On MILP-Based Automatic Search for Differential Trails Through Modular Additions with Application to Bel-T

Muhammad ElSheikh, Ahmed Abdelkhalek, Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Québec, Canada
youssef@ciise.concordia.ca

Abstract. Using modular addition as a source of nonlinearity is frequently used in many symmetric-key structures such as ARX and Lai–Massey schemes. At FSE'16, Fu *et al.* proposed a Mixed Integer Linear Programming (MILP)-based method to handle the propagation of differential trails through modular additions assuming that the two inputs to the modular addition and the consecutive rounds are independent. However, this assumption does not necessarily hold. In this paper, we study the propagation of the XOR difference through the modular addition at the bit level and show the effect of the carry bit. Then, we propose a more accurate MILP model to describe the differential propagation through the modular addition taking into account the dependency between the consecutive modular additions. The proposed MILP model is utilized to launch a differential attack against Bel-T-256, which is a member of the Bel-T block cipher family that has been adopted recently as a national standard of the Republic of Belarus. In particular, we employ the concept of partial Differential Distribution Table to model the 8-bit S-Box of Bel-T using a MILP approach in order to automate finding a differential characteristic of the cipher. Then, we present a $4\frac{1}{7}$-round (out of 8) differential attack which utilizes a 3-round differential characteristic that holds with probability 2^{-111}. The data, time and memory complexities of the attack are 2^{114} chosen plaintexts, $2^{237.14}$ $4\frac{1}{7}$-round encryptions, and 2^{224} 128-bit blocks, respectively.

Keywords: Differential cryptanalysis · MILP · Modular Addition · ARX · Bel-T

1 Introduction

Differential cryptanalysis, which was introduced by Biham and Shamir [4], is one of the most powerful attacks that are used to evaluate the security of symmetric-key primitives. For an n-bit primitive, the crucial step of the differential attack is to find a distinguisher ($\Delta P \rightarrow \Delta C$) where an XOR difference of two plaintexts (ΔP) gives, after some rounds, another XOR difference (ΔC) with probability higher than 2^{-n}, independent of the secret key. Using this distinguisher, a key recovery attack can be performed by appending (prepending) some rounds after (before) the distinguisher and guessing the round keys.

Different optimization techniques such as Mixed Integer Linear Programming (MILP) attracted the attention of many cryptanalysis researchers. The first attempt to utilize MILP technique in symmetric-key cryptanalysis was developed by Mouha *et al.* [17] in which they applied a MILP technique to prove security bounds against both differential and linear cryptanalysis. Later, Cui *et al.* [6] proposed a MILP model for both impossible differential and zero-correlation attacks. Sasaki and Todo [19] developed a new search tool for impossible differential using MILP. Recently, Xiang *et al.* [25] defined systematic rules for constructing integral distinguishers using MILP. Then, Sun *et al.* complemented this work by handling ARX-based ciphers (`modulo` operations) [21] and ciphers with non-bit-permutation linear layer [22]. One of the downsides of these MILP models was the inability to efficiently describe the Difference Distribution Table (DDT) of large (8-bit) S-boxes which was tackled by Abdelkhalek *et al.* [2]. Regarding ARX-based block ciphers, Fu *et al.* [10] represented the conditions developed by Lipmaa and Moriai [15] (hereafter referred to as Lipmaa's conditions) by a set of MILP constraints in order to automate the search for the best differential trail through the modular addition. In this representation, the authors assume that the two inputs to modular addition and the consecutive component of the cipher's round function are independent. However, this assumption is very often not satisfied, especially with round functions that have two or more consecutive modular operations, see [24]. In the same context, Leurent [14] provides a tool based on finite state machines to automate the search for differential characteristics through the modular addition considering the constraints due to several consecutive bits of the modular addition inputs. However, the complexity of this analysis is linear in the number of states, and the number of states can be exponential in the size of the system, which according to the authors, makes this approach suitable only to study systems with a limited number of states.

In this work, we revisit the conditions stated by Lipmaa and Moriai [15] to verify the possibility of an XOR difference of two inputs of addition modulo 2^n to produce a specific XOR difference at the output. In particular, we deduce the conditions on the bits of the inputs and the output of addition modulo 2^n that have to be satisfied in order to propagate an XOR difference of the inputs to a particular XOR difference at the output. Using these conditions, we describe some examples showing that using Lipmaa's conditions with the independence assumption between the consecutive components of a block cipher is not enough to ensure the validity of the derived differential characteristic. To address this problem, we propose a new MILP model considering the dependency between two or more successive modular additions.

