IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E104-A NO. 9
SEPTEMBER 2021

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

IEICE TRANS. FUNDAMENTALS, VOL.E104–A, NO.9 SEPTEMBER 2021

PAPER

Update on Analysis of Lesamnta-LW and New PRF Mode LRF*

Shoichi HIROSE[†a)], **Yu SASAKI**[††b)], ***and*** **Hirotaka YOSHIDA**[†††c)], ***Members***

SUMMARY We revisit the design of Lesamnta-LW, which is one of the three lightweight hash functions specified in ISO/IEC 29192-5:2016. Firstly, we present some updates on the bounds of the number of active S-boxes for the underlying 64-round block cipher. While the designers showed that the Viterbi algorithm ensured 24 active S-boxes after 24 rounds, our tool based on Mixed Integer Linear Programming (MILP) in the framework of Mouha et al. ensures the same number of active S-boxes only after 18 rounds. The tool completely evaluates the tight bound of the number of active S-boxes, and it shows that the bound is 103 for full (64) rounds. We also analyze security of the Shuffle operation in the round function and resistance against linear cryptanalysis. Secondly, we present a new mode for a pseudorandom function (PRF) based on Lesamnta-LW. It is twice as efficient as the previous PRF modes based on Lesamnta-LW. We prove its security both in the standard model and the ideal cipher model.

key words: *Lesamnta-LW, differential cryptanalysis, MILP, PRF, modes, standard model, ideal cipher model*

1. Introduction

The design of cryptographic schemes has been studied for a long time. In early days, a dedicated primitive was designed for each functionality such as encryption, authentication and hashing. In contrast, the recent approach is to design a single primitive and provide multiple functionalities by modes of operations. Permutation-based crypto, typically using SHA-3 [24], is an example that provides various functionalities from a single primitive. This modular approach is particularly useful for lightweight cryptography that needs to provide multiple functionalities with limited resource. Such a feature is actually taken into account by the ongoing lightweight cryptography standardization process by NIST, which considers algorithms implementing both of the authenticated encryption (AE) functionality and the hashing functionality.

Manuscript received September 15, 2020.
Manuscript revised January 31, 2021.
Manuscript publicized March 16, 2021.

[†]The author is with Faculty of Engineering, University of Fukui, Fukui-shi, 910-8507 Japan.
[††]The author is with NTT Secure Platform Laboratories, Musashino-shi, 180-8585 Japan.
[†††]The author is with National Institute of Advanced Industrial Science and Technology, Tokyo, 135-0064 Japan.

*An earlier version of this paper appeared in the proceedings of the ACNS 2020 conference [14]. The current article newly added results on linear cryptanalysis, full security proof, and an example code of MILP.

a) E-mail: hrs_shch@u-fukui.ac.jp
b) E-mail: yu.sasaki.sk@hco.ntt.co.jp
c) E-mail: hirotaka.yoshida@aist.go.jp

DOI: 10.1587/transfun.2020EAP1109

Lesamnta-LW is a lightweight hash function designed by Hirose et al. [8], [9], which is a successor of the hash function Lesamnta; one of the first-round candidates in the NIST SHA-3 competition [11]. Some symmetric property of Lesamnta was pointed out by a third-party [4] that was caused by a small amount of constants, and the designers protected it by replacing the round constants right after the detection [12]. This experience seems to offer a certain level of reliability for Lesamnta-LW. Indeed, in 2016, Lesamnta-LW was internationally standardized by the ISO/IEC JTC 1 SC 27 technical committee. Lesamnta-LW is the only lightweight hash function optimized for software implementations specified in ISO/IEC 29192-5:2016 [17].

Lesamnta-LW is a Merkle-Damgård hash function with its compression function a dedicated block cipher, which we call Lesamnta-LW-BC. It is a 64-round block cipher and its block size and key size are 256 bits and 128 bits, respectively. The use of the bigger block size than the key size can be found only in a limited number of designs, e.g. Rijndael [5] and SHACAL-2 [7].

Lesamnta-LW-BC consists of two parts: the mixing function and the key schedule function. Its one-round is depicted in Fig. 1. Lesamnta-LW-BC adopts the 4-branch type-1 generalized Feistel network (GFN). Each branch of the mixing function consists of 8 bytes. The updating function G operates on two columns of 4 bytes. The 4-byte round-key is added to the left column and then each column is updated by the function Q, which applies the AES [23] 1-byte S-box (SubBytes, SB) followed by the AES column-wise linear operation (MixColumns, MC). Finally, 8 bytes are permuted by the shuffle operation where the byte positions $(0, 1, 2, 3, 4, 5, 6, 7)$ move to $(4, 5, 2, 3, 0, 1, 6, 7)$. Each branch of the key schedule function also consists of 4 bytes. The updating function adds the round-constant and applies the function Q. A more formal description of Lesamnta-LW-BC is given in Appendix A.

The designers developed keyed modes of a pseudorandom function (PRF) based on Lesamnta-LW, which is interesting in terms of the integration of the hash function and the keyed function based on the single primitive Lesamnta-LW-BC. For short messages, a key-prefix (KP) mode [8], [9] gains significant advantage over the standard method HMAC-SHA-256. This KP mode has recently standardized as Tsudik's keymode specified in ISO/IEC 29192-6:2019 [18]. A similar mode using the Merkle-Damgård-Permutation (MDP) was also proposed [1].

There are few pieces of work on evaluation of

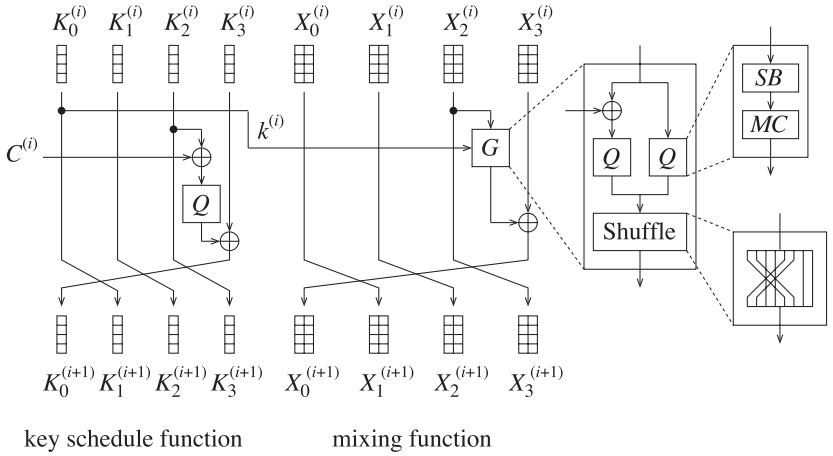

Fig. 1 One round of Lesamnta-LW-BC.

Lesamnta-LW. Except for the design extension by the designers, as far as we know, there is no follow-up security analysis even for Lesamnta-LW-BC. No follow-up work exists for the PRF modes, either. Lesamnta-LW is an ISO standard and it needs further analyses. For security evaluation of Lesamnta-LW, Lesamnta-LW-BC is an important target because its ideal behaivor is the core of the security proof of Lesamnta-LW.

1.1 Our Contributions

We revisit the security analysis and the designs of Lesamnta-LW and its PRF mode. The paper contains the following contributions.

First we improve the security evaluation of Lesamnta-LW-BC against differential cryptanalysis. The designers used the Viterbi algorithm and evaluated the number of active S-boxes [8]. By their evaluation, it was shown to be lower bounded by 24 for 24 rounds. In this work, we evaluate it with MILP in the framework of Mouha et al. [22] and obtain the following results.

- 24 active S-boxes are ensured only by 18 rounds. It implies that the number of total rounds may be reduced to 48 rounds ($= 64 \times 18/24$) without reducing the security level originally expected by the designers.
- Considering that the block size of Lesamnta-LW-BC is 256 bits, we derive the bounds for more rounds and show that 30 rounds are sufficient to ensure 43 active S-box with maximum characteristic probability of $2^{-6\times 43} = 2^{-258}$.
- After two weeks, we found that the minimum number of active S-boxes for the full (64) rounds is 103. With this result, the problem of evaluating the security of Lesamnta-LW-BC against differential cryptanalysis was closed.

We also provide the analysis of the Shuffle operation, where the designers borrowed it from the MUGI stream cipher [27] based on the fact that MUGI has been specified in ISO/IEC 18033-4:2005 [16] (thus reliable), while no security analysis dedicated to the structure of Lesamnta-LW is given. Note that security analysis of existing design components is important especially for standardized designs, and there are several previous researches in this line e.g. against SHA-1 [26] and SIMON [19]. It is possible to imagine that the designers adopted a two-byte-wise permutation to optimize implementations by 16-bit micro-controllers. However, we may have better security by replacing the Shuffle with a byte-wise permutation. We show that the original Shuffle is one of the best even including byte-wise permutations with respect to the number of active S-boxes as well as micro-controller implementations.

In addition, we investigate the security of Lesamnta-LW-BC against linear cryptanalysis for the first time. Note that Lesamnta-LW is a keyless hash function, and thus the impact of linear cryptanalysis is unclear. However, Lesamnta-LW-based PRF modes are keyed modes, thus linear cryptanalysis may apply. Our analysis shows that the number of rounds that can be attacked by linear cryptanalysis with less than 2^{128} data is at most 20 rounds. This implies that the number of rounds to resist linear cryptanalysis (20 rounds) is a bit more than the case with differential cryptanalysis (18 rounds), but still lower than designer's original expectation.

Second, we propose a new mode for PRF which achieves double throughput and reduces the key size by half compared with the previous ones [1], [9]. We call it LRF. It processes 256 bits of message per block cipher call, while the previous modes process 128 bits of message. Its key size is 128 bits. Previous PRF modes and LRF are depicted in Fig. 2 and Fig. 3, respectively. In Fig. 2, the first mode truncates the output and the second mode adopts the MDP mode that applies a light public permutation to a part of the plaintext input before the last block cipher call. LRF in Fig. 3 adopts MDP to the key input. Here, the public permutation is XOR with some predefined constant which is the best possible to keep the scheme light. We prove the security of LRF both in the standard and the ideal cipher models. From the

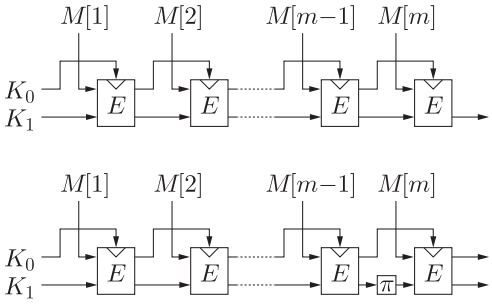

Fig. 2 Previous PRF modes of Lesamnta-LW-BC. π is a permutation.

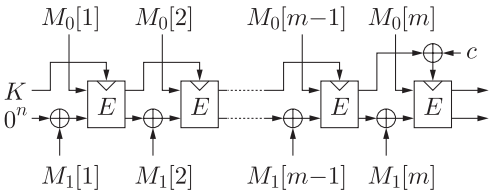

Fig. 3 LRF: New PRF mode of Lesamnta-LW-BC. $c \in \{0,1\}^n \setminus \{0^n\}$ is a constant. Compared to the previous modes, two 128-bit message blocks are absorbed per block cipher call.

result in the ideal cipher model, it is secure up to the birthday bound of the underlying block cipher, which is 128 bits for Lesamnta-LW-BC. We should mention that, due to the application of MDP to the key input, LRF requires Lesamnta-LW-BC to be secure against related-key attacks. However, the related key attacks are able to only exploit the XOR relation for the predefined constant.

If a nonce is prepended to the input message, LRF is a variant of the leakage-resilient re-keying MAC in [25]. Thus, it is expected to be secure against side channel attacks if protected by the leveled implementation [6], [25]. LRF is a non-trivial and improved variant since it accepts variable-length inputs, while the re-keying MAC only accepts fixed-length inputs. In addition, it uses a block cipher with its block size larger than its key size, while the re-keying MAC only considers a block cipher with its block size equal to its key size.

Lastly we provide several discussions to have better understanding.

- If we apply the byte-wise truncated differential search in the related-key setting against Lesamnta-LW-BC, the number of active S-boxes can be zero for any number of rounds. However such an efficient trail cannot be satisfied by taking into account the bit-level difference propagation.
- LRF is inspired by the previous schemes achieving higher throughput such as the boosting Merkle-Damgård MAC [28] and the full-state keyed sponge [3]. We considered some other candidates similar to LRF. However, their security turned out to be quite different: They allow distinguishing attacks with about 2^{64} complexity.

