state words Z_3 and Z_4 are updated linearly every round, while Z_1 and Z_2 follow a Feistel structure with additional input from Z_3 and Z_4. Particularly, and as illustrated in Figure 3, the key state words are updated as follows:

$$\begin{aligned}(Z_3^{(r)}, Z_4^{(r)}) &= L_1(Z_3^{(r-1)}, Z_4^{(r-1)})\\ Z_1^{(r)} &= Z_2^{(r-1)}\\ Z_2^{(r)} &= Z_1^{(r-1)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)})\end{aligned}$$

where L_1 is a specific linear transformation and the function F_σ consists of a level of SBoxes followed by another, different than L_1, linear transformation.

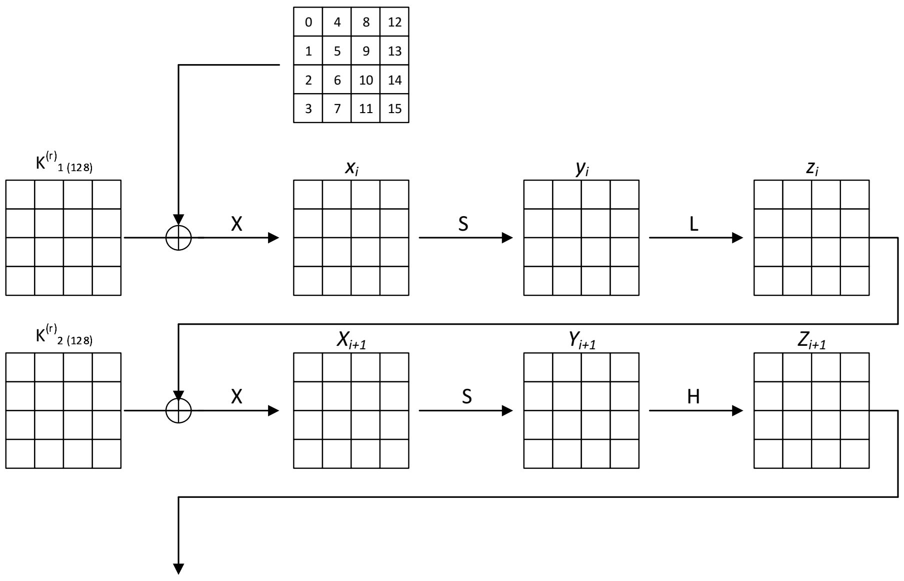


Fig. 2. Alternative representation of one round of Hierocrypt-3.

Then 64-bit keys $k_1^{(r)}, k_2^{(r)}, k_3^{(r)}$ and $k_4^{(r)}$ are generated as follows:

$$\begin{aligned}
k_1^{(r)} &= Z_1^{(r-1)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_2^{(r)} &= Z_3^{(r)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_3^{(r)} &= Z_4^{(r)} \oplus F_\sigma(Z_2^{(r-1)} \oplus Z_3^{(r)}) \\
k_4^{(r)} &= Z_4^{(r)} \oplus Z_2^{(r-1)}
\end{aligned}$$

where $r = 1, 2, \cdots, 5$ then the 128-bit $K_{1(128)}^{(r)}$ is set to $k_1^{(r)} \| k_2^{(r)}$ and $K_{2(128)}^{(r)}$ is set to $k_3^{(r)} \| k_4^{(r)}$. It is to be noted that in our attacks, we number the keys sequentially from K_1 up to K_{17} (for the full cipher) where $K_i = K_{1(128)}^{\lceil i/2 \rceil}$ when i is odd and $K_i = K_{2(128)}^{\lceil i/2 \rceil}$ when i is even.

For further details regarding the SBox, the linear transformations or the key schedule, the reader is referred to [28].

2.1 Notations

The following notations are used throughout the paper:

– x_i, y_i: The 16-byte state after the X, S transformations at layer i, respectively.

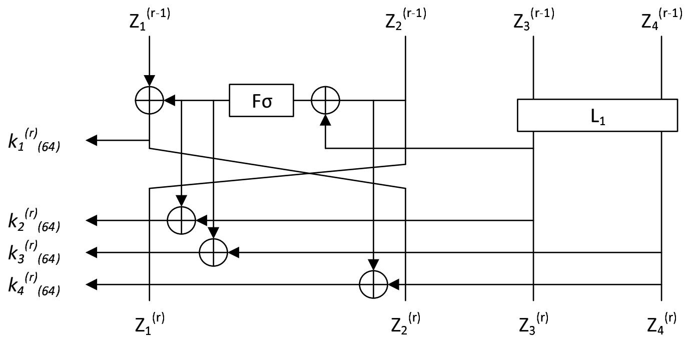


Fig. 3. 1 Round of Hierocrypt-3 key schedule.