To illustrate the effectiveness of our approach, we apply our method to attack the block cipher Bel-T, which is a family of block ciphers that has been approved as the national standard of the Republic of Belarus [1], formerly known by its Russian name Belorussia. The Bel-T family includes three block ciphers, denoted as Bel-T-k, all of them have the same block size of 128 bits and a variable key length (k) of 128, 192 or 256 bits. The designers of Bel-T combined a Lai-Massey scheme [12] with a Feistel network [9] to build a complex round function

with 7 S-box layers per round. The round function is iterated 8 times to construct the whole cipher. Concretely, we employ our MILP approach beside a Hamming weight-based partial DDT to search for a differential distinguisher for Bel-T. Then, we mount a $4\frac{1}{7}$-round differential attack on round-reduced Bel-T-256 which, up to our knowledge, is the best published attack against this cipher in the single-key setting. Moreover, we show that the Bel-T block cipher is not a Markov cipher [13] *i.e.,* the validity of the differential characteristic depends on the used secret key. In this context, we also provide a systematic method to define the set of keys that can be attacked using our differential characteristic.

Few cryptanalysis results on Bel-T block ciphers have been published including fault-based attacks [11] and the related-key differential attack on round-reduced Bel-T-256 [3]. Recently, ElSheikh *et al.* [8] presented two integral attacks on ($3\frac{2}{7}$ and $3\frac{6}{7}$)-round reduced Bel-T-256 in the single-key setting. It should be noted that in the related-key differential attack presented in [3] , the modular addition is modeled using the method proposed by Fu *et al.* [10] with the independency assumption. We verified the distingusiher presented in [3] and found it to be invalid as it involves two modular additions that share the same input and have conflicting condition. Table 1 contrasts our attack with the integral attacks in [8].

The rest of this paper is organized as follows. In Section 2, we briefly revisit the XOR differential characteristic of modular addition. The developed MILP-based method, which is used to search for the differential characteristic, is explained in Section 3. In Section 4, we describe how we apply the new MILP model to find a differential distinguisher for Bel-T. Then, the details of our attack are presented in Section 5. Finally, the paper is concluded in Section 6.

Table 1: Attack results on Bel-T-256

Model	Attack	#Rounds	Data	Time	Memory	Reference
Single Key	Integral	$3\frac{2}{7}$	2^{13}	$2^{199.33}$	-	[8]
		$3\frac{6}{7}$	2^{33}	$2^{254.61}$	-	[8]
	Differential	$4\frac{1}{7}$	2^{114}	$2^{237.14}$	2^{224}	Sec. 5

2 XOR-Differential Characteristics of Modular Addition

Definition 1. *Let α, β and γ be fixed n-bit XOR differences. The XOR-differential probability (DP) of addition modulo 2^n (xdp^+) is the probability with which α and β propagate to γ through the modular addition operation, computed over all pairs of n-bit inputs (x,y):*

$$xdp^+(\alpha, \beta \rightarrow \gamma) = 2^{-2n} \times \#\{(x, y) : ((x \oplus \alpha) \boxplus (y \oplus \beta)) \oplus (x \boxplus y) = \gamma\}.$$

Lipmaa and Moriai [15] stated the following two conditions that have to be satisfied in order for the XOR input differences (α, β) to propagate to an output difference (γ) through the addition modulo 2^n:

1. The bit-wise XOR of the least significant bit of the inputs and output differences must be 0, *i.e.,* $\alpha_0 \oplus \beta_0 \oplus \gamma_0 = 0$ which is equivalent to $\gamma_0 = \alpha_0 \oplus \beta_0$.
2. If the three bits α_i, β_i, and γ_i are equal, then the XOR of the subsequent bits $\alpha_{i+1}, \beta_{i+1}$, and γ_{i+1} must equal these bits as well, *i.e.,* $\alpha_{i+1} \oplus \beta_{i+1} \oplus \gamma_{i+1} = \alpha_i = \beta_i = \gamma_i$ for $0 \leq i \leq n-2$.