1.2 Paper Outline

In Sect. 2, we explain the MILP-based differential trail search. In Sect. 3, we show the security analysis of Lesamnta-LW-BC. In Sect. 4, we present the new PRF mode and prove its security. In Sect. 5, we give several useful discussion and conclude this paper.

2. Truncated Differential Search with MILP

Mouha et al. [22] showed that the problem of finding truncated differential trails with minimum number of active S-boxes can be converted into a minimization problem in the framework of MILP. Problems for MILP are defined by three factors; objective function, constraints of variables represented by linear inequalities, and variables with their value ranges. In the differential trail search, those three factors are intuitively explained as follows.

Variables. A binary variable is assigned to each byte in order to represent whether the byte has non-zero difference (active) or zero-difference (inactive). Let $x_i \in \{0,1\}$ be a binary variable for the i-th byte of the state. Then we define

$$\begin{cases} x_i = 1 & \text{if the } i\text{-th byte is active,} \\ x_i = 0 & \text{if the } i\text{-th byte is inactive.} \end{cases}$$

Objective Function. The goal is to find a pattern of x_i for all i such that the total number of active S-boxes is minimized. Given the above definition of x_i, the objective function is defined as

$$\text{minimize} \sum_i x_i.$$

Constraint Linear Inequalities. Active and inactive byte positions must be consistent with differential propagation patterns specified by cipher's algorithm. Those valid patterns must be described by linear inequalities, and cryptographer's main task is to find such conversions between cipher's operation and linear inequalities. Exact forms of linear inequalities largely depend on cipher's operation and we explain it by using a toy example below.

After a problem for MILP is defined, it can be given to any MILP solver, e.g. Gurobi Optimizer [15], and the solver returns the optimal solution if exists.

2.1 Example: MILP Model for Toy Cipher

We explain details of the model making of the framework of Mouha et al. [22] by using a 2-round toy cipher shown in Fig. 4, in which the state consists of 4 bytes and the round function consists of the key addition with k_0 and k_1, SubBytes and MixColumns.

The internal state consists of 12 bytes for 2 rounds. The

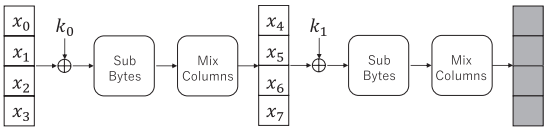

Fig. 4 Toy example.

last 4 bytes (in grey) do not affect to the number of active S-boxes, thus can be ignored simply. 8 binary variables $x_0, x_1, \ldots, x_7$ represent whether each byte is active or not. The objective function is to minimize the number of active S-boxes, which is defined as

$$\text{minimize} \quad x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7.$$

The remaining is to specify the valid differential propagation with linear inequalities. The key addition and SubBytes do not affect the active status of bytes thus those operations are simply ignored. The MixColumns operation has the property that the sum of the input and output active bytes is 5 or more, otherwise 0. Mouha et al. [22] showed that this property can be modeled by using an additional binary variable and 9 linear inequalities, where a dummy variable represents whether the column is active or not. Let $d_0 \in \{0, 1\}$ be a dummy variable for the first round. Then, linear inequalities for the first round are as follows.

$$x_0 + x_1 + \cdots + x_7 - 5d_0 \geq 0, \tag{1}$$

$$\begin{cases} d_0 - x_0 \geq 0, & d_0 - x_1 \geq 0, \\ d_0 - x_2 \geq 0, & d_0 - x_3 \geq 0, \\ d_0 - x_4 \geq 0, & d_0 - x_5 \geq 0, \\ d_0 - x_6 \geq 0, & d_0 - x_7 \geq 0. \end{cases} \tag{2}$$

When $d_0 = 0$ (the column is inactive), Eq. (1) is always true and 8 inequalities in Eq. (2) ensure that all related bytes are inactive. When $d_0 = 1$ (the column is active), Eq. (1) ensures that the sum of related bytes is at least 5 and Eq. (2) is always true.

The model is general and can be extended even if the toy cipher has more than 2 rounds. For example, valid propagation for the second round can be modeled by introducing another dummy variable d_1 and 9 more linear inequalities.

Besides the operations in the toy cipher, Lesamnta-LW-BC has the XOR operation in the generalized Feistel network. Here, we explain how to model the XOR operation. Let a, b, c be three bytes under the relationship of $a \oplus b = c$, and let three binary variables $x_a, x_b, x_c \in \{0, 1\}$ denote whether each of a, b, c is active or not. Then the propagation is invalid when the XOR of a non-active byte and an active byte results in the non-active byte, or the XOR of two non-active bytes results in the active byte. Namely, the following cases are invalid: $(x_a, x_b, x_c) = (1, 0, 0), (0, 1, 0), (0, 0, 1)$. (Note that the XOR of two active bytes can be active because a, b, c are byte values, not bit values.) Those can be modeled by replacing the coefficients '5' in Eqs. (1), (2) with '2' by introducing a new dummy variable. It is also possible to model them one by one with the following inequalities.

$$\begin{cases} -x_a + x_b + x_c \geq 0, \\ x_a - x_b + x_c \geq 0, \\ x_a + x_b - x_c \geq 0. \end{cases} \tag{3}$$

For example, in the top inequality in Eq. (3), the minimum value of the left-hand side is -1 and this occurs only when $(x_a, x_b, x_c) = (1, 0, 0)$. Hence, by restricting the sum to satisfy '≥ 0,' only the case with $(x_a, x_b, x_c) = (1, 0, 0)$ is removed from the solution space, while the other patterns will stay in the solution space. The second and third inequalities in Eq. (3) can be similarly explained to remove the other two invalid patterns.

2.2 Advancement of the MILP Model

Many follow-up works are available. One of the most relevant articles to our research is the simple combination of Matsui's search strategy [21] with MILP proposed by Zhang et al. [29].

An overall idea is to take into account the search results for $R-1$ rounds, $R-2$ rounds, $R-3$ rounds, etc when we search for the lower bound of the number of active S-boxes for R rounds. Let B_r and s be the lower bound of the number of active S-boxes for r rounds and the number of S-boxes per round, respectively. Then we have constraints that among sr S-boxes in the first r rounds, at least B_r S-boxes are active for $r = 1, 2, \ldots, R-1$, which is expressed as

$$\sum_{i=0}^{sr-1} x_i \geq B_r, \qquad \text{for } r = 1, 2, \ldots, R-1.$$

The same argument can be applied to the sr S-boxes in the last r rounds, which is expressed as $\sum_{i=s(R-r)}^{sR-1} x_i \geq B_r$ for $r = 1, 2, \ldots, R-1$.

Note that this method works efficiently only when the search of the bounds is easier when the number of rounds is smaller, and thus it is natural to assume that when we search for the bound for R rounds, we have already searched for the bounds up to $R-1$ rounds. This assumption is true for almost all the previous researches.

Example 1: In the above toy cipher, the lower (tight) bound of the number of active bytes for 1, 2, 3, and 4 rounds are 1, 5, 6, and 10, respectively. Suppose that we search for the bound for 5 rounds, where the objective function is "minimize $x_0 + x_1 + \cdots + x_{19}$." The method of Zhang et al. adds the following 8 constraints in addition to the framework by Mouha et al.

$$\begin{aligned} x_0 + x_1 + \cdots + x_3 &\geq 1, \\ x_0 + x_1 + \cdots + x_7 &\geq 5, \\ x_0 + x_1 + \cdots + x_{11} &\geq 6, \\ x_0 + x_1 + \cdots + x_{15} &\geq 10, \\ x_{16} + x_{17} + \cdots + x_{19} &\geq 1, \\ x_{12} + x_{17} + \cdots + x_{19} &\geq 5, \\ x_8 + x_{17} + \cdots + x_{19} &\geq 6, \\ x_4 + x_{17} + \cdots + x_{19} &\geq 10. \end{aligned}$$

This strategy enables us to evaluate significantly more rounds of Lesamnta-LW-BC than the simple application of the framework by Mouha et al.

2.3 Extension to Linear Trail Search

The model for the truncated differential trails can be converted into the truncated linear trail in which we only trace if each active S-box is involved in the linear trail or not, i.e. we do not care the actual linear masks for each S-box. It is particularly simple for AES because the SubBytes operation does not impact to the model and the ShiftRows and MixColumns operations have exactly the same effect in differential and linear cryptanalysis.

Regarding Lesamnta-LW-BC, the model for linear cryptanalysis is different from the one for differential cryptanalysis only in the branching and XOR operations in the generalized Feistel network. Let a, b, c be three bytes with a relationship and $x_a, x_b, x_c \in \{0, 1\}$ be binary variables to denote whether each of a, b, c is active or not.

XOR. Suppose that $a \oplus b = c$. In the linear cryptanalysis, if c is involved in the trail, then $a \oplus b$ must be in the trail to cancel the existence of c. Hence both a and b must be in the trail. If c is not involved, then a and b are not involved. Nemaly, the only valid patterns are $(x_a, x_b, x_c) = (0, 0, 0), (1, 1, 1)$. $x_a = x_b = x_c$ always holds, thus no additional variables and constraints need to be introduced. Note that this is exactly the same as the branching operation in the differential cryptanalysis.

Branching. Suppose that $a = b = c$ holds. If one of them is involved in the linear trail, then one of the other two must be in the trail to cancel each other. This is the same as the XOR operation in the differential cryptanalysis, thus can be modeled in the same fashion by applying Eq. (3).

3. Security Analysis of Lesamnta-LW-BC

3.1 New Bounds of the Number of Active S-Boxes

We first describe our model to search for the truncated differential trails of Lesamnta-LW-BC with the minimum number of active S-boxes. Variables related to the first round are shown in Fig. 5.

3.1.1 Variables

In round i, 8 binary variables from $x_{8(i-1)}$ to $x_{8(i-1)+7}$ are assigned to the right most input branch. Similarly, 8 variables from x_{8i}, $x_{8(i+1)}$, and $x_{8(i+2)}$ are assigned to the second right most, second left most, and the left most branches, respectively. 8 binary variables from $y_{8(i-1)}$ are introduced to describe whether each byte after MixColumns is active or inactive. In addition, two dummy variables $d_{(i-1)}$ and $e_{(i-1)}$ are introduced to efficiently model MixColumns for the left and right columns. In summary, the r-round transformation is modeled by defining $32 + 8r$ variables $x_0, \ldots, x_{32+8r-1}$, $8r$ variables $y_0, \ldots, y_{8r-1}$, r variables $d_0, \ldots, d_{r-1}$, and r variables $e_0, \ldots, e_{r-1}$.

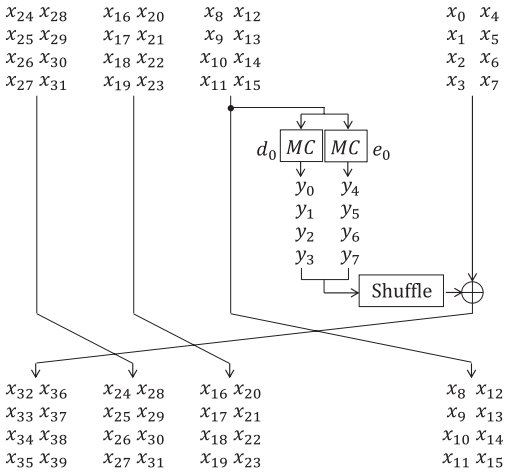

Fig. 5 All related variables to model the first round of Lesamnta-LW-BC.

3.1.2 Objective Function

For round i where $i = 1, 2, \ldots, r$, 8 bytes denoted by $x_{8i}, x_{8i+1}, \ldots, x_{8i+7}$ go through the S-boxes. The objective function for r rounds is represented as "minimize $\sum_{i=8}^{8r+7} x_i$."

3.1.3 Constraint Linear Inequalities

MixColumns in round i from $x_{8i}, x_{8i+1}, x_{8i+2}, x_{8i+3}$ to $y_{8(i-1)}, y_{8(i-1)+1}, y_{8(i-1)+2}, y_{8(i-1)+3}$, are modeled by 9 linear inequalities and an additional variable d_{i-1} by Eqs. (1) and (2). The same applies to the other column.

The XOR operation from G function's output to the right most branch is model by applying Eq. (3) to each byte. Considering the Shuffle operation, we model the XOR of $y_{8i+\mathrm{Shuffle}(j)}, x_{8i+j}, x_{8i+j+32}$ for $j = 0, 1, \ldots, 7$ in round i.