- z_i: The 16-byte state after the linear transformation at layer i where the linear transformation is L (resp. H) when i is odd (resp. even).
- $x_i[j]$: The j^{th} byte of the state x_i, where $j = 0, 1, \cdots, 15$, and the bytes are indexed as shown in Figure 2.
- $x_i[j \cdots k]$: The bytes between the j^{th} position and k^{th} position of the state x_i.
- $\Delta x_i, \Delta x_i[j]$: The difference at state x_i and byte $x_i[j]$, respectively.

We measure memory complexity of our attacks in number of 128-bit HC-3 blocks and time complexity in reduced-round HC-3 encryptions.

3 A Differential Enumeration MitM Attack on HC-3

In a MitM attack, an r-round reduced block cipher is split into 3 consecutive parts of r_1, r_2 and r_3 rounds, $r = r_1 + r_2 + r_3$, such that a particular set of messages may verify a certain property in the middle r_2 rounds. In an offline phase, that particular property is evaluated independently of the keys used in the middle rounds. Then in an online phase, correct key candidates for the r_1 and r_3 rounds are checked whether they verify this distinguishing property or not. In our attacks, the chosen property is a truncated differential characteristic, such that when its input is a δ-set [12] captured by Definition 1, the set of each byte of the output states forms an ordered sequence or rather a multiset as in Dunkleman's attack and captured in Definition 2.

Definition 1 (δ-set of HC-3) *Let a δ-set be a set of* 256 *HC-3 states that are all different in one state byte (the active byte) and all equal in the other state bytes (the inactive bytes).*

Definition 2 (Multisets of bytes) *A multiset generalizes the set concept by allowing elements to appear more than once. In our case, a multiset of 256 bytes can take as many as* $\binom{2^8+2^8-1}{2^8} \approx 2^{506.17}$ *different values.*

Our first proposed 8-layer (4-round) MitM attack relies on a 6-layer distinguisher. The distinguisher, as illustrated in Figure 4, starts at x_1 and ends at the SBox transformation in layer 6, i.e., y_6. Proposition 2, below, is the core of our attack.

Proposition 2 *If a message m belongs to a pair of states conforming to the truncated differential characteristic of Figure 4, then the multiset of differences $\Delta y_6[3]$ obtained from the δ-set constructed from m in $x_1[0]$ is fully determined by the following 27 bytes: $\Delta y_1[0]$, $x_2[0\cdots 3]$, $x_3[0\cdots 15]$, $\Delta y_6[3]$, $y_6[3]$ and $y_5[0\cdots 3]$.*

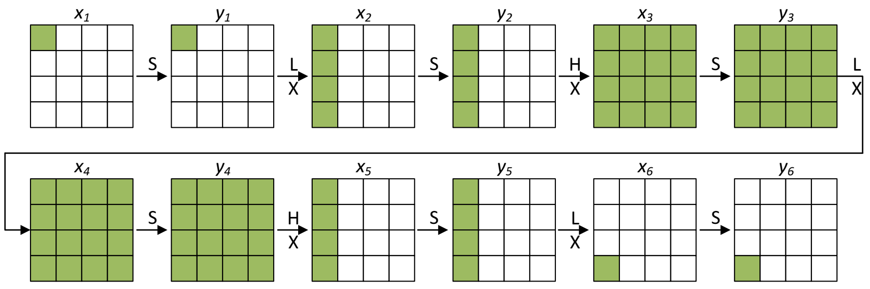


Fig. 4. The distinguisher used in the MitM attack on HC-3 using differential enumeration.

Proof. The proof uses rebound-like arguments borrowed from the hash function cryptanalysis [22] and used in [15]. Let (m, m') be a right pair that conforms to the truncated differential characteristic in Figure 4. We show in the following how the knowledge of these 27 particular bytes is enough to compute the mulisets. The 27 bytes $\Delta y_1[0]$, $x_2[0\cdots 3]$, $x_3[0\cdots 15]$, $\Delta y_6[3]$, $y_6[3]$ and $y_5[0\cdots 3]$ can take as many as $2^{8\times 27} = 2^{216}$ possible values, and for each of them, we can determine the values of all the differences shown in Figure 4. $\Delta y_1[0]$ can be propagated linearly through L to compute $\Delta x_2[0\cdots 3]$. With the knowledge of $x_2[0\cdots 3]$, we can bypass the SBox of layer 2 to reach y_2, then linearly through H to compute $\Delta x_3[0\cdots 15]$. With the knowledge of $x_3[0\cdots 15]$, we can bypass the SBox of layer 3 to reach y_3 and then linearly through L to compute $\Delta x_4[0\cdots 15]$. Similarly in the other direction, $\Delta y_6[3]$ and $y_6[3]$ enable us to bypass the SBox of layer 6 and compute $\Delta z_5[3]$, then linearly through L^{-1} we compute $\Delta y_5[0\cdots 3]$. With the knowledge of $y_5[0\cdots 3]$, we bypass the SBox of layer 5 to compute $\Delta z_4[0\cdots 3]$ and then linearly through H^{-1} we can compute $\Delta y_4[0\cdots 15]$. By the differential property of the HC-3 SBox (Proposition 1), there is, on average, one solution for each of the 16 bytes of the state x_4.