If these two conditions above are satisfied, then the probability of the differential characteristic (xdp^+) can be calculated as:

$$xdp^+(\alpha, \beta \rightarrow \gamma) = 2^{-\sum_{i=0}^{n-2} \neg eq(\alpha_i, \beta_i, \gamma_i)}$$

where $\neg eq$ is 0 when $(\alpha_i, \beta_i, \gamma_i)$ are the same, and 1 otherwise. By using these conditions, we can determine if a differential characteristic $(\alpha, \beta \rightarrow \gamma)$ is a valid one or not. For example, the characteristic $(\alpha, \beta \rightarrow \gamma) = (0001, 0001 \rightarrow 0001)$ is impossible because it breaks the first condition.

In the remaining of this section, we show our interpretation of these two conditions by deriving the relationship between the input and output differences at the bit level.

Let $\boldsymbol{x} = (x_{n-1}, x_{n-2}, \ldots, x_1, x_0)$[1], $\boldsymbol{y} = (y_{n-1}, y_{n-2}, \ldots, y_1, y_0)$, and $\boldsymbol{z} = (z_{n-1}, z_{n-2}, \ldots, z_1, z_0)$ be n-bit vectors where $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$. Then, z_i can be iteratively expressed as follows:

$$z_0 = x_0 \oplus y_0 \oplus c_0, \qquad c_0 = 0, \tag{1}$$

$$z_{i+1} = x_{i+1} \oplus y_{i+1} \oplus c_{i+1}, \quad c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i \quad \forall i = 0, 1, \ldots, n-2. \tag{2}$$

It is obvious that the Lipmaa's conditions are based on equations (1) and (2). Consider that we have two pairs $(\boldsymbol{x}, \boldsymbol{x^*})$ and $(\boldsymbol{y}, \boldsymbol{y^*})$ such that $\boldsymbol{\Delta x} = \boldsymbol{x} \oplus \boldsymbol{x^*}$, and $\boldsymbol{\Delta y} = \boldsymbol{y} \oplus \boldsymbol{y^*}$. The relation between the XOR input differences $\boldsymbol{\Delta x}, \boldsymbol{\Delta y}$ and the XOR output difference $\boldsymbol{\Delta z} = \boldsymbol{z} \oplus \boldsymbol{z^*}$ can be derived as follows: Let $\boldsymbol{\Delta x} = (\delta x_{n-1}, \delta x_{n-2}, \ldots, \delta x_1, \delta x_0)$, $\boldsymbol{\Delta y} = (\delta y_{n-1}, \delta y_{n-2}, \ldots, \delta y_1, \delta y_0)$, and $\boldsymbol{\Delta z} = (\delta z_{n-1}, \delta z_{n-2}, \ldots, \delta z_1, \delta z_0)$ be the XOR difference where $\delta x_i = x_i \oplus x_i^*$, $\delta y_i = y_i \oplus y_i^*$, and $\delta z_i = z_i \oplus z_i^*$, respectively. The Lipmaa's first condition comes from equation (1) in which $\delta z_0 = \delta x_0 \oplus \delta y_0 \oplus \delta c_0$, but $\delta c_0 = 0$ as $c_0 = c_0^* = 0$. Therefore, for $(\boldsymbol{\Delta x}, \boldsymbol{\Delta y} \rightarrow \boldsymbol{\Delta z})$ to be a possible differential characteristic, the relation $(\delta z_0 = \delta x_0 \oplus \delta y_0)$ must be satisfied.

For given input and output differences at two successive bits $((\delta x_i, \delta y_i, \delta z_i)$ and $(\delta x_{i+1}, \delta y_{i+1}, \delta z_{i+1}))$, we can use equation (2) to calculate the XOR difference at the carry bit δc_{i+1} using the following two equations:

$$\begin{aligned} \delta c_{i+1} &= c_{i+1} \oplus c_{i+1}^* \\ &= x_i y_i \oplus x_i c_i \oplus y_i c_i \oplus x_i^* y_i^* \oplus x_i^* c_i^* \oplus y_i^* c_i^*, \end{aligned} \tag{3}$$

$$\delta c_{i+1} = \delta z_{i+1} \oplus \delta x_{i+1} \oplus \delta y_{i+1} \tag{4}$$

[1] We use little-endian representation where x_0 is the least significant bit.

To have a valid differential characteristic, the value of δc_{i+1} evaluated from these two equations must be consistent. For example, if we have $\delta x_i = \delta y_i = \delta z_i = 0$, this implies that $\delta c_i = 0$, *i.e.,* if $x_i^* = x_i$, $y_i^* = y_i$, $z_i^* = z_i$ then $c_i^* = c_i$. Therefore, from equation (3), $\delta c_{i+1} = 0$. Consequently, $\delta z_{i+1} \oplus \delta x_{i+1} \oplus \delta y_{i+1} = 0$ must hold with probability 1.