The model for 3-round Lesamnta-LW-BC is fully described in Fig. A· 10.

3.1.4 Evaluation Results

By solving MILP, we obtained the bounds given in Table 1. Compared to the previous lower bound evaluated by the designers, which is 24 for 24 rounds, more active S-boxes is ensured, which is 33 for 24 rounds. Hence, Lesamnta-LW-BC is more secure against differential cryptanalysis than originally expected. More interestingly, 24 active S-boxes can be ensured only with 18 rounds. By applying the same scale, the total number of rounds may be reduced to 48 rounds ($= 64 \times 18/24$) by preserving the same security level as was originally expected. The bound is tight, i.e. there exist truncated differential trails confirming the bound. As an example, we show a 24-round trail with 33 active S-boxes in Fig. 6.

Table 1 Tight bounds of the number of active S-boxes of Lesamnta-LW-BC.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bounds	0	0	0	1	1	1	2	6	6	7	11	13	14	18	20	21
Rounds	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bounds	22	25	25	26	27	29	30	33	34	35	39	41	42	46	47	49
Rounds	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Bounds	50	50	51	54	55	56	58	61	62	63	67	69	70	73	73	75
Rounds	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Bounds	76	78	79	81	83	84	86	89	90	91	95	97	98	98	100	103

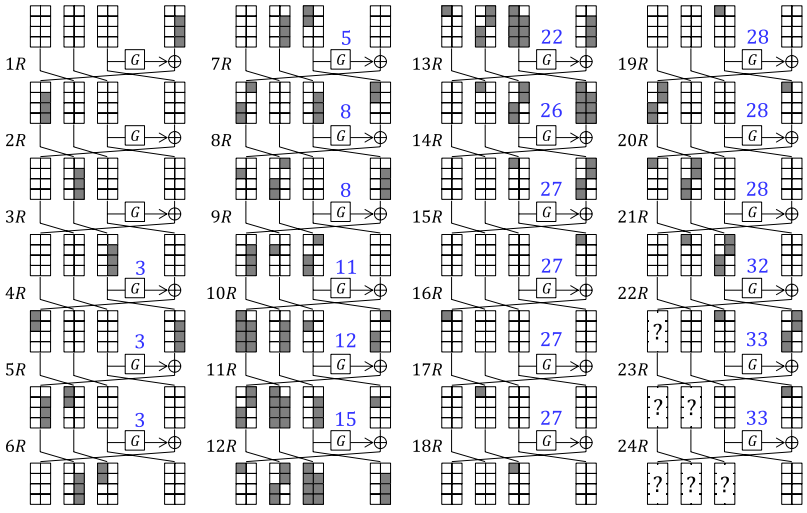

Fig. 6 24-Round Truncated Differential Trail with 33 Active S-boxes. Blue numbers by the G function represent the accumulated number of active S-boxes from the first round. Active patterns of the update states in the last three rounds are not related to the number of active S-boxes, which is represented by the '?' symbol.

The block size of Lesamnta-LW-BC is 256 bits. The bound to ensure 43 active S-boxes is interesting because the maximum differential characteristic probability becomes smaller than 2^{-256} after going through 43 active S-boxes ($2^{-6\times 43} = 2^{-258}$). This motivates us to extend the evaluation to full rounds. We found that 43 active S-boxes are ensured after 30 rounds.

3.1.5 Computational Time of the Tool

Only with the framework by Mouha et al. [22], we could obtain the bounds up to 48 rounds. The computation for 47 and 48 rounds took 370,078 seconds (about 103 hours) and 247,771 seconds (about 69 hours), respectively, and we gave up evaluating more rounds because of too much computation time.

We then introduced the method by Zhang et al. [29]. The computational time for 47 and 48 rounds decreased to 19,913 seconds (5.5 hours) and 15,117 seconds (4.2 hours) respectively. This improvements allowed us to derive the bounds for the full rounds. The heaviest instance was for 61 rounds, which required 1,269,330 seconds (352.6 hours or 14.7 days) to find the tight bound.

3.2 Security Analysis of Shuffle Operation

Shuffle of Lesamnta-LW is originally from the byte-shuffling function in MUGI [27]. The rational of its choice seems to rely on the fact that MUGI has been specified in ISO/IEC 18033-4:2005 [16]. However, no specific security analysis was given to explain the validity of the choice. This motivates us to evaluate the security of various choices of the Shuffle operation.

The original choice is a 2-byte-wise permutation, which may be because Lesamnta-LW was designed to be efficient in micro-controllers with 16-bit registers. In this section, we relax this constraint and consider any byte-wise permutation to investigate the existence of choices with higher security.

3.2.1 The Number of Crossing Bytes N_X

In each round, outside G, the cancellation of differences only occurs between the right most input state and the output of the G function. Moreover, MixColumns in the G function has the property that the active-byte-position patterns after MixColumns only depend on the weight of the

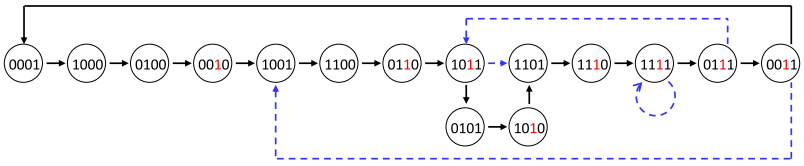

Fig. 7 Possible Differential Propagation of Branch-wise Truncated Differential Trails for 4-Branch GFN. Nodes with two outgoing arrows can propagate to two differentials depending on the cancellation of the difference. Dotted lines in blue show the propagation when differences are not cancelled (even though it is possible). Nodes with red color increase the number of active S-boxes.

Table 2 Truncated differential trail for $N_X = 4$ activating only half of the state.

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
000L	L000	0L00	00L0	R00L	LR00	0LR0	L0LR	0L0L	L0L0	RL0L	LRL0	RLRL	0RLR	00RL	000R
000L	L000	0L00	00L0	R00L	LR00	0LR0	L0LR	RL0L	LRL0	RLRL	0RLR	00RL	000R		

input differences. Stating differently, any input truncated difference having the same number of active S-boxes can produce the same output-difference patterns through MixColumns. This property makes the position of input differences of MixColumns equivalent with respect to the number of active S-boxes.

Example 2: Two shuffles parameters "45230167" and "45236701" are equivalent with respect to the bounds of the number of active S-boxes, because two parameters are only different in the byte positions of '0', '1', '6' and '7' inside the right column. Indeed we searched for the bounds for "45236701" up to 32 rounds, and they match the ones in Table 1.

Given that the position inside the column is irrelevant, the important issue is the number of bytes that move from one side of the column to the other side of the column through the Shuffle operation. We call those bytes "crossing bytes" and denote its number by N_X. For example, N_X of the original Shuffle "45230167" is 2 because the byte positions 0 and 1 move to the right column, and similarly byte positions 4 and 5 move to the left column.

The range of N_X is from 0 to 4 because 1 column consists of 4 bytes. It is obvious that $N_X = 0$ is insecure because 16 bytes located in the left half of each state and the other 16 bytes located in the right half of each state never interact each other. In the following, we will explain that all the parameters having $N_X = 0$, 1, or 3 allow efficient truncated differential trails, thus choosing $N_X = 2$ is best both in security as well as implementation efficiency.

3.2.2 Truncated Differential Trails General to Type-1 4-Branch GFN

Before we explain the analysis for $N_X = 0$, 1, or 3, we describe truncated differential trails in the branch-wise level that are general to type-1 4-branch GFN where the G function is bijective.

By only considering whether each state is active or inactive, the 4 branch state only has 15 possible patterns 0001, 0010, ..., 1111. Note that 0000 never appears in the single-key differential trail. When the active patterns of the round input does not allow the differential cancellation, the active patterns of the output is uniquely determined. For example, when the input pattern is 0001, the output pattern is always 1000. When the differential cancellation occurs in the round, there are two possible output patterns. For example, when the input pattern is 1011 the output pattern is either 1101 (without difference cancellation) or 0101 (with difference cancellation). By applying the same analysis for all 15 patterns, we can describe all possible differential propagation in the state-change diagram, which is shown in Fig. 7.

For a large number of rounds, the differential trail will be iterative in the branch-wise level. Most of the patterns that do not increase the number of active bytes (states having 0 in the second right most branch) only exist in long iterations. Indeed the ratio of the number of rounds with active S-boxes to the number of rounds for the whole iteration becomes smallest (8/15) or the second smallest (7/13) when 1-active branch states are included in the iteration. Hence, in the branch level, the 15-round iterative trail with ratio 8/15 is the most powerful. However, the actual number of rounds depends on the details of G or the parameter of Shuffle in G. In the following, we look more details for each N_X.

3.2.3 Existence of Efficient Trails with $N_X = 4$

In this parameter, 4 bytes in the left (resp. right) columns move to the right (resp. left) column, where the order inside each column can be any order. "47653102" is an example.

We found that $N_X = 4$ allows attackers to construct truncated differential trails only by activating one of the columns for each state. Let L and R be the state that has some active bytes in the left and right column, respectively. Then the 13-round and 15-round generic branch-wise truncated differentials can be instantiated as shown in Table 2. The number of active bytes in L and R must be a valid relationship over MixColumns, namely the sum of the number of active bytes is 5. To explain the above 15-round trail as an example, it takes L and R as input of the G function 5 times and 3 times, respectively. Hence by setting L an R to

Table 3 Tight bounds of the number of linearly active S-boxes of Lesamnta-LW-BC.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bounds	0	0	0	1	1	1	2	6	6	7	11	12	13	14	15	16
Rounds	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bounds	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

have only 1 active byte and 4 active bytes respectively, the above 15-round trails will have $1 \times 5 + 4 \times 3 = 17$ active S-boxes. However, although the above 15-round trail iterative in the branch-wise level, the second iteration will start with R. Hence 2 iterations of the 15-round trail will take each of L and R as input 8 times and have $8 \times 5 = 40$ active S-boxes. Asymptotically the number of active bytes for $30x$ rounds for a positive integer x is $40x$. Note that as shown in Table 1, the original shuffle parameter of Lesamnta-LW-BC ensures at 97 active S-boxes for 60 rounds, which is significantly larger than the case with $N_X = 4$.

3.2.4 Existence of Efficient Trails with $N_X = 1$ or $N_X = 3$

In those parameters, permutations are no longer 2-byte-wise. Hence implementation efficiency in 16-bit CPUs will decrease. One of the most surprising analysis in Sect. 3.2 is that by introducing such byte-wise permutation, not only the efficiency but also the security will decrease.

The analysis is similar to the case with $N_X = 4$. Indeed attackers can build the same trails in Table 2 with slightly more constraints for L and R. The strategy is to set L and R be 2-byte active and 3-byte active, respectively (or its vise verse). Namely, for $N_X = 1$, the behavior of the differential propagation is the same as $N_X = 0$ except for the crossing 1 byte. For $N_X = 3$, the behavior of the differential propagation is the same as $N_X = 4$ except for the staying 1 byte. By avoiding both of L and R be fully active, attackers can ensure that the crossing 1 byte for $N_X = 1$ and the staying 1 byte for $N_X = 3$ is always inactive. As a consequence, trails in Table 2 can be instantiated with 2- and 3-active byte state. Moreover, the asymptotic property having $40x$ active bytes in $30x$ rounds is the same as the parameters with $N_X = 4$.

We emphasize that the original specification of Lesamnta-LW-BC has $N_X = 2$, which is the best against differential cryptanalysis. For $N_X = 2$, as demonstrated in Fig. 6, it is inevitable to activate both columns simultaneously to construct 15-round or 13-round iterative trails.

3.3 Linear Cryptanalysis of Lesamnta-LW-BC

Given the model for the differential cryptanalysis, to change the model for the linear cryptanalysis is simple. The variables used to model the first round are shown in Fig. 8, and the constraints are generated as explained in Sect. 2.3. The bounds derived by the MILP up to 32 rounds are given in Table 3.

There is no analysis in the original document of Lesamnta-LW, which is reasonable because Lesamnta-LW is originally a hash function and the impact of linear cryptanalysis on the hash function is not clear. Compared to the differential bound in Table 1, the number of active S-boxes is the same up to 11 rounds. However, from 12 rounds, the number of linearly active S-boxes is always the same as the number of rounds. We investigated the details of the linear trails and detected that Lesamnta-LW-BC allows a 1-round iterative truncated linear trail, which is shown in Fig. 9.