To construct the multiset for each of the 2^{216} possible values of the 27 bytes from proposition 2, we consider all the 255 possible values for $\Delta y_1[0]$ and propagate them until y_6 with the help of the internal state solutions we have. This results in a multiset of 255 differences in $\Delta y_6[3]$. As the HC-3 SBox is a permutation over $\mathbb{F}_{256}$, the sequence in $\Delta y_1[0]$ allows to derive the sequence in $\Delta x_1[0]$.

Attack Procedure. Our attack recovers the 128-bit last round key K_9 and 13 bytes of $U_8 = L^{-1}(K_8)$ and is composed of a precomputation phase and an online phase.

Precomputation phase. In the precomputation phase, we iterate over the 2^{216} possible values for the 27 bytes of proposition 2 and for each of them, we deduce the possible values of the internal states as discussed in the above proof. These internal state values are then used to generate the multiset. We store all the multisets in a hash table.

Online Phase. The online phase is further divided into two steps; data collection and key recovery. First, we try to find pairs of messages that conform to the truncated differential characteristic in Figure 5 in which the previous 6-layer characteristic is placed at the top. Next, the found pairs are used to create a δ-set and test them against the stored hash table to identify the correct key.

Data collection. In this step, we query the encryption oracle with structures of chosen plaintexts to get enough pairs such that one conforms to the 8-layer truncated differential characteristic in Figure 5. Each structure is composed of 2^8 plaintexts, where byte 0 takes all the 2^8 possible values while each of the other 15 bytes take any, possibly distinct, fixed value. Thus, each structure is expected to generate $2^8 \times (2^8 - 1)/2 \approx 2^{15}$ pairs. The probability that the whole truncated differential characteristic is verified is $2^{-7\times 8 - 8\times 8} = 2^{-120}$ because of the $16 \to 9$ transition over L^{-1} $(x_8 \to z'_7)$ and the $9 \to 1$ transition over H^{-1} $(x_7 \to z'_6)$, see Figure 5. While the probability of the former transition over L^{-1} is trivial to deduce, the probability of the latter transition over H^{-1} was deduced by observing that if the input of H^{-1} consists of certain 9 active bytes having the same difference, then the output will have just one active byte in a specific position. This position can be either $3, 7, 11$ or 15 depending on the position of the 9 bytes. Hence, the probability of the $9 \to 1$ transition is equivalent to the probability of 9 active bytes having the same difference, i.e., $2^8/2^{8\times 9} = 2^{-8\times 8} = 2^{-64}$. This observation is also applicable to H, but the positions of the 9 active input bytes and the active output byte differs from the ones corresponding to H^{-1}. Since the probability of the whole truncated differential characteristic is 2^{-120}, then in order to find one pair that conforms to it, we need 2^{120} pairs which means 2^{105} structures of 2^8 messages. Therefore, we ask for the encryption of $2^{105+8} = 2^{113}$ messages to get the required 2^{120} pairs.

Key Recovery. For each of the 2^{120} pairs, we get $2^{8\times(1+9)} = 2^{80}$ suggestions for the 25 key bytes: $K_9[0..15]$ and $U_8[1, 3 \cdots 6, 8, 10, 13, 15]$. This is done as follows: we guess $\Delta y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ and propagate it linearly through L to compute Δx_8. From the other side, Δx_9 is in fact the difference in the ciphertext pair and is also equal to Δy_8. So now, we have Δx_8 and Δy_8, so we use the differential property of the SBox and get a solution for each byte of x_8 and y_8 which enables us to deduce a key candidate for the whole K_9 by xoring the ciphertext with y_8, which in turn, helps compute $z_7^{'}$. Then, we guess $\Delta y_6[3]$ and propagate it through H to get $\Delta x_7[1, 3 \cdots 6, 8, 10, 13, 15]$. Once again, we use the differential property of the SBox along with $\Delta y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ which we guessed above to deduce a solution for the 9 bytes $y_7[1, 3 \cdots 6, 8, 10, 13, 15]$ which with $z_7^{'}$ enables us to compute the corresponding 9 bytes in U_8. To summarize this part, we guess 10 bytes that help deduce 25 key bytes.