As another example, let us consider the following XOR differences: $\delta x_i = \delta y_i = 0$, and $\delta z_i = 1$, this implies that $\delta c_i = 1$, *i.e.,* if $x_i^* = x_i$, $y_i^* = y_i$ and $z_i^* = \bar{z}_i$ then $c_i^* = \bar{c}_i$ where $\bar{z}_i$, $\bar{c}_i$ are the bit-wise NOT of z_i, c_i, respectively. As a result, the value of δc_{i+1} from equation (3) will depend on the relation between x_i and y_i as follows: $\delta c_{i+1} = x_i \oplus y_i$. If δc_{i+1} is 0, then the condition $x_i = y_i$ must be satisfied. In this case, from equation (2), the output bit z_i will equal to c_i and the carry bit c_{i+1} will be equal to x_i.

By iterating over all possible values of $\delta x_i, \delta y_i, \delta z_i$ and δc_{i+1}, we can drive the conditions on the bits x_i, y_i, z_i, c_i and c_{i+1} to have a valid differential characteristic. We summarize these conditions in Table 2, in which the condition column is divided into three sub-columns: the first one is the direct condition similar to the one we derived in the previous examples. The second and third sub-columns are the values of z_i and c_{i+1} in case the direct condition, the first sub-column, is satisfied.

It should be noted that Lipmaa's second condition is specified by the first two rows and last two rows of Table 2, *i.e.,* if $\delta x_i, \delta y_i$ and δz_i are equal, then $\delta c_{i+1} = \delta z_{i+1} \oplus \delta x_{i+1} \oplus \delta y_{i+1}$ has to equal them.

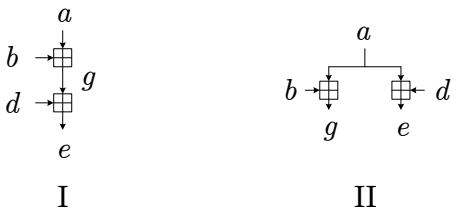

Fig. 1: Examples of Incompatible Conditions

2.1 Examples of Incompatible Conditions

In this section, we show some examples in which using Lipmaa's conditions with the independency assumption between the consecutive components of the block cipher is not enough to ensure the validity of the differential characteristic.

Example 1: Consider the two cascaded modular operations shown in Figure (1.I) and the following XOR differences:

$$\begin{array}{llll} \Delta \boldsymbol{a} = 00000001 & \boldsymbol{g} = \boldsymbol{a} \boxplus \boldsymbol{b} & & \boldsymbol{e} = \boldsymbol{g} \boxplus \boldsymbol{d} \\ \Delta \boldsymbol{b} = 00000000 & \Delta \boldsymbol{g} = 00001111 & \Delta \boldsymbol{d} = 00000000 & \Delta \boldsymbol{e} = 00001101 \end{array}$$

Table 2: Relation between $\delta x_i, \delta y_i, \delta z_i$ and δc_{i+1}

δz_i	δy_i	δx_i	δc_i	δc_{i+1}	Condition
0	0	0	0	0	No condition
0	0	0	0	1	Invalid
0	0	1	1	0	$x_i = \bar{c}_i \mid z_i = \bar{y}_i \mid c_{i+1} = y_i = \bar{z}_i$
0	0	1	1	1	$x_i = c_i \mid z_i = y_i \mid c_{i+1} = x_i = c_i$
0	1	0	1	0	$y_i = \bar{c}_i \mid z_i = \bar{x}_i \mid c_{i+1} = x_i = \bar{z}_i$
0	1	0	1	1	$y_i = c_i \mid z_i = x_i \mid c_{i+1} = y_i = c_i$
0	1	1	0	0	$x_i = \bar{y}_i \mid z_i = \bar{c}_i \mid c_{i+1} = c_i = \bar{z}_i$
0	1	1	0	1	$x_i = y_i \mid z_i = c_i \mid c_{i+1} = x_i = y_i$
1	0	0	1	0	$x_i = y_i \mid z_i = c_i \mid c_{i+1} = x_i = y_i$
1	0	0	1	1	$x_i = \bar{y}_i \mid z_i = \bar{c}_i \mid c_{i+1} = c_i = \bar{z}_i$
1	0	1	0	0	$y_i = c_i \mid z_i = x_i \mid c_{i+1} = y_i = c_i$
1	0	1	0	1	$y_i = \bar{c}_i \mid z_i = \bar{x}_i \mid c_{i+1} = x_i = \bar{z}_i$
1	1	0	0	0	$x_i = c_i \mid z_i = y_i \mid c_{i+1} = x_i = c_i$
1	1	0	0	1	$x_i = \bar{c}_i \mid z_i = \bar{y}_i \mid c_{i+1} = y_i = \bar{z}_i$
1	1	1	1	0	Invalid
1	1	1	1	1	No condition