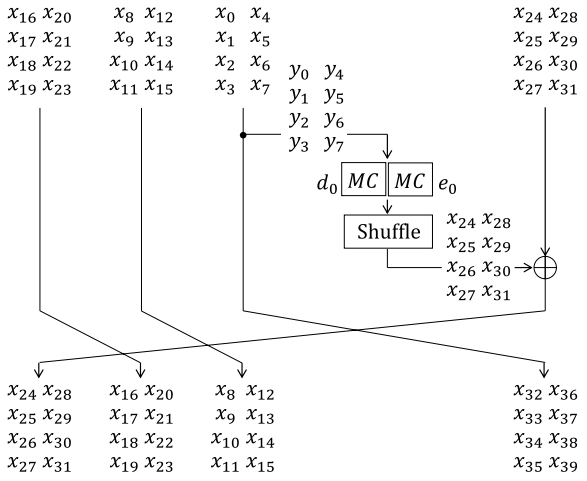

Fig. 8 All related variables to model the first round of Lesamnta-LW-BC in linear cryptanalysis.

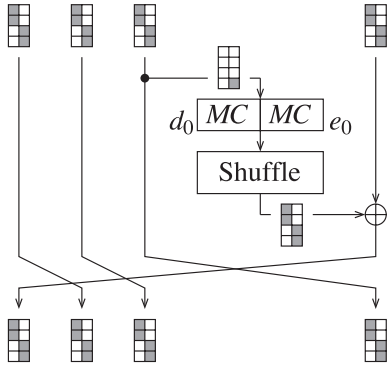

Fig. 9 1-round iterative truncated linear trail.

The analysis shows that to ensure 24 linearly active bytes, the round function needs to be iterated at least 24 times, which happens to match the original evaluation by the designers for differential cryptanalysis. However, we note that the complexity evaluation is different between differential and linear cryptanalysis, and 21 rounds is sufficient to resist linear cryptanalysis under 128-bit security.

The analysis is as follows. The largest bias of the AES S-box is know to be 2^{-4}. From the piling-up lemma [20], the largest bias when X active S-boxes are combined is com-

puted as $2^{X-1} \cdot (2^{-4})^X$. When the bias is p, the data complexity becomes p^{-2} and we want it to be smaller than 2^{-128}. Hence, the number of rounds C that will ensure the data complexity of less than 2^{128} is computed as

$$2^{X-1} \cdot (2^{-4})^X > 2^{-64}, \tag{4}$$

which is $-3X - 1 > -64$. Hence, $X < 21$ rounds is the condition to apply the linear attack with data complexity of less than 2^{128}.

4. New PRF Mode Based on Lesamnta-LW-BC

In this section, we propose a new PRF mode for Lesamnta-LW-BC, which we call LRF. It achieves double throughput compared to the previous modes [9], [13]. Its specification is described in Sect. 4.1. The security in the standard model and in the ideal model is discussed in Sect. 4.2 and Sect. 4.3, respectively.

For a set $\mathcal{S}$, let $\mathcal{S}^* := \bigcup_{i=0}^{\infty} \mathcal{S}^i$ and $\mathcal{S}^+ := \bigcup_{i=1}^{\infty} \mathcal{S}^i$. Let $\mathcal{F}_{\mathcal{D},\mathcal{R}}$ be the set of all functions with their domain and range $\mathcal{D}$ and $\mathcal{R}$, respectively. Let $\mathcal{P}_{\mathcal{D}}$ be the set of all permutations on $\mathcal{D}$. Let $s \twoheadleftarrow \mathcal{S}$ represent substitution of an element chosen uniformly at random from $\mathcal{S}$ to s. For $\{0,1\}$-sequences x and y, let $x\|y$ be their concatenation. Let ε be the sequence of length 0.

4.1 Description of LRF

Let $E : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ be Lesamnta-LW-BC with its key space $\{0,1\}^n$. The proposed PRF mode $\mathrm{LRF}^E : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ with its key space $\{0,1\}^n$ is defined as follows. For a given input $(K, M) \in \{0,1\}^n \times \{0,1\}^*$, it first applies injective padding pad to M and gets a sequence of length a positive multiple of $2n$. Let $M \leftarrow \mathsf{pad}(M)$ and $m := |M|/(2n)$. It divides M into $2n$-bit blocks so that $M = M[1]\|M[2]\|\cdots\|M[m] \in (\{0,1\}^{2n})^m$. Then, it computes $V[i] \leftarrow E(V_0[i-1], M_0[i]\|(M_1[i] \oplus V_1[i-1]))$ for $1 \le i \le m-1$, and $V[m] \leftarrow E(V_0[m-1] \oplus c, M_0[m]\|(M_1[m] \oplus V_1[m-1]))$, where $V_0[0] \leftarrow K$, $V_1[0] \leftarrow 0^n$, $V[i] := V_0[i]\|V_1[i]$ and $M[i] := M_0[i]\|M_1[i]$ such that $|V_0[i]| = |V_1[i]| = |M_0[i]| = |M_1[i]| = n$. Finally, it returns $V[m]$ as its output. LRF^E is also depicted in Fig. 3.

During the discussion of security of LRF^E, without loss of generality, it is assumed that $\mathrm{LRF}^E : \{0,1\}^n \times (\{0,1\}^{2n})^+ \to \{0,1\}^n$ since any injective padding works for LRF^E.

4.2 Security in the Standard Model

4.2.1 Definition

Let $f \in \mathcal{F}_{\mathcal{K}\times\mathcal{D},\mathcal{R}}$ be a keyed function with its key space $\mathcal{K}$. For any $K \in \mathcal{K}$, $f_K(\cdot) := f(K, \cdot) \in \mathcal{F}_{\mathcal{D},\mathcal{R}}$. Let $\mathbf{D}$ be an adversary against f. $\mathbf{D}$ has oracle access to functions in $\mathcal{F}_{\mathcal{D},\mathcal{R}}$ and outputs 0 or 1. Then, the prf-advantage of $\mathbf{D}$ against f is defined by

$$\mathrm{Adv}_f^{m\text{-prf}}(\mathbf{D}) := \left|\Pr[\mathbf{D}^{f_{K_1},\ldots,f_{K_m}} = 1] - \Pr[\mathbf{D}^{\rho_1,\ldots,\rho_m} = 1]\right|,$$

where K_j's and ρ_j's are chosen uniformly and independently at random from $\mathcal{K}$ and $\mathcal{F}_{\mathcal{D},\mathcal{R}}$, respectively. In particular, $\mathrm{Adv}_f^{\mathrm{prf}}(\mathbf{D}) := \mathrm{Adv}_f^{1\text{-prf}}(\mathbf{D})$.

If f is a keyed permutation on $\mathcal{D}$ and $\mathbf{D}$ has oracle access to m permutations in $\mathcal{P}_{\mathcal{D}}$, then the prp-advantage of $\mathbf{D}$ against f is denoted by $\mathrm{Adv}_f^{m\text{-prp}}(\mathbf{D})$. $\mathrm{Adv}_f^{\mathrm{prp}}(\mathbf{D})$ is defined similarly.

A PRF under related-key attacks is formalized by Bellare and Kohno [2]. Let $\Phi \subset \mathcal{F}_{\mathcal{K},\mathcal{K}}$ be a set of related-key-derivation functions and let $\mathsf{rk} \in \mathcal{F}_{\Phi\times\mathcal{K},\mathcal{K}}$ be a function such that $\mathsf{rk}(\varphi, K) := \varphi(K)$. Let $\mathbf{D}$ be an adversary against $f \in \mathcal{F}_{\mathcal{K}\times\mathcal{D},\mathcal{R}}$. $\mathbf{D}$ has oracle access to the functions of the form $g(\mathsf{rk}(\cdot, K), \cdot)$, where $g \in \mathcal{F}_{\mathcal{K}\times\mathcal{D},\mathcal{R}}$ and $K \in \mathcal{K}$. $g(\mathsf{rk}(\cdot, K), \cdot)$ receives $(\varphi, x) \in \Phi \times \mathcal{D}$ as a query and returns $g(\varphi(K), x)$. Let $g[K] := g(\mathsf{rk}(\cdot, K), \cdot)$ to make the notation simpler. The prf-rka-advantage of $\mathbf{D}$ making a Φ-related-key attack (Φ-RKA) against f is defined by

$$\mathrm{Adv}_{\Phi,f}^{m\text{-prf-rka}}(\mathbf{D}) := \left|\Pr[\mathbf{D}^{f[K_1],\ldots,f[K_m]} = 1] - \Pr[\mathbf{D}^{\rho_1[K_1],\ldots,\rho_m[K_m]} = 1]\right|,$$

where K_j's and ρ_j's are chosen uniformly and independently at random from $\mathcal{K}$ and $\mathcal{F}_{\mathcal{K}\times\mathcal{D},\mathcal{R}}$, respectively. In particular, $\mathrm{Adv}_{\Phi,f}^{\mathrm{prf\text{-}rka}}(\mathbf{D}) := \mathrm{Adv}_{\Phi,f}^{1\text{-prf-rka}}(\mathbf{D})$. $\mathrm{Adv}_{\Phi,f}^{m\text{-prp-rka}}(\mathbf{D})$ and $\mathrm{Adv}_{\Phi,f}^{\mathrm{prp\text{-}rka}}(\mathbf{D})$ are defined similarly.

4.2.2 Result

The following theorem implies that LRF^E is a PRF if the underlying block cipher E is a PRP under $\{\mathsf{id}, \mathsf{ac}\}$-related key attacks, where id is the identity permutation over $\{0,1\}^n$ and ac is a permutation over $\{0,1\}^n$ such that $\mathsf{ac}(K) := K \oplus c$. Let T_E represent the time to compute E.

Theorem 1: Let $\mathbf{A}$ be any prf-adversary against LRF^E. For $\mathbf{A}$, let t be its running time, q be the number of its queries, and ℓ be the upper bound on the number of message blocks in each of its queries. Then, there exists some prp-adversary $\mathbf{B}$ making a related-key attack on E such that

$$\mathrm{Adv}_{\mathrm{LRF}^E}^{\mathrm{prf}}(\mathbf{A}) \le \ell q \cdot \mathrm{Adv}_{\{\mathsf{id},\mathsf{ac}\},E}^{\mathrm{prp\text{-}rka}}(\mathbf{B}) + \ell q^2/2^{2n+1}.$$

$\mathbf{B}$ runs in time at most about $t + O(\ell q \mathrm{T}_E)$ and makes at most q queries.

Theorem 1 follows from the two lemmas given below.

Lemma 1: Let $\mathbf{A}$ be any prf-adversary against LRF^E. For $\mathbf{A}$, let t be its running time, q be the number of its queries, and ℓ be the upper bound on the number of message blocks in each of its queries. Then, there exists some prf-adversary $\mathbf{B}$ making a related-key attack on E such that

$$\mathrm{Adv}_{\mathrm{LRF}^E}^{\mathrm{prf}}(\mathbf{A}) \le \ell \cdot \mathrm{Adv}_{\{\mathsf{id},\mathsf{ac}\},E}^{q\text{-prf-rka}}(\mathbf{B}).$$

$\mathbf{B}$ runs in time at most about $t + O(\ell q \mathrm{T}_E)$ and makes at most

q queries.