To compute the multiset at $\Delta y_6[3]$ which is equivalent to $\Delta z_6^{'}[3]$, we noticed that extra key bytes are required. For example, if byte 3 is active in state y_i then, through H, it has an impact on 9 bytes in state z_i, these 9 bytes are $[1, 3 \cdots 6, 8, 10, 13, 15]$ (cf. 4^{th} column in H). However, byte 3, through H^{-1}, is impacted by 11 bytes in state z_i, these 11 bytes are $[0, 2 \cdots 6, 8 \cdots 10, 12, 13]$ (cf. 4^{th} row in H^{-1}). Hence, to compute the multiset, we need to guess the key bytes $[0, 2, 9, 12]$ on top of the above deduced keys. These key bytes are the gray cells in Figure 5. This means that for each of the 2^{120} pairs, we have $2^{80+4\times 8} = 2^{112}$ key candidates. For each pair and for every key candidate, we build the plaintext δ-set, get the corresponding ciphertexts, compute the multiset and look for a match in the precomutation table. If a match is not found, we can discard that key candidate. The probability of a wrong key producing a valid multiset is given by $2^{120+112+216-467.6} = 2^{-19.6}$ which is negligible. Note that the probability of randomly having a match in the table is $2^{-467.6}$ (and not $2^{-506.7}$) because the number of ordered sequences associated to a multiset is not constant [15].

Now, our attack recovers the 16-byte K_9 and 13 bytes of U_8 which is the equivalent key of K_8 so, in order to recover the master key, we guess 3 bytes of U_8 and get 2^{24} candidates for K_8. This means that we have recovered the 64-bit keys $k_3^{(4)}, k_4^{(4)}, k_1^{(5)}$ and $k_2^{(5)}$. According to the key schedule, they are calculated as follows:

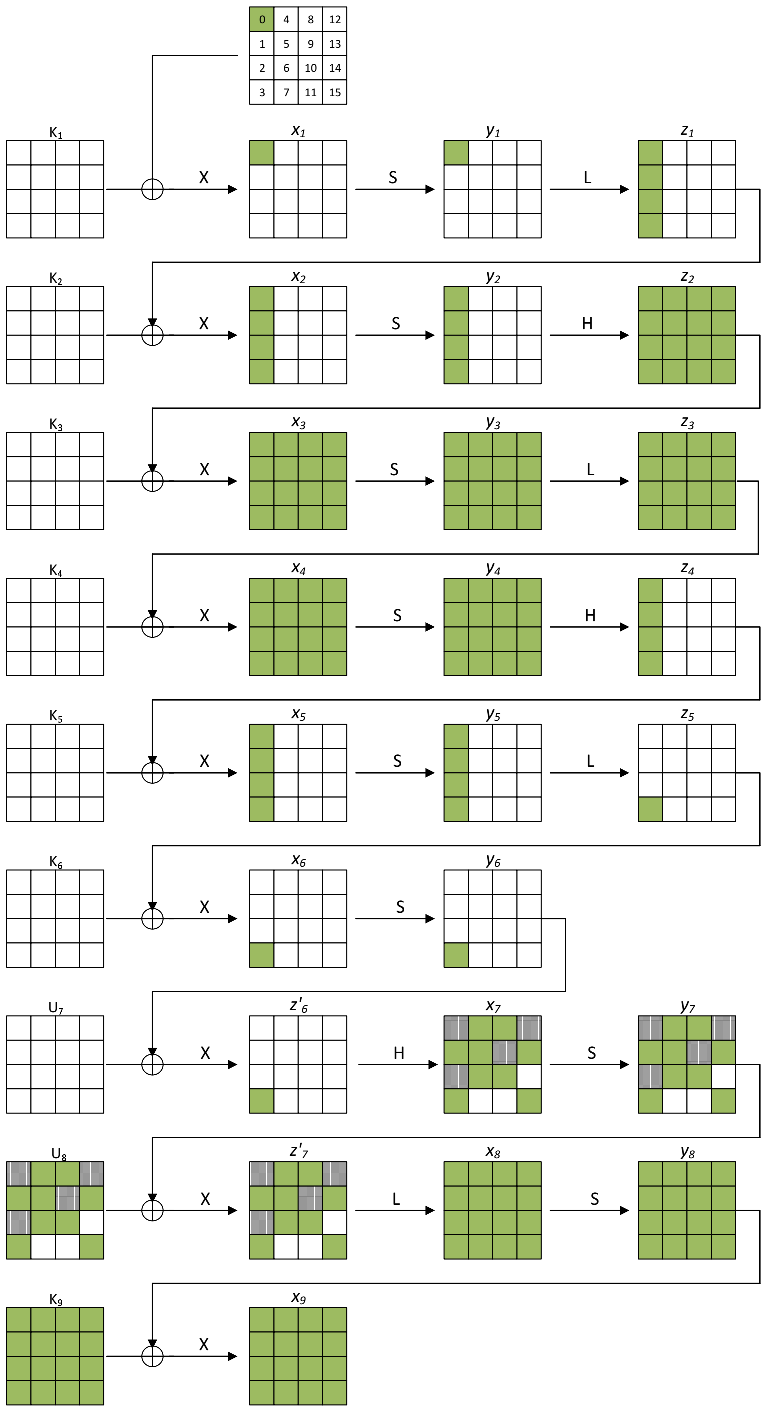
$$k_3^{(4)} = Z_4^{(4)} \oplus F_\sigma(Z_2^{(3)} \oplus Z_3^{(4)}) \tag{1}$$