When looking at each modular addition operation individually, each one satisfies the Lipmaa's conditions and holds with probability 2^{-4}. Assuming independency, the whole differential characteristic should hold with probability 2^{-8}, however, it is actually an impossible characteristic. To explain, using Table 2, we can show that if the characteristic holds for the first operation, $\boldsymbol{g} = (g_{n-1}, \cdots, g_1, g_0)$ will have a specific pattern $(g_1 = g_0)$ due to the carry effect. On the other hand, the characteristic will hold for the second modular addition if $\boldsymbol{g}$ has a specific pattern $(g_1 = \bar{g}_0)$, also due to the carry effect.

To further explain this carry effect, consider for the first operation the differences of the first three bits $(\delta g_0, \delta b_0, \delta a_0) = (1, 0, 1)$, $(\delta g_1, \delta b_1, \delta a_1) = (1, 0, 0)$ and $(\delta g_2, \delta b_2, \delta a_2) = (1, 0, 0)$. We access Table 2 twice with $(\delta z_i, \delta y_i, \delta x_i, \delta c_i, \delta c_{i+1})$ $= (\delta g_0, \delta b_0, \delta a_0, \delta c_0, \delta c_1) =$ (1,0,1,0,1) where the carry $\delta c_0 = \delta g_0 \oplus \delta b_0 \oplus \delta a_0$ and the carry $\delta c_1 = \delta g_1 \oplus \delta b_1 \oplus \delta a_1$, and with $(\delta z_i, \delta y_i, \delta x_i, \delta c_i, \delta c_{i+1}) = (\delta g_1, \delta b_1, \delta a_1,$ $\delta c_1, \delta c_2) =$ (1,0,0,1,1) where the carry $\delta c_2 = \delta g_2 \oplus \delta b_2 \oplus \delta a_2$. From the first access, we get the following condition:

$$b_0 = \bar{c}_0 \Rightarrow g_0 = \bar{a}_0 \text{ and } c_1 = a_0 = \bar{g}_0 \tag{5}$$

And from the second access, we get the condition:

$$a_1 = \bar{b}_1 \Rightarrow g_1 = \bar{c}_1 \text{ and } c_2 = c_1 = \bar{g}_1 \tag{6}$$

From equation (5), if the characteristic is valid for the first bit, the carry bit c_1 will equal to $\bar{g}_0$. Also, if the characteristic is valid for the second bit, the same carry bit c_1 will have a relation with g_1 as determined by equation (6). By combining these two relations, we prove that the output $\boldsymbol{g}$ has the pattern $(g_1 = g_0)$.

Using the same methodology, we can also prove that the characteristic will hold for the second operation if the input $\boldsymbol{g}$ has the pattern $(g_1 = \bar{g}_0)$ which contradicts with the output of the first operation. All these patterns have also been verified experimentally.

Example 2: Let us consider another ordering of two modular operations as shown in Figure (1.II) and the following XOR differences:

$$\begin{array}{llll} \Delta\boldsymbol{a} = 00001111 & \boldsymbol{g} = \boldsymbol{a} \boxplus \boldsymbol{b} & & \boldsymbol{e} = \boldsymbol{a} \boxplus \boldsymbol{d} \\ \Delta\boldsymbol{b} = 00000001 & \Delta\boldsymbol{g} = 00010000 & \Delta\boldsymbol{d} = 00000001 & \Delta\boldsymbol{e} = 00000000 \end{array}$$

Again, the two operations individually satisfy the Lipmaa's conditions. However, the first operation requires the input $\boldsymbol{a}$ to be in a specific pattern $(a_0 = a_1 = a_2 = a_3)$ and the second operation requires the input $\boldsymbol{a}$ to be in another contradicting pattern $(a_0 = a_1 = a_2 = \bar{a}_3)$.