Proof : For $M = M[1]\|M[2]\|\cdots\|M[m]$, where $M[j] \in \{0,1\}^{2n}$ for $1 \le j \le m$, let $M[j_1, j_2] := M[j_1]\|M[j_1+1]\|\cdots\|M[j_2]$ for $1 \le j_1 \le j_2 \le m$ and $M[j_1, j_2] := \varepsilon$ if $j_1 > j_2$. For $i \in \{0, 1, \ldots, \ell\}$, let $\Gamma_i : (\{0,1\}^{2n})^+ \to \{0,1\}^{2n}$ be a random function such that

$$\Gamma_i(M) := \begin{cases} \gamma_0(M) & \text{if } m \le i, \\ \mathrm{LRF}^E(\gamma_1(M[1,i]), M[i+1,m]) & \text{otherwise,} \end{cases}$$

where γ_0 and γ_1 are independent random functions such that γ_0 is chosen uniformly at random from $\mathcal{F}_{(\{0,1\}^{2n})^+,\{0,1\}^{2n}}$ and γ_1 is chosen uniformly at random from $\{\gamma \,|\, \gamma \in \mathcal{F}_{(\{0,1\}^{2n})^*,\{0,1\}^{2n}} \wedge \gamma(\varepsilon) \in \{0,1\}^n \times \{0^n\}\}$. Let $P_i := \Pr[\mathbf{A}^{\Gamma_i} = 1]$. Then, since each query made by $\mathbf{A}$ has at most ℓ message blocks,

$$\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{LRF}^E}(\mathbf{A}) = \left|P_0 - P_\ell\right|.$$

Let us consider the following prf-adversary $\mathbf{B}$ making a $\{\mathsf{id}, \mathsf{ac}\}$-RKA against E. $\mathbf{B}$ is given access to q oracles, which are either $E[K_1], \ldots, E[K_q]$ or $\rho_1[K_1], \ldots, \rho_q[K_q]$, where K_j's and ρ_j's are chosen independently and uniformly at random from $\{0,1\}^n$ and $\mathcal{F}_{\{0,1\}^n\times\{0,1\}^{2n},\{0,1\}^{2n}}$, respectively. $\mathbf{B}$ simulates two independent random functions β_0 and β_1 via lazy sampling: β_0 is chosen uniformly at random from $\mathcal{F}_{(\{0,1\}^{2n})^+,\{0,1\}^{2n}}$ and β_1 is chosen uniformly at random from $\{\beta \,|\, \beta \in \mathcal{F}_{(\{0,1\}^{2n})^*,\{0,1\}^n} \wedge \beta(\varepsilon) = 0^n\}$. $\mathbf{B}$ first samples $r \in \{1, \ldots, \ell\}$ uniformly at random. Then, $\mathbf{B}$ runs $\mathbf{A}$. Finally, $\mathbf{B}$ outputs the same output as $\mathbf{A}$.

For $1 \le k \le q$, let $M^{(k)}$ be the k-th query made by $\mathbf{A}$. Suppose that $M^{(k)}$ has m blocks. If $m \ge r$, then $\mathbf{B}$ makes a query to its $\mathsf{p}(k)$-th oracle, where $\mathsf{p}(k) \leftarrow \mathsf{p}(k')$ if there exists a previous query $M^{(k')}$ $(k' < k)$ such that $M^{(k')}[1, r-1] = M^{(k)}[1, r-1]$, and $\mathsf{p}(k) \leftarrow k$ otherwise. $\mathbf{B}$ asks to its $\mathsf{p}(k)$-th oracle $(\mathsf{ac}, X^{(k)})$ if $m = r$ and $(\mathsf{id}, X^{(k)})$ if $m \ge r+1$, where $X^{(k)} := M^{(k)}_0[r]\|(\beta_1(M^{(k)}[1, r-1]) \oplus M^{(k)}_1[r])$. Then, in response to $M^{(k)}$, $\mathbf{B}$ returns

$$\begin{cases} \beta_0(M^{(k)}) & \text{if } m < r, \\ g_{\mathsf{p}(k)}(K_{\mathsf{p}(k)} \oplus c, X^{(k)}) & \text{if } m = r, \\ \mathrm{LRF}^E(g_{\mathsf{p}(k)}(K_{\mathsf{p}(k)}, X^{(k)}), M^{(k)}[r+1, m]) & \text{if } m > r, \end{cases}$$

where $g_{\mathsf{p}(k)}$ is either E or $\rho_{\mathsf{p}(k)}$, which depends on $\mathbf{B}$'s oracles.

Suppose that $\mathbf{B}$'s oracles are $E[K_1], \ldots, E[K_q]$. Then, since $K_{\mathsf{p}(k)}$ can be regarded as an output of a random function for an input $M^{(k)}[1, r-1]$, $\mathbf{B}$ implements Γ_{r-1} for $\mathbf{A}$. Thus,

$$\Pr[\mathbf{B}^{E[K_1],\ldots,E[K_q]} = 1] = \frac{1}{\ell}\sum_{i=1}^{\ell} P_{i-1} \quad .$$

Suppose that $\mathbf{B}$'s oracles are $\rho_1[K_1], \ldots, \rho_q[K_q]$. Then, since $\rho_{\mathsf{p}(k)}(K_{\mathsf{p}(k)} \oplus c, \cdot)$ and $\rho_{\mathsf{p}(k)}(K_{\mathsf{p}(k)}, \cdot)$ are independent, $\mathbf{B}$ implements Γ_r for $\mathbf{A}$. Thus,

$$\Pr[\mathbf{B}^{\rho_1[K_1],\ldots,\rho_q[K_q]} = 1] = \frac{1}{\ell}\sum_{i=1}^{\ell} P_i.$$

From the discussions above,

$$\begin{aligned} &\mathrm{Adv}^{q\text{-prf}}_{E}(\mathbf{B}) \\ &= \left|\Pr[\mathbf{B}^{E[K_1],\ldots,E[K_q]} = 1] - \Pr[\mathbf{B}^{\rho_1[K_1],\ldots,\rho_q[K_q]} = 1]\right| \\ &= \frac{1}{\ell}\mathrm{Adv}^{\mathrm{prf}}_{\mathrm{LRF}^E}(\mathbf{A}). \end{aligned}$$

$\mathbf{B}$ makes at most q queries and runs in time at most about $t + O(\ell q \mathrm{T}_E)$. □

Lemma 2: Let $\mathbf{A}$ be any prf-adversary making a related-key attack on E. For $\mathbf{A}$, let t be its running time and q be the number of its queries. Then, there exists some prp-adversary $\mathbf{B}$ making a related-key attack on E such that

$$\mathrm{Adv}^{m\text{-prf-rka}}_{\{\mathsf{id},\mathsf{ac}\},E}(\mathbf{A}) \le m \cdot \mathrm{Adv}^{\mathrm{prp\text{-}rka}}_{\{\mathsf{id},\mathsf{ac}\},E}(\mathbf{B}) + q^2/2^{2n+1}.$$

$\mathbf{B}$ runs in time at most about $t + O(q\mathrm{T}_E)$ and makes at most q queries.

The proof is omitted since it is similar to that of Lemma 2 in [10].

For the upper bound of Theorem 1, due to the exhaustive key search, $\mathrm{Adv}^{\mathrm{prp\text{-}rka}}_{\{\mathsf{id},\mathsf{ac}\},E}(\mathbf{B}) = \Omega(t_{\mathbf{B}}/2^n)$, where $t_{\mathbf{B}}$ is the running time of $\mathbf{B}$. It seems reasonable to assume that $t_{\mathbf{B}} = \Omega(\ell q)$, which suggests that Theorem 1 guarantees at most $(n/2)$-bits of security. The exhaustive key search is generic and does not exploit the internal structure of the target block cipher, however, and the result in the next subsection implies that the proposed PRF mode may have n-bits of security against such kind of generic attacks.

4.3 Security in the Ideal Model

In this section, the indistinguishability of LRF^E from a random oracle is discussed in the ideal cipher model. Namely, E is an ideal block cipher chosen uniformly at random from the set of the keyed permutations over $\{0,1\}^{2n}$ with their key space $\{0,1\}^n$.

4.3.1 Definition

Let C^E be a construction of a keyed function using the ideal block cipher E. Let C^E_K be C^E with its key K chosen uniformly at random. Let R be a random oracle chosen uniformly at random from all the functions which have the same domain and range as C^E. Then, the indistinguishability advantage of an adversary $\mathbf{A}$ against C^E is defined by

$$\mathrm{Adv}^{\mathrm{ind}}_{C^E}(\mathbf{A}) := \left|\Pr[\mathbf{A}^{C^E_K,E,E^{-1}} = 1] - \Pr[\mathbf{A}^{R,E,E^{-1}} = 1]\right|.$$

4.3.2 Result

The following theorem implies that LRF^E has the n-bit indistinguishability in the ideal cipher model. Thus, LRF^E is secure up to the birthday bound of the size of its internal state against generic distinguishing attacks without exploiting the internal structure of E.

Theorem 2: Let $\mathbf{A}$ be any adversary against LRF^E. For $\mathbf{A}$,

let q_e and q_d be the numbers of its encryption and decryption queries to E, respectively, q be the number of its queries to the oracle accepting variable-length inputs, and σ be the total number of message blocks in the q queries. Then,

$$\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{LRF}^E}(\mathbf{A}) \leq \frac{(\sigma + q_e + q_d)^2}{2^{2n}} + \frac{\sigma q}{2^{2n}} + \frac{q_e + q_d}{2^n}.$$

The proof is given in Appendix B.

5. Discussion and Conclusion

In this section, we discuss more observations about Lesamnta-LW.

5.1 Related-Key Security of Lesamnta-LW-BC

The key schedule of Lesamnta-LW-BC adopts the byte-wise structure, thus the MILP model in Sect. 3 can be extended to related-key. It turned out that the related-key security of Lesamnta-LW-BC cannot be obtained only by analyzing whether each byte is active or not.

Only by considering whether each byte is active or not, attackers can build the related-key differential trail with no active S-box in any number of rounds. This is achieved by activating all the key bytes and the left half of the round function state. As shown in Fig. 10, by assuming no cancellation in the key schedule, 4 bytes of round keys are always active and this can be cancelled with 4-byte difference in the state. Then, the input to SubBytes is inactive in all the rounds.

However, this cannot be exploited by actual Lesamnta-LW-BC because active bytes do not always cancel each other during the round-key addition. Indeed, in the trail in Fig. 10, difference in the round function state never changes while the difference values in the key state must change due to the GFN transformation.

5.2 Insecurity of Constructions Similar to LRF

One may wonder whether there exist other methods to absorb 256-bit input per block cipher call. Indeed, we considered several other constructions, however it turned out that many of them would not be as secure as LRF. Here we discuss two such constructions as failure examples.

5.2.1 Insecure Construction 1

The first construction is depicted in Fig. 11. It simply applies boosting-MD MAC [28] to the previous PRF in the upper side of Fig. 2 in order to absorb the 256-bit input per block and it tries to achieve the security by truncating the last output.

This construction is distinguished from a 128-bit random function only with 2^{64} queries by an extension attack though its internal state size is 256 bits. The attack is depicted in Fig. 12 and its procedure is as follows.

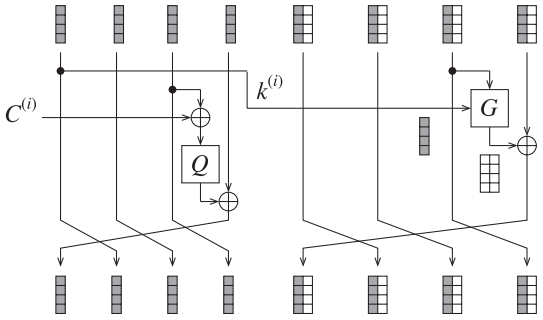

Fig. 10 Key schedule function and byte-wise related-key differential trail.

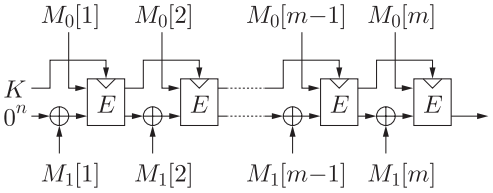

Fig. 11 Insecure construction 1 (IC1).

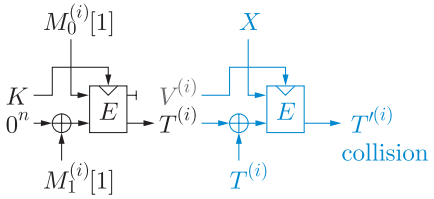

Fig. 12 Distinguisher on IC1.

1. Make 2^{64} queries by choosing distinct 1-block messages $M_0^{(i)}[1]\|M_1^{(i)}[1]$ for $i = 1, 2, \ldots, 2^{64}$ to observe the output $T^{(i)}$. (black in Fig. 12)
2. Fix $M_0[2]$ to some value denoted by X.
3. For each i, set $M_1^{(i)}[2]$ to $T^{(i)}$. Then, the attacker queries $M_0^{(i)}[1]\|M_1^{(i)}[1]\|X\|T^{(i)}$ to observe the corresponding output $T'^{(i)}$. (blue in Fig. 12)
4. There should be a collision of 128-bit output $T'^{(i)}$. Let i_1 an i_2 be the indices of colliding pair. Then, choose new X and check if $M_0^{(i)}[1]\|M_1^{(i)}[1]\|X\|T^{(i)}$ for replaced X collide again with $i = i_1, i_2$.

2^{64} queries are made at Step 1, which generates a collision in the upper half of the block cipher output (denoted by $V^{(i)}$ in grey in Fig. 12, which is undisclosed to the attacker). Hence after adjusting the lower half of the second block cipher input by $M_1^{(i)}[2] \leftarrow T^{(i)}$, the collision of $V^{(i)}$ is preserved to the collision of $T'^{(i)}$.