$$k_4^{(4)} = Z_4^{(4)} \oplus Z_2^{(3)} \tag{2}$$

$$k_1^{(5)} = Z_1^{(4)} \oplus F_\sigma(Z_2^{(4)} \oplus Z_3^{(5)}) \tag{3}$$

$$k_2^{(5)} = Z_3^{(5)} \oplus F_\sigma(Z_2^{(4)} \oplus Z_3^{(5)}) \tag{4}$$

We start by guessing $Z_2^{(3)}$ which is of 64-bit length then from equation (2) we compute $Z_4^{(4)}$. From equation (1), we compute $Z_3^{(4)}$. From the key schedule, knowing $Z_3^{(4)}$ and $Z_4^{(4)}$ enables us to compute $Z_3^{(5)}$ and $Z_4^{(5)}$. Afterwards from equation (3) and considering that $Z_1^{(4)} = Z_2^{(3)}$, we compute $Z_2^{(4)}$. We use equation (4) as a 2^{-64} filter and we end up with one solution for $Z_2^{(3)}, Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}$, $Z_4^{(4)}, Z_3^{(5)}$ and $Z_4^{(5)}$. Since we have $Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}$, $Z_4^{(4)}$ we can recover the master



0 4 8 12
1 5 9 13
2 6 10 14
3 7 11 15
K_1 X x_1 S y_1 L z_1
K_2 X x_2 S y_2 H z_2
K_3 X x_3 S y_3 L z_3
K_4 X x_4 S y_4 H z_4
K_5 X x_5 S y_5 L z_5
K_6 X x_6 S y_6
U_7 X z'_6 H x_7 S y_7
U_8 X z'_7 L x_8 S y_8
K_9 X x_9

Fig. 5. Complete 8-layer truncated differential characteristic used in the attack using differential enumeration.

key and get 2^{24} candidates for the master key corresponding to the 2^{24} candidates for K_8. We exhaustively search through these master key candidates to find the correct master key with no significant impact on the attack time complexity.

Attack Complexity. The memory requirement of the attack is due to the precomputation table needed to store 2^{216} multisets, each of 512 bits. Hence, the memory complexity of the attack is $2^{216} \times 512/128 = 2^{218}$ 128-bit blocks. The data complexity of the attack is attributed to the data collection step of the online phase where we query the encryption oracle with 2^{113} chosen plaintexts to generate 2^{120} pairs. The time complexity of the offline phase to construct the table is due to performing 2^{216} partial encryptions on 256 messages, which is equivalent to $2^{216+8} \times 2^{-2} = 2^{222}$ encryptions. The time complexity of the online phase to recover 29 key bytes (16-byte K_9 and 13 bytes from U_8) is the time required to partially decrypt the 2^8 values in a δ-set with all the 2^{112} key candidates for all the 2^{120} generated pairs, which is equivalent to $2^{120+112+8} \times 2^{-2} = 2^{238}$. The time complexity of the exhaustive search among the 2^{24} master key candidates using 2 plaintext/ciphertext pairs is $2 \times 2^{24} = 2^{25}$. Therefore, the time complexity of the attack is dominated by the time complexity of the online phase and is equivalent to $2^{238} + 2^{222} + 2^{25} \approx 2^{238}$.

4 A *plain* MitM Attack on HC-3

Our second proposed MitM attack is also launched on 8-layer of the 256-bit key version of HC-3 in a data/memory tradeoff approach. Although its memory and time complexities, as will be shown, are higher than the previous attack by a factor of 2^{21} and 2^7, respectively, it reduces the data complexity dramatically by a factor of 2^{81}. It follows the strategy used by Demirci and Selçuk, which we named above as the *plain* MitM. As illustrated in Figure 6, the distinguisher in this attack covers 4 middle layers starting from x_2 to x_6. Proposition 3 is the base of our attack.