3 New MILP Model for Differential Characteristics of Modular Addition

Fu *et al.* [10] represent Lipmaa's conditions by a set of MILP constraints in order to automate the search for the best differential trail through the modular addition. As explained in the previous section, Lipmaa's conditions are not enough to ensure the validity of the derived differential characteristic especially when the block cipher structure has two or more consecutive modular additions. We propose a more accurate MILP model to automate the search for differential characteristics through modular additions taking into account the dependency between two consecutive modular additions that put more constraints on the values of input and output bits.

In order to represent the relation between two consecutive bits i and $i-1$ on a variable $\boldsymbol{x}$, we define a new variable called $x_i^{\oplus} = x_i \oplus x_{i-1}$ which can take a value of $\{0, 1, ?\}$; it is set to 0 if the condition $x_i = x_{i-1}$ is required and set to 1 if the condition $x_i = \bar{x}_{i-1}$ is required. Also, $x_i^{\oplus}$ can be kept undetermined (?) which means it can be 0 or 1 if there is no restriction on the relation between x_i and x_{i-1}.

Evaluation of $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})$ for a modular addition. The relation between the bits x_i and x_{i-1}, for the input $\boldsymbol{x}$ in a modular addition comes through the carry bit c_i. Therefore the variable $x_i^{\oplus}$ can be evaluated as:

$$x_i^{\oplus} = (x_i \oplus c_i) \oplus (c_i \oplus x_{i-1})$$

where $x_i \oplus c_i$ and $c_i \oplus x_{i-1}$ can take a value of $\{0, 1, ?\}$ like $x_i^{\oplus}$ and the bit-wise XOR of ? with any value equals to ?. Based on Table 2, the values of $(x_i \oplus c_i)$ and $(c_i \oplus x_{i-1})$ reflect the situation where there are conditions that should be satisfied to get the XOR differences $(\delta z_i, \delta y_i, \delta x_i, \delta c_{i+1})$ and $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta c_i)$, respectively. Thus, the values of $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})$ will be determined based on the XOR differences $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i, \delta c_{i+1})$. We develop Algorithm 1 to determine these values. The input of our proposed algorithm is a general-purpose data structure dictionary $\mathbb{D}$ which is obtained by reformatting the valid rows in Table 2 where the relations between the current bits (z, y, x) with the current carry bit c and the subsequent carry bit c_{+1} are derived from the condition column in Table 2 and indexed by the value of the XOR difference of these bits, see Table 3. The output of Algorithm 1 is the truth table $\mathbb{T}$ of $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})$ as a function of the possible XOR differences $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i, \delta c_{i+1})$. Out of $2^7 = 128$ values of these bits, there are only 98 values that can be used as possible differences. Table 4 shows part of the derived truth table $\mathbb{T}$.

MILP constraints for Modular Addition. To automate the process of the search for the differential characteristic using MILP technique, we have to transform the truth table $\mathbb{T}$ into a set of linear constraints. To this end, we represent the rows of $\mathbb{T}$ combined with the value of $\neg eq(\delta z_i, \delta y_i, \delta x_i)$ as a set of points in 11-dimensional binary vector space by substituting ? with all possible values *e.g.,* the row (0010010??1) associated with $\neg eq(0, 0, 1) = 1$ will be described by 4 binary vectors: (00100100011, 00100100111, 00100101011, 00100101111). After this step, we have 640 binary vectors which have a convex hull. We use the `inequality_generator()` function in Sage[2] to obtain the H-Representation which is a set of linear inequalities that describe the vectors of this convex hull. We can use this set of inequalities as MILP constraints to present the possible XOR differences in two successive bits $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i)$ and the carry of the third bits (δc_{i+1}) combined with the conditions on the value of these bits represented as $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})$. In our case, the number of generated inequalities is 313, which is very large to be handled by any MILP optimizer. Therefore, we employ the Greedy algorithm proposed by Sun *et al.* in [23] to reduce this set to only 24 inequalities. In order to link the current bit with the following bits, we encoded equation (4), which is a bit-wise XOR of three inputs and one output, by 8 linear inequalities utilizing the truth table of the bit-wise XOR and `inequality_generator()` function in Sage. In this manner, we have represented the relation between three successive bits using $24 + 8 = 32$ inequalities and this representation is repeated for $i = 1, 2, \ldots, n-2$. In order to complete the MILP modeling for the modular addition, we describe the condition on the first bit $(i = 0)$ $\delta z_0 \oplus \delta y_0 \oplus \delta x_0 = 0$ associated with $\neg eq(\delta z_0, \delta y_0, \delta x_0)$ by 4 linear inequalities. Accordingly, we can represent the difference propagation through the addition modulo 2^n taking into account the relation between the value of two successive bits using $32 \times (n-2) + 4$ inequalities. The objective function of