5.2.2 Insecure Construction 2

The second construction applies the MDP (public permutation before the last block) in the lower half of the internal state. The construction is depicted in Fig. 13.

This construction can also be distinguished from a 256-bit random function only with 2^{64} queries for Lesamnta-LW-

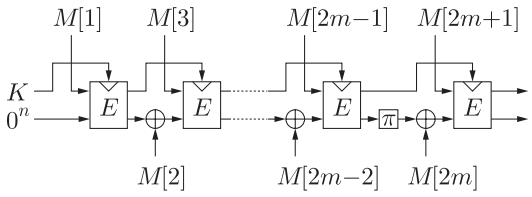

Fig. 13 Insecure Construction 2 (IC2).

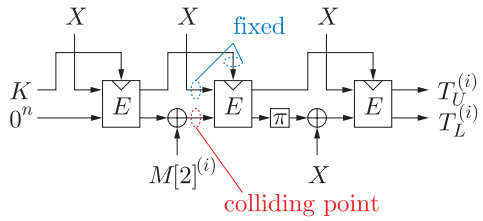

Fig. 14 3-block query.

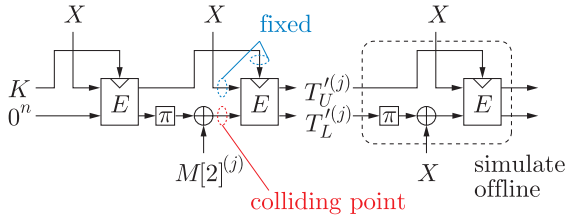

Fig. 15 2-block query plus offline extension.

BC by a bit different attack procedure.

1. Fix $M[1], M[3], M[4], M[5]$ to some value X. Query $X\|M[2]^{(i)}\|X\|X$ for $i = 1, 2, \ldots, 2^{64}$ to obtain 256-bit $T_U^{(i)}\|T_L^{(i)}$, where $T_U^{(i)}$ and $T_L^{(i)}$ are the upper and the lower halves of the function's output, respectively. (Fig. 14)
2. Query $X\|M[2]^{(j)}\|X$ for $j = 1, 2, \ldots, 2^{64}$ to obtain 256-bit $T_U'^{(j)}\|T_L'^{(j)}$. Moreover, simulate the computation for the extension with additional input $X\|X$ offline. Namely, compute $E_{T_U'^{(j)}}(X\|(\pi(T_L'^{(j)}) \oplus X))$. (Fig. 15)
3. Check if there exists a collision between the results of the above two steps.

By fixing $M[1]$ and $M[3]$, the key and the upper half of the block input to the second block is fixed. After Step 1 and Step 2, a collision should occur in the lower half of the block input to the second block. Let i' and j' be the indices for the colliding pair. For this pair, the 256-bit output of the second block cipher call also collides. Given the value of $T_U'^{(j)}\|T_L'^{(j)}$ in Step 2, the simulation for the third block cipher call has no secret value, hence the results of the simulation for j' and the output for i' always collide.

5.3 Concluding Remarks

In this paper we revisited the security of an ISO standard Lesamnta-LW. We first improved the bound of the number of active S-boxes with MILP to show that Lesamnta-LW activates more S-boxes than originally expected and derived the tight bound of the full cipher. We then analyzed the Shuffle operation to show that 2-byte-wise shuffle is better than byte-wise shuffle, and finally evaluated security against linear cryptanalysis.

In the second part, we proposed a new PRF mode based on Lesamnta-LW-BC that doubles the number of processed message bits per block-cipher call. We provided the security proofs both in the standard and the ideal cipher models.

Finally, we discussed the observation of the related-key truncated differentials in the branch-wise truncation and failure examples.

We believe the ISO standard Lesamnta-LW deserves more attention from the community and this research provides deeper understanding of its security.

Acknowledgments

The first author was supported in part by JSPS KAKENHI Grant Number JP18H05289.

References

[1] A. Akhimullah and S. Hirose, "Lightweight hashing using Lesamnta-LW compression function mode and MDP domain extension,"CANDAR 2016, pp.590–596, IEEE Computer Society, 2016.
[2] M. Bellare and T. Kohno, "A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications," E. Biham, ed., EUROCRYPT 2003, volume 2656 of LNCS, pp.491–506. Springer, 2003.
[3] G. Bertoni, J. Daemen, M. Peeters, and G.V. Assche, "Permutation-based encryption, authentication and authenticated encryption," Workshop Records of DIAC 2012, pp.159–170, 2012.
[4] C. Bouillaguet, O. Dunkelman, G. Leurent, and P. Fouque, "Another look at complementation properties,"S. Hong and T. Iwata, eds., FSE 2010, volume 6147 of LNCS, pp.347–364, Springer, 2010.
[5] J. Daemen and V. Rijmen, AES Proposal: Rijndael (Document version 2), Submission to NIST, 1999.
[6] C. Guo, O. Pereira, T. Peters, and F. Standaert, "Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract)," P. Schwabe and N. Thériault, eds., LATINCRYPT 2019, volume 11774 of LNCS, pp.150–172, Springer, 2019.
[7] H. Handschuh and D. Naccache, SHACAL, Modifications to NESSIE submissions selected for 2nd Phase, 2001.
[8] S. Hirose, K. Ideguchi, H. Kuwakado, T. Owada, B. Preneel, and H. Yoshida, "A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-LW," K.H. Rhee and D. Nyang, eds., ICISC 2010, volume 6829 of LNCS, pp.151–168, Springer, 2010.
[9] S. Hirose, K. Ideguchi, H. Kuwakado, T. Owada, B. Preneel, and H. Yoshida, "An AES based 256-bit hash function for lightweight applications: Lesamnta-LW," IEICE Trans. Fundamentals, vol.E95-A, no.1, pp.89–99, Jan. 2012.
[10] S. Hirose and H. Kuwakado, "Efficient pseudorandom-function modes of a block-cipher-based hash function," IEICE Trans. Fundamentals, vol.E92-A, no.10, pp.2447–2453, Oct. 2009.
[11] S. Hirose, H. Kuwakado, and H. Yoshida, "SHA-3 Proposal: Lesamnta," Submission to NIST, 2008.
[12] S. Hirose, H. Kuwakado, and H. Yoshida, "A minor change to Lesamnta — Change of round constants —," Available at webpage, 2010.
[13] S. Hirose, H. Kuwakado, and H. Yoshida, "A pseudorandom-function mode based on Lesamnta-LW and the MDP domain extension and its applications," IEICE Trans. Fundamentals, vol.E101-A,

no.1, pp.110–118, Jan. 2018.
[14] S. Hirose, Y. Sasaki, and H. Yoshida, "Lesamnta-LW revisited: Improved security analysis of primitive and new PRF mode," M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, eds., ACNS 2020, volume 12146 of LNCS, pp.89–109. Springer, 2020.
[15] G.O. Inc. Gurobi optimizer 7.0. Official webpage, http://www.gurobi.com/, 2016.
[16] ISO/IEC JTC 1. ISO/IEC 18033-4-5:2005 Information technology – Security techniques – Encryption algorithms – Part 4: Stream ciphers, first edition, July 2005.
[17] ISO/IEC JTC 1. ISO/IEC 29192-5:2016 Information technology – Security techniques – Lightweight cryptography – Part 5: Hash-functions, first edition, Aug. 2016.
[18] ISO/IEC JTC 1. ISO/IEC 29192-6:2019 Information technology – Security techniques – Lightweight cryptography – Part 6: Message Authentication Codes, first edition edition, Sept. 2019.
[19] K. Kondo, Y. Sasaki, and T. Iwata, "On the design rationale of Simon block cipher: Integral attacks and impossible differential attacks against Simon variants," M. Manulis, A. Sadeghi, and S. Schneider, eds., ACNS 2016, volume 9696 of LNCS, pp.518–536, Springer, 2016.
[20] M. Matsui, "Linear cryptanalysis method for DES cipher," T. Helleseth, ed., EUROCRYPT'93, volume 765 of LNCS, pp.386–397, Springer, 1993.
[21] M. Matsui, "On correlation between the order of S-boxes and the strength of DES," A.D. Santis, ed., EUROCRYPT'94, volume 950 of LNCS, pp.366–375, Springer, 1994.
[22] N. Mouha, Q. Wang, D. Gu, and B. Preneel, "Differential and linear cryptanalysis using mixed-integer linear programming," C. Wu, M. Yung, and D. Lin, eds., Inscrypt 2011, volume 7537 of LNCS, pp.57–76, Springer, 2011.
[23] National Institute of Standards and Technology, FIPS 197: Advanced Encryption Standard (AES), Nov. 2001.
[24] National Institute of Standards and Technology, FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Aug. 2015.
[25] O. Pereira, F. Standaert, and S. Vivek, "Leakage-resilient authentication and encryption from symmetric cryptographic primitives," Proc. 22nd ACM SIGSAC, pp.96–108, 2015.
[26] N. Pramstaller, C. Rechberger, and V. Rijmen, "Impact of rotations in SHA-1 and related hash functions," B. Preneel and S.E. Tavares, eds., SAC 2005, volume 3897 of LNCS, pp.261–275, Springer, 2005.
[27] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, and B. Preneel, "A new keystream generator MUGI," J. Daemen and V. Rijmen, eds., FSE 2002, volume 2365 of LNCS, pp.179–194, Springer, 2002.
[28] K. Yasuda, "Boosting Merkle-Damgård hashing for message authentication," K. Kurosawa, ed., ASIACRYPT 2007, volume 4833 of LNCS, pp.216–231, Springer, 2007.
[29] Y. Zhang, S. Sun, J. Cai, and L. Hu, "Speeding up MILP aided differential characteristic search with Matsui's strategy," L. Chen, M. Manulis, and S. Schneider, eds., ISC 2018, volume 11060 of LNCS, pp.101–115, Springer, 2018.

Appendix A: Lesamnta-LW-BC

The mixing function updates its state in the i-th round as follows:

$$X_0^{(i+1)} = X_3^{(i)} \oplus G(X_2^{(i)}, k^{(i)}), \quad X_1^{(i+1)} = X_0^{(i)},$$
$$X_2^{(i+1)} = X_1^{(i)}, \quad X_3^{(i+1)} = X_2^{(i)}.$$

The updating function G is defined as follows:

$$G(Y, k^{(i)}) := \mathrm{Shuffle}(Q(Y_0 \oplus k^{(i)}), Q(Y_1)),$$

where $Y = (Y_0, Y_1) \in \{0,1\}^{32\times 2}$. For a 4-byte input $(s_0, s_1, s_2, s_3) \in \{0,1\}^{8\times 4}$, Q applies the AES S-box to each byte and the AES MixColumns in this order. Shuffle is defined as follows: For a 8-byte input $s = (s_0, s_1, \ldots, s_7) \in \{0,1\}^{8\times 8}$,

$$\mathrm{Shuffle}(s) := (s_4, s_5, s_2, s_3, s_0, s_1, s_6, s_7).$$

In the i-th round, the key schedule function outputs the round key $k^{(i)} \leftarrow K_0^{(i)}$ and updates its state as follows:

$$K_0^{(i+1)} = K_3^{(i)} \oplus Q(K_2^{(i)} \oplus C^{(i)}), \quad K_1^{(i+1)} = K_0^{(i)},$$
$$K_2^{(i+1)} = K_1^{(i)}, \quad K_3^{(i+1)} = K_2^{(i)}.$$

We ask readers to refer to the specification in [9] for the details of the round-constants $C^{(i)}$.

Appendix B: Proof of Theorem 2

It is assumed without loss of generality that the adversary **A** does not make trivial queries.

For the game Gr1 described in Fig. A· 1, the oracle $\mathcal{R}$ implements LRF^E and the oracles $\mathcal{E}$ and $\mathcal{D}$ implement the encryption and the decryption of E, respectively.