Proposition 3 *The ordered sequence of differences $\Delta x_6[2]$ obtained from the δ-set constructed by varying $x_2[3]$ is fully determined by the following 30 bytes: $x_2[3]$, $x_3[1, 3 \cdots 6, 8, 10, 13, 15]$, $x_4[0 \cdots 15]$ and $x_5[0 \cdots 3]$.*

Proof. The proof is similar to the proof of proposition 2 without using the rebound-like arguments. We show in the following how the knowledge of these 30 particular bytes is enough to compute the ordered sequence of differences at $x_6[2]$. $\Delta x_2[3]$ and $x_2[3]$ help us bypass the SBox of layer 2 to compute $\Delta y_2[3]$ which is then propagated linearly through H to compute $\Delta x_3[1, 3 \cdots 6, 8, 10, 13, 15]$. With the knowledge of $x_3[1, 3 \cdots 6, 8, 10, 13, 15]$, we can bypass the SBox of layer 3 and then linearly through L, we compute $\Delta x_4[0 \cdots 15]$. With the knowledge of $x_4[0 \cdots 15]$, we can bypass the SBox of layer 4 and once again linearly through H, we compute $\Delta x_5[0 \cdots 15]$. With the knowledge of $x_5[0 \cdots 3]$, we can bypass the SBox of layer 5 in these 4 bytes to reach $y_5[0 \cdots 3]$ and then linearly through

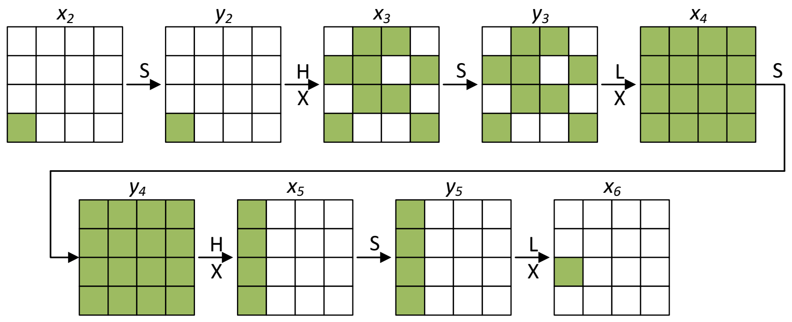


Fig. 6. The 4-layer distinguisher used in the MitM attack on HC-3 using Demirci and Selçuk approach.

L to compute $\Delta x_6[2]$. The byte where the distinguisher starts was chosen such that it minimizes the number of parameters. $x_2[7, 11, 15]$ are other possible positions that would result in the same number of parameters. The byte where the ordered sequence is computed was chosen to minimize the number of key guesses in the online phase. $x_6[6, 10, 14]$ are other positions that would require the same number of key bytes.

Attack Procedure. In this attack, we recover the 16-byte K_9, 9 bytes of $U_8 = L^{-1}(K_8)$, 1 byte of $U_7 = L^{-1}(K_7)$ and 4 bytes of K_1. Similar to the previous attack, this one is composed of a precomputation phase and an online phase. The precomputation phase in this attack is similar to the precomputation phase in the previous one, we build a hash table for the ordered sequence of $\Delta x_6[2]$ by iterating over all the possible values of the 30 parameters. The online phase is a bit different and thus explained in details below.

Online Phase. In the online phase, we choose an arbitrary plaintext as P^0, guess 4 key bytes $K_1[0 \cdots 3]$ and partially encrypt P^0 through the first layer to reach z_1. We create the δ-set at $z_1[3]$, which is equivalent to creating the δ-set at $x_2[3]$, and decrypt it backward to identify the plainttexts forming the δ-set containing P^0 and ask for their corresponding ciphertexts. From these ciphertexts and by guessing 26 key bytes, we compute the ordered sequence of $\Delta x_6[2]$ and look for a match in the stored hash table. The 26 key bytes to be guessed are $K_9[0 \cdots 15]$, $U_8[0, 1, 3, 6, 7, 10, 11, 13, 15]$ and $U_7[2]$ as depicted in Figure 7. The probability for a false positive, i.e. a wrong key guess resulting in a match, is given by $2^{8\times30-2040} = 2^{-1800}$ and as we try 2^{240} key candidates, we expect that only $2^{240-1800} = 2^{-1560}$ remain after this step. We notice that the probability of a wrong key producing a valid $2^8 - 1$ bytes ordered sequence is negligible and can be relaxed. Hence, we use the partial sequence matching idea proposed in [3]. Instead of matching $2^8 - 1$ bytes ordered sequence, we match b bytes such that $b < 2^8$ and the probability of error, chosen to be 2^{-32}, is still small enough

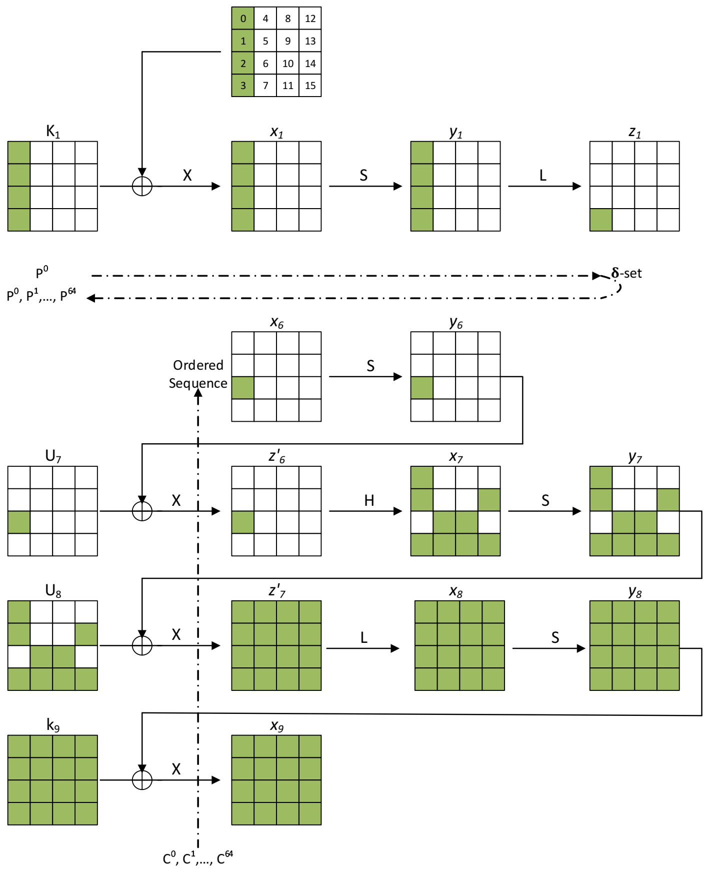


Fig. 7. The online phase of the 8-layer MitM attack on HC-3 using Demirci and Selçuk approach.

to be able to identity the right key. In that case, the number of required bytes b is calculated by $2^{-32} = 2^{8\times30+240-8b}$, which yields $b = 64$. This means that it is enough to match 64 bytes of the ordered sequence to identify the right key with a negligible error probability of 2^{-32}. It should be noted that in this attack, we opted for using the ordered sequence rather than the multiset because in the case of multiset, the probability of error is not small enough to identify the right key. Finally, as we recover 9 bytes from U_8, we have 2^{56} candidates for K_8 and following the same approach as in the previous attack, we get 2^{56} candidates for the master key which we exhaustively search to retrieve the correct master key with no major impact on the overall attack time complexity.

Attack Complexity. The memory requirement of the attack is due to the precomputation table needed to store the partial ordered sequence and is estimated to be $2^{240} \times 64/128 = 2^{239}$ 128-bit blocks. The data complexity of the attack is 2^{32} chosen plaintexts. The time complexity of the offline phase is equivalent to $2^{240} \times 65 \times 2^{-2} \approx 2^{244}$ encryptions. The time complexity of the online phase to recover 30 key bytes (The 16-byte K_9, 4 bytes from K_1, 9 bytes from U_8 and 1 byte from U_7) is equivalent to $2^{240} \times 65 \times 2^{-2} = 2^{244}$. Therefore, the total time complexity of the attack is $2^{244} + 2^{244} + 2^{57} \approx 2^{245}$.

5 Conclusion

In this work, we have presented two MitM attacks on the 256-bit key version of one of the Japanese e-Government candidate recommended block ciphers; Hierocrypt-3. The first attack employs the differential enumeration technique while the second one follows the strategy of Demirci and Selçuk. Both attacks are mounted against 8 SBox layers (4 rounds) and the complexities of the first attack are 2^{113} chosen plaintexts, 2^{238} 4-round reduced Hierocrypt-3 encryptions and 2^{218} 128-bit blocks. The other attack has much less data complexity of 2^{32} chosen plaintexts but higher time and memory complexities of 2^{245} encryptions and 2^{239} 128-bit block, respectively. We have noticed that the first attack based on the differential enumeration technique works in the chosen ciphertext context as well with exactly the same data, time and memory complexities. To the best of our knowledge, these are the best attacks on the 256-bit version of Hierocrypt-3 in the single-key setting.

References

1. AlTawy, R., and Youssef, A. Preimage Attacks on Reduced-Round Stribog. In *Progress in Cryptology AFRICACRYPT 2014*, D. Pointcheval and D. Vergnaud, Eds., vol. 8469 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 109–125.
2. AlTawy, R., and Youssef, A. M. Second Preimage Analysis of Whirlwind. In *Inscrypt* (2014), Lin, Dongdai, Yung, Moti, Zhou, Jianying (Eds.) *to appear*.
3. AlTawy, R., and Youssef, A. M. Meet in the Middle Attacks on Reduced Round Kuznyechik. Cryptology ePrint Archive, Report 2015/096, 2015. http://eprint.iacr.org/.
4. AlTawy, R., and Youssef, A. Differential Sieving for 2-step matching meet-in-the-middle attack with application to LBlock. In *Lightsec* (2014), Lecture Notes in Computer Sience, Springer.
5. Barreto, P., Rijmen, V., Nakahara, Jorge, J., Preneel, B., Vandewalle, J., and Kim, H. Improved Square Attacks against Reduced-Round Hierocrypt. In *Fast Software Encryption*, M. Matsui, Ed., vol. 2355 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 165–173.
6. Bogdanov, A., Khovratovich, D., and Rechberger, C. Biclique Cryptanalysis of the Full AES. In *Advances in Cryptology ASIACRYPT 2011*, D. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 344–371.