The transformation from Gr1 to Gr2 only changes the functions E and D, which are given in Fig. A· 2. In Gr2, both E and D choose each reply uniformly at random. Gr2 is equivalent to Gr1 until *bad* gets `true` in Gr2. Thus,

$$|\Pr[\mathbf{A}^{\mathrm{Gr1}} = 1] - \Pr[\mathbf{A}^{\mathrm{Gr2}} = 1]|$$

Initialization:
100: $K \xleftarrow{\$} \{0,1\}^n$
101: $\mathsf{E}[W,X] \leftarrow \perp$ for every (W,X)
102: $\mathsf{D}[W,Y] \leftarrow \perp$ for every (W,Y)
103: $P_W \leftarrow \{\}$; $C_W \leftarrow \{\}$

Oracle $\mathcal{R}(M)$:
200: $M[1]\|M[2]\|\cdots\|M[m] \leftarrow M$
201: $V \leftarrow K\|0^n$ ▷ $V = V_0\|V_1, |V_0| = |V_1| = n$
202: **for** $1 \le i \le m-1$ **do**
203: $\quad V \leftarrow \mathsf{E}(V_0, M_0[i]\|(M_1[i] \oplus V_1))$
204: **end for**
205: $V \leftarrow \mathsf{E}(V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1))$
206: **return** V

Oracle $\mathcal{E}(W,X)$:
300: **return** $\mathsf{E}(W,X)$

Oracle $\mathcal{D}(W,Y)$:
400: **return** $\mathsf{D}(W,Y)$

Function $\mathsf{E}(W,X)$:
500: **if** $\mathsf{E}[W,X] = \perp$ **then**
501: $\quad Y \xleftarrow{\$} \{0,1\}^{2n} \setminus C_W$
502: $\quad P_W \leftarrow P_W \cup \{X\}$
503: $\quad C_W \leftarrow C_W \cup \{Y\}$
504: $\quad \mathsf{E}[W,X] \leftarrow Y$
505: $\quad \mathsf{D}[W,Y] \leftarrow X$
506: **end if**
507: **return** $\mathsf{E}[W,X]$

Function $\mathsf{D}(W,Y)$:
600: **if** $\mathsf{D}[W,Y] = \perp$ **then**
601: $\quad X \xleftarrow{\$} \{0,1\}^{2n} \setminus P_W$
602: $\quad P_W \leftarrow P_W \cup \{X\}$
603: $\quad C_W \leftarrow C_W \cup \{Y\}$
604: $\quad \mathsf{E}[W,X] \leftarrow Y$
605: $\quad \mathsf{D}[W,Y] \leftarrow X$
606: **end if**
607: **return** $\mathsf{D}[W,Y]$

Fig. A· 1 Game Gr1.

Function $\mathsf{E}(W, X)$:	**Function** $\mathsf{D}(W, Y)$:
500: **if** $\mathtt{E}[W, X] = \perp$ **then**	600: **if** $\mathtt{D}[W, Y] = \perp$ **then**
501: $Y \twoheadleftarrow \{0,1\}^{2n}$	601: $X \twoheadleftarrow \{0,1\}^{2n}$
502: **if** $Y \in C_W$ **then**	602: **if** $X \in P_W$ **then**
503: $bad \leftarrow \mathtt{true}$	603: $bad \leftarrow \mathtt{true}$
504: **end if**	604: **end if**
505: $P_W \leftarrow P_W \cup \{X\}$	605: $P_W \leftarrow P_W \cup \{X\}$
506: $C_W \leftarrow C_W \cup \{Y\}$	606: $C_W \leftarrow C_W \cup \{Y\}$
507: $\mathtt{E}[W, X] \leftarrow Y$	607: $\mathtt{E}[W, X] \leftarrow Y$
508: $\mathtt{D}[W, Y] \leftarrow X$	608: $\mathtt{D}[W, Y] \leftarrow X$
509: **end if**	609: **end if**
510: **return** $\mathtt{E}[W, X]$	610: **return** $\mathtt{D}[W, Y]$

Fig. A· 2 Functions E and D of game Gr2.

Function $\mathsf{E}(W, X)$:	**Function** $\mathsf{D}(W, Y)$:
500: **if** $\mathtt{E}[W, X] = \perp$ **then**	600: **if** $\mathtt{D}[W, Y] = \perp$ **then**
501: $Y \twoheadleftarrow \{0,1\}^{2n}$	601: $X \twoheadleftarrow \{0,1\}^{2n}$
502: $\mathtt{E}[W, X] \leftarrow Y$	602: $\mathtt{E}[W, X] \leftarrow Y$
503: $\mathtt{D}[W, Y] \leftarrow X$	603: $\mathtt{D}[W, Y] \leftarrow X$
504: **end if**	604: **end if**
505: **return** $\mathtt{E}[W, X]$	605: **return** $\mathtt{D}[W, Y]$

Fig. A· 3 Functions E and D of game Gr3.

Initialization:	**Function** $\mathsf{E}(W, X)$:
100: $P_W \leftarrow \{\}$; $C_W \leftarrow \{\}$	500: $Y \twoheadleftarrow \{0,1\}^{2n} \setminus C_W$
Oracle $\mathcal{R}(M)$:	501: $P_W \leftarrow P_W \cup \{X\}$
200: $V \twoheadleftarrow \{0,1\}^{2n}$	502: $C_W \leftarrow C_W \cup \{Y\}$
201: **return** V	503: **return** Y
Oracle $\mathcal{E}(W, X)$:	**Function** $\mathsf{D}(W, Y)$:
300: **return** $\mathsf{E}(W, X)$	600: $X \twoheadleftarrow \{0,1\}^{2n} \setminus P_W$
Oracle $\mathcal{D}(W, Y)$:	601: $P_W \leftarrow P_W \cup \{X\}$
400: **return** $\mathsf{D}(W, Y)$	602: $C_W \leftarrow C_W \cup \{Y\}$
	603: **return** X

Fig. A· 4 Game Gi1.

$$\leq \Pr[bad \text{ gets } \mathtt{true} \text{ in Gr2}] \leq \frac{(\sigma + q_{\mathrm{e}} + q_{\mathrm{d}})^2}{2^{2n+1}}. \quad (\mathrm{A}\cdot 1)$$

The transformation from Gr2 to Gr3 also only changes the functions E and D, which are given in Fig. A· 3. The changes are minor and

$$\Pr[\mathbf{A}^{\mathrm{Gr2}} = 1] = \Pr[\mathbf{A}^{\mathrm{Gr3}} = 1]. \quad (\mathrm{A}\cdot 2)$$

For the game Gi1 in Fig. A· 4, the oracle $\mathcal{R}$ implements a function from $(\{0,1\}^{2n})^+$ to $\{0,1\}^{2n}$ chosen uniformly at random and the oracles $\mathcal{E}$ and $\mathcal{D}$ implement the encryption and the decryption of E, respectively.

The transformation from Gi1 to Gi2 in Fig. A· 5 only changes the functions E and D. It is similar to the transformation from Gr1 to Gr2, and

$$|\Pr[\mathbf{A}^{\mathrm{Gi1}} = 1] - \Pr[\mathbf{A}^{\mathrm{Gi2}} = 1]| \leq \Pr[bad \text{ gets } \mathtt{true} \text{ in Gi2}] \leq \frac{(q_{\mathrm{e}} + q_{\mathrm{d}})^2}{2^{2n+1}}. \quad (\mathrm{A}\cdot 3)$$

Notice that $\mathcal{R}$ makes no call to E in Gi2.

The transformation from Gi2 to Gi3 in Fig. A· 6 only changes the functions E and D, which makes **Initialization** unnecessary. The changes are minor and

Initialization:	**Function** $\mathsf{E}(W, X)$:
100: $P_W \leftarrow \{\}$; $C_W \leftarrow \{\}$	500: $Y \twoheadleftarrow \{0,1\}^{2n}$
Oracle $\mathcal{R}(M)$:	501: **if** $Y \in C_W$ **then**
200: $V \twoheadleftarrow \{0,1\}^{2n}$	502: $bad \leftarrow \mathtt{true}$
201: **return** V	503: **end if**
Oracle $\mathcal{E}(W, X)$:	504: $P_W \leftarrow P_W \cup \{X\}$
300: **return** $\mathsf{E}(W, X)$	505: $C_W \leftarrow C_W \cup \{Y\}$
Oracle $\mathcal{D}(W, Y)$:	506: **return** Y
400: **return** $\mathsf{D}(W, Y)$	**Function** $\mathsf{D}(W, Y)$:
	600: $X \twoheadleftarrow \{0,1\}^{2n}$
	601: **if** $X \in P_W$ **then**
	602: $bad \leftarrow \mathtt{true}$
	603: **end if**
	604: $P_W \leftarrow P_W \cup \{X\}$
	605: $C_W \leftarrow C_W \cup \{Y\}$
	606: **return** X

Fig. A· 5 Game Gi2.

Oracle $\mathcal{R}(M)$:	**Function** $\mathsf{E}(W, X)$:
200: $V \twoheadleftarrow \{0,1\}^{2n}$	500: $Y \twoheadleftarrow \{0,1\}^{2n}$
201: **return** V	501: **return** Y
Oracle $\mathcal{E}(W, X)$:	**Function** $\mathsf{D}(W, Y)$:
300: **return** $\mathsf{E}(W, X)$	600: $X \twoheadleftarrow \{0,1\}^{2n}$
Oracle $\mathcal{D}(W, Y)$:	601: **return** X
400: **return** $\mathsf{D}(W, Y)$	

Fig. A· 6 Game Gi3.

$$\Pr[\mathbf{A}^{\mathrm{Gi2}} = 1] = \Pr[\mathbf{A}^{\mathrm{Gi3}} = 1]. \quad (\mathrm{A}\cdot 4)$$

The transformation from Gi3 to Gi4 in Fig. A· 7 only changes $\mathcal{R}$. $\mathcal{R}$ in Gi4 implements $\mathrm{LRF}^{\tilde{E}}$, where $\tilde{E}$ is independent from E (E and D). It implements $\tilde{E}$ using $\tilde{\mathsf{E}}$. $\mathcal{R}$ in Gi4 is equivalent to $\mathcal{R}$ in Gi3 if the output for each query is chosen uniformly at random. $\mathsf{new}(M[1, i])$ is true if and only is there exists no previous query M' such that $|M'|/(2n) \geq i + 1$ and $M'[1, i] = M[1, i]$. A collision among inputs to $\tilde{E}$ causes $bad \leftarrow \mathtt{true}$, and Gi4 is equivalent to Gi3 until bad gets $\mathtt{true}$ in Gi4. Thus,

$$|\Pr[\mathbf{A}^{\mathrm{Gi3}} = 1] - \Pr[\mathbf{A}^{\mathrm{Gi4}} = 1]| \leq \Pr[bad \text{ gets } \mathtt{true} \text{ in Gi4}] \leq \frac{\sigma^2}{2^{2n+1}} + \frac{\sigma}{2^n}. \quad (\mathrm{A}\cdot 5)$$

The transformation from Gi4 to Gi5 only changes $\mathcal{R}$, which is given in Fig. A· 8. The changes are minor and

$$\Pr[\mathbf{A}^{\mathrm{Gi4}} = 1] = \Pr[\mathbf{A}^{\mathrm{Gi5}} = 1]. \quad (\mathrm{A}\cdot 6)$$

The transformation from Gi5 to Gi6 in Fig. A· 9 changes $\mathcal{R}$, E and D. $\mathcal{R}$ in Gi6 implements LRF^{E} using the function E. Gi6 is equivalent to Gi5 if E and D do not receive repeated queries. Namely, Gi6 is equivalent to Gi5 until bad gets $\mathtt{true}$ in Gi6. The probability that E receives a repeated query is at most $q_{\mathrm{e}}/2^n + \sigma q_{\mathrm{e}}/2^{2n}$. The probability that D receives a repeated query is at most $q_{\mathrm{d}}/2^n + \sigma q/2^{2n} + \sigma q_{\mathrm{d}}/2^{2n}$. $q_{\mathrm{d}}/2^n + \sigma q/2^{2n}$ is an upper bound on the probability that

Initialization:
100: $K \twoheadleftarrow \{0,1\}^n$
101: $\tilde{\mathsf{E}}[W,X] \leftarrow \perp$ for every (W,X)

Oracle $\mathcal{R}(M)$:
200: $M[1]\|M[2]\|\cdots\|M[m] \leftarrow M$
201: $V \leftarrow K\|0^n$
202: **for** $1 \le i \le m-1$ **do**
203: **if** new$(M[1,i])$ **then**
204: **if** $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] = \perp$ **then**
205: $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] \twoheadleftarrow \{0,1\}^{2n}$
206: **else**
207: $bad \leftarrow$ `true`
208: **end if**
209: **end if**
210: $V \leftarrow \tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)]$
211: **end for**
212: **if** $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] = \perp$ **then**
213: $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] \twoheadleftarrow \{0,1\}^{2n}$
214: **else**
215: $bad \leftarrow$ `true`
216: **end if**
217: $V \leftarrow \tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)]$
218: **return** V

Oracle $\mathcal{E}(W,X)$:
300: **return** $\mathsf{E}(W,X)$

Oracle $\mathcal{D}(W,Y)$:
400: **return** $\mathsf{D}(W,Y)$

Function $\mathsf{E}(W,X)$:
500: $Y \twoheadleftarrow \{0,1\}^{2n}$
501: **return** Y

Function $\mathsf{D}(W,Y)$:
600: $X \twoheadleftarrow \{0,1\}^{2n}$
601: **return** X

Fig. A· 7 Game Gi4.