7. Bogdanov, A., and Rechberger, C. A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In *Selected Areas in Cryptography*, A. Biryukov, G. Gong, and D. Stinson, Eds., vol. 6544 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 229–240.
8. Canteaut, A., Naya-Plasencia, M., and Vayssire, B. Sieve-in-the-Middle: Improved MITM Attacks. In *Advances in Cryptology CRYPTO 2013*, R. Canetti and J. Garay, Eds., vol. 8042 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 222–240.
9. CRYPTEC. e-Government Candidate Recommended Ciphers List, 2013. http://www.cryptrec.go.jp/english/method.html.
10. CRYPTEC. e-Government Recommended Ciphers List, 2003. http://www.cryptrec.go.jp/english/images/cryptrec_01en.pdf.
11. CRYPTEC. Specification on a Block Cipher: Hierocrypt-3. http://www.cryptrec.go.jp/cryptrec_03_spec_cypherlist_files/PDF/08_02espec.pdf.
12. Daemen, J., Knudsen, L., and Rijmen, V. The block cipher SQUARE. In *Fast Software Encryption*, E. Biham, Ed., vol. 1267 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1997, pp. 149–165.
13. Demirci, H., and Seluk, A. A Meet-in-the-Middle Attack on 8-Round AES. In *Fast Software Encryption*, K. Nyberg, Ed., vol. 5086 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 116–126.
14. Demirci, H., Takn, h., oban, M., and Baysal, A. Improved Meet-in-the-Middle Attacks on AES. In *Progress in Cryptology - INDOCRYPT 2009*, B. Roy and N. Sendrier, Eds., vol. 5922 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 144–156.
15. Derbez, P., Fouque, P.-A., and Jean, J. Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting. In *Advances in Cryptology EUROCRYPT 2013*, T. Johansson and P. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 371–387.
16. Diffie, W., and Hellman, M. E. Special feature exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer 10*, 6 (1977), 74–84.
17. Dunkelman, O., Keller, N., and Shamir, A. Improved Single-Key Attacks on 8-Round AES-192 and AES-256. In *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed., vol. 6477 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 158–176.
18. Hao, Y., Bai, D., and Li, L. A Meet-in-the-Middle Attack on Round-Reduced mCrypton Using the Differential Enumeration Technique. In *Network and System Security*, M. Au, B. Carminati, and C.-C. Kuo, Eds., vol. 8792 of *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 166–183.
19. Hong, D., Koo, B., and Sasaki, Y. Improved Preimage Attack for 68-Step HAS-160. In *Information, Security and Cryptology ICISC 2009*, D. Lee and S. Hong, Eds., vol. 5984 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 332–348.
20. Jung Hee Cheon, MunJu Kim, and Kwangjo Kim. Impossible Differential Cryptanalysis of Hierocrypt-3 Reduced to 3 Rounds. NESSIE report, 2002.
21. Li, L., Jia, K., and Wang, X. Improved Meet-in-the-Middle Attacks on AES-192 and PRINCE. Cryptology ePrint Archive, Report 2013/573, 2013. http://eprint.iacr.org/.
22. Mendel, F., Rechberger, C., Schlffer, M., and Thomsen, S. The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grϕstl. In *Fast Software Encryption*, O. Dunkelman, Ed., vol. 5665 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 260–276.

23. New European Schemes for Signatures, Integrity, and Encryption. https://www.cosic.esat.kuleuven.be/nessie.
24. Ohkuma, K., Muratani, H., Sano, F., and Kawamura, S. The Block Cipher Hierocrypt. In *Selected Areas in Cryptography*, D. Stinson and S. Tavares, Eds., vol. 2012 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 72–88.
25. Rechberger, C. Security evaluation of 128-bit block ciphers AES, CIPHERUNICORN-A, and Hierocrypt-3 against biclique attacks. *CRYPTREC* (2012).
26. Sasaki, Y., Wang, L., Wu, S., and Wu, W. Investigating Fundamental Security Requirements on Whirlpool: Improved Preimage and Collision Attacks. In *Advances in Cryptology ASIACRYPT 2012*, X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 562–579.
27. S.Furuya and V.Rijmen. Observations on Hierocrypt-3/L1 key-scheduling algorithms. 2nd NESSIE Workshop, 2001.
28. Toshiba Corporation. Block Cipher Family Hierocrypt . http://www.toshiba.co.jp/rdc/security/hierocrypt/index.htm.