Oracle $\mathcal{R}(M)$:
200: $M[1]\|M[2]\|\cdots\|M[m] \leftarrow M$
201: $V \leftarrow K\|0^n$
202: **for** $1 \le i \le m-1$ **do**
203: **if** $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] = \perp$ **then**
204: $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] \twoheadleftarrow \{0,1\}^{2n}$
205: **end if**
206: $V \leftarrow \tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)]$
207: **end for**
208: **if** $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] = \perp$ **then**
209: $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] \twoheadleftarrow \{0,1\}^{2n}$
210: **end if**
211: $V \leftarrow \tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)]$
212: **return** V

Fig. A· 8 The oracle $\mathcal{R}$ of game Gi5.

there exists a query made by $\mathbf{A}$ that collides with a query made by $\mathcal{R}$ for a returned value of $\mathcal{R}$, where $\sigma q/2^{2n}$ is an upper bound on the probability that there exists a returned value of $\mathcal{R}$ that collides with a returned value of E. Thus,

$$\begin{aligned} &|\Pr[\mathbf{A}^{\mathrm{Gi5}} = 1] - \Pr[\mathbf{A}^{\mathrm{Gi6}} = 1]| \\ &\quad \le \Pr[bad \text{ gets } \texttt{true} \text{ in Gi6}] \\ &\quad \le \frac{q_{\mathrm{e}} + q_{\mathrm{d}}}{2^n} + \frac{\sigma(q + q_{\mathrm{e}} + q_{\mathrm{d}})}{2^{2n}}. \end{aligned} \tag{A· 7}$$

Initialization:
100: $K \twoheadleftarrow \{0,1\}^n$
101: $\tilde{\mathsf{E}}[W,X] \leftarrow \perp$ for every (W,X)
102: $\mathtt{E}[W,X] \leftarrow \perp$ for every (W,X)
103: $\mathtt{D}[W,Y] \leftarrow \perp$ for every (W,Y)

Oracle $\mathcal{R}(M)$:
200: $M[1]\|M[2]\|\cdots\|M[m] \leftarrow M$
201: $V \leftarrow K\|0^n$
202: **for** $1 \le i \le m-1$ **do**
203: **if** $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] = \perp$ **then**
204: $Z \leftarrow \mathsf{E}(V_0, M_0[i]\|(M_1[i] \oplus V_1))$
205: $\tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)] \leftarrow Z$
206: **end if**
207: $V \leftarrow \tilde{\mathsf{E}}[V_0, M_0[i]\|(M_1[i] \oplus V_1)]$
208: **end for**
209: **if** $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] = \perp$ **then**
210: $Z \leftarrow \mathsf{E}(V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1))$
211: $\tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)] \leftarrow Z$
212: **end if**
213: $V \leftarrow \tilde{\mathsf{E}}[V_0 \oplus c, M_0[m]\|(M_1[m] \oplus V_1)]$
214: **return** V

Oracle $\mathcal{E}(W,X)$:
300: **return** $\mathsf{E}(W,X)$

Oracle $\mathcal{D}(W,Y)$:
400: **return** $\mathsf{D}(W,Y)$

Function $\mathsf{E}(W,X)$:
500: **if** $\mathtt{E}[W,X] = \perp$ **then**
501: $Y \twoheadleftarrow \{0,1\}^{2n}$
502: $\mathtt{E}[W,X] \leftarrow Y$
503: $\mathtt{D}[W,Y] \leftarrow X$
504: **else**
505: $bad \leftarrow$ `true`
506: **end if**
507: **return** $\mathtt{E}[W,X]$

Function $\mathsf{D}(W,Y)$:
600: **if** $\mathtt{D}[W,Y] = \perp$ **then**
601: $X \twoheadleftarrow \{0,1\}^{2n}$
602: $\mathtt{E}[W,X] \leftarrow Y$
603: $\mathtt{D}[W,Y] \leftarrow X$
604: **else**
605: $bad \leftarrow$ `true`
606: **end if**
607: **return** $\mathtt{D}[W,Y]$

Fig. A· 9 Game Gi6.

The differences between Gi6 and Gr3 are minor and

$$\Pr[\mathbf{A}^{\mathrm{Gi6}} = 1] = \Pr[\mathbf{A}^{\mathrm{Gr3}} = 1]. \tag{A· 8}$$

From (A· 1) to (A· 8),

$$\mathrm{Adv}^{\mathrm{ind}}_{\mathrm{LRF}^E}(\mathbf{A}) \le \frac{(\sigma + q_{\mathrm{e}} + q_{\mathrm{d}})^2}{2^{2n}} + \frac{\sigma q}{2^{2n}} + \frac{q_{\mathrm{e}} + q_{\mathrm{d}}}{2^n}.$$

This completes the proof.

Appendix C: Complete MILP Model for 3 Rounds of Lesamnta-LW-BC

```
Minimize
x8 +  x9 + ... + x30 + x31

Subject To
x8 + x9 + x10 + x11 + y0
+ y1 + y2 + y3 -5 d0 >= 0
d0 - x8 >= 0
d0 - x9 >= 0
d0 - x10 >= 0
d0 - x11 >= 0
d0 - y0 >= 0
d0 - y1 >= 0
d0 - y2 >= 0
d0 - y3 >= 0
x12 + x13 + x14 + x15 + y4
+ y5 + y6 + y7 -5 d1 >= 0
d1 - x12 >= 0
d1 - x13 >= 0
d1 - x14 >= 0
d1 - x15 >= 0
d1 - y4 >= 0
d1 - y5 >= 0
d1 - y6 >= 0
d1 - y7 >= 0
  y4 + x0 - x32 >= 0
  y4 - x0 + x32 >= 0
- y4 + x0 + x32 >= 0
  y5 + x1 - x33 >= 0
  y5 - x1 + x33 >= 0
- y5 + x1 + x33 >= 0
  y2 + x2 - x34 >= 0
  y2 - x2 + x34 >= 0
- y2 + x2 + x34 >= 0
  y3 + x3 - x35 >= 0
  y3 - x3 + x35 >= 0
- y3 + x3 + x35 >= 0
  y0 + x4 - x36 >= 0
  y0 - x4 + x36 >= 0
- y0 + x4 + x36 >= 0
  y1 + x5 - x37 >= 0
  y1 - x5 + x37 >= 0
- y1 + x5 + x37 >= 0
  y6 + x6 - x38 >= 0
  y6 - x6 + x38 >= 0
- y6 + x6 + x38 >= 0
  y7 + x7 - x39 >= 0
  y7 - x7 + x39 >= 0
- y7 + x7 + x39 >= 0
x16 + x17 + x18 + x19 + y8
+ y9 + y10 + y11 -5 d2 >= 0
d2 - x16 >= 0
d2 - x17 >= 0
d2 - x18 >= 0
d2 - x19 >= 0
d2 - y8 >= 0
d2 - y9 >= 0
d2 - y10 >= 0
d2 - y11 >= 0
x20 + x21 + x22 + x23 + y12
+ y13 + y14 + y15 -5 d3 >= 0
d3 - x20 >= 0
d3 - x21 >= 0
d3 - x22 >= 0
d3 - x23 >= 0
d3 - y12 >= 0
d3 - y13 >= 0
d3 - y14 >= 0
d3 - y15 >= 0
y12 + x8 - x40 >= 0
y12 - x8 + x40 >= 0
- y12 + x8 + x40 >= 0
  y13 + x9 - x41 >= 0
  y13 - x9 + x41 >= 0
- y13 + x9 + x41 >= 0
  y10 + x10 - x42 >= 0
  y10 - x10 + x42 >= 0
- y10 + x10 + x42 >= 0
  y11 + x11 - x43 >= 0
  y11 - x11 + x43 >= 0
- y11 + x11 + x43 >= 0
  y8 + x12 - x44 >= 0
  y8 - x12 + x44 >= 0
- y8 + x12 + x44 >= 0
  y9 + x13 - x45 >= 0
  y9 - x13 + x45 >= 0
- y9 + x13 + x45 >= 0
  y14 + x14 - x46 >= 0
  y14 - x14 + x46 >= 0
- y14 + x14 + x46 >= 0
  y15 + x15 - x47 >= 0
  y15 - x15 + x47 >= 0
- y15 + x15 + x47 >= 0
x24 + x25 + x26 + x27 + y16
+ y17 + y18 + y19 -5 d4 >= 0
d4 - x24 >= 0
d4 - x25 >= 0
d4 - x26 >= 0
d4 - x27 >= 0
d4 - y16 >= 0
d4 - y17 >= 0
d4 - y18 >= 0
d4 - y19 >= 0
x28 + x29 + x30 + x31 + y20
+ y21 + y22 + y23 -5 d5 >= 0
d5 - x28 >= 0
d5 - x29 >= 0
d5 - x30 >= 0
d5 - x31 >= 0
d5 - y20 >= 0
d5 - y21 >= 0
d5 - y22 >= 0
d5 - y23 >= 0
y20 + x16 - x48 >= 0
y20 - x16 + x48 >= 0
- y20 + x16 + x48 >= 0
y21 + x17 - x49 >= 0
y21 - x17 + x49 >= 0
- y21 + x17 + x49 >= 0
y18 + x18 - x50 >= 0
y18 - x18 + x50 >= 0
- y18 + x18 + x50 >= 0
y19 + x19 - x51 >= 0
y19 - x19 + x51 >= 0
- y19 + x19 + x51 >= 0
y16 + x20 - x52 >= 0
y16 - x20 + x52 >= 0
- y16 + x20 + x52 >= 0
y17 + x21 - x53 >= 0
y17 - x21 + x53 >= 0
- y17 + x21 + x53 >= 0
y22 + x22 - x54 >= 0
y22 - x22 + x54 >= 0
- y22 + x22 + x54 >= 0
y23 + x23 - x55 >= 0
y23 - x23 + x55 >= 0
- y23 + x23 + x55 >= 0
x0 + x1 + ... + x30 + x31 >= 1

Binary
x0 x1 ... x54 x55
y0 y1 ... y22 y23
d0 d1 d2 d3 d4 d5
End
```

Fig. A·10 MILP model for 3 rounds of Lesamnta-LW-BC.

Shoichi Hirose received the B.E., M.E. and D.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1988, 1990 and 1995, respectively. From 1990 to 1998, he was a research associate at Faculty of Engineering, Kyoto University. From 1998 to 2005, he was a lecturer at Graduate School of Informatics, Kyoto University. From 2005 to 2009, he was an associate professor at Faculty of Engineering, University of Fukui. From 2009, he is a professor at Graduate School of Engineering, University of Fukui. His current interests include cryptography and information security. He received Young Engineer Award from IEICE in 1997, and KDDI Foundation Research Award in 2008.

Yu Sasaki received Bachelor of Engineering and Master of Engineering from The University of Electro-Communications in 2005 and 2007. In 2020, he received Ph.D. degrees from The University of Electro-Communications, focusing on the symmetric-key cryptography under the supervision of Kazuo Ohta. Since 2007, he has been a researcher at NTT Secure Platform Laboratories. His current research interests are in cryptography, including design and security analysis of symmetric-key cryptographic schemes. He was awarded a paper prize from SCIS 2007, IEICE Trans. in 2009 and IEICE Trans. in 2018. He also received best paper awards from IWSEC 2009, SECRYPT 2012, and IWSEC 2012.

Hirotaka Yoshida received the B.S. degree from Meiji University, Japan, in 1999, the M.S. degree from Tokyo Institute of Technology, Japan, in 2001, and the Ph.D. degree in electrical engineering from KU Leuven, Belgium, in 2013. From 2001 to 2016, he was with the Research & Development Group, Hitachi, Ltd. He is currently a team leader at the National Institute of Advanced Industrial Science and Technology (AIST). He is a member of IACR, IPSJ, and JSAE. In 2013, he won the award of industrial standardization that has been granted by the Japanese Ministry of Economy, Trade and Industry (METI).