[2] http://www.sagemath.org/

the MILP optimizer would minimize $\sum_{i=0}^{n-2} \neg eq(\delta z_i, \delta y_i, \delta x_i)$, which denotes the log_2 probability of the underlying characteristic.

Algorithm 1: Truth table generator

Input : The Dictionary $\mathbb{D}$.
Output: The truth table $\mathbb{T}$ of $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})$ as a function of the possible XOR differences $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i, \delta c_{i+1})$

begin
  $\mathbb{T} = \emptyset$
  for 2^7 possible values of $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i, \delta c_{i+1})$ **do**
    $\delta c_{i-1} \leftarrow \delta z_{i-1} \oplus \delta y_{i-1} \oplus \delta x_{i-1}$
    $\delta c_i \leftarrow \delta z_i \oplus \delta y_i \oplus \delta x_i$
    if $(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta c_{i-1}, \delta c_i)$ in $\mathbb{D}$.keys `AND` $(\delta z_i, \delta y_i, \delta x_i, \delta c_i, \delta c_{i+1})$ in $\mathbb{D}$.keys **then**
      $\mathsf{RCarry1} \leftarrow \mathbb{D}[(\delta z_i, \delta y_i, \delta x_i, \delta c_i, \delta c_{i+1})][0]$
      $\mathsf{RCarry2} \leftarrow \mathbb{D}[(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta c_{i-1}, \delta c_i)][1]$
      $(z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus}) \leftarrow \mathsf{RCarry1} \oplus \mathsf{RCarry2}$
      $\mathbb{T} \leftarrow \mathbb{T} \cup \{(\delta z_{i-1}, \delta y_{i-1}, \delta x_{i-1}, \delta z_i, \delta y_i, \delta x_i, \delta c_{i+1}, z_i^{\oplus}, y_i^{\oplus}, x_i^{\oplus})\}$
    end
  end
  return $\mathbb{T}$
end

4 Application on Bel-T

4.1 Bel-T Specification

Since the official Bel-T specification is available only in Russian, we rely on the English version of the specification that is provided by Jovanovic and Polian, who presented fault-based attacks on the Bel-T block cipher family [11]. Bel-T has a 128-bit block size and a variable key length of 128, 192 or 256 bits. The 128-bit plaintext P is split into 4 32-bit words, *i.e.,* $P = A_0^0||B_0^0||C_0^0||D_0^0$. The round function of Bel-T consists of 7 S-box layers in which a 32-bit mapping function (G_r) is combined with one or two modulo operations as illustrated in Fig. 2. Then, this round function is repeated 8 times for all versions of Bel-T. The function G_r (G-box) maps a 32-bit word $w = w_1||w_2||w_3||w_4$, with $w_i \in \{0,1\}^8$, as follows: $G_r(w) = (H(w_1)||H(w_2)||H(w_3)||H(w_4)) \lll r$. Here, H is an 8-bit S-box and $\lll r$ denotes left shift rotation by r positions ($r \in \{5, 13, 21\}$). The specification of the 8-bit S-box can be found in [11].

Key Schedule. In all versions of Bel-T, the 128-bit plaintext block P is encrypted using a 256-bit encryption key denoted as $K_1||\ldots||K_8$, where K_i is a 32-bit word for $1 \leq i \leq 8$. The encryption key is distributed among the round

Table 3: The dictionary $\mathbb{D}$.

$\mathbb{D}$.keys					$\mathbb{D}[*][0]$	$\mathbb{D}[*][1]$
δz	δy	δx	δc	δc_{+1}	$c \oplus (z,y,x)$	$c_{+1} \oplus (z,y,x)$
0	0	0	0	0	(? , ? , ?)	(? , ? , ?)
0	0	1	1	0	(? , ? , 1)	(1 , 0 , ?)
0	0	1	1	1	(? , ? , 0)	(? , ? , 0)
0	1	0	1	0	(? , 1 , ?)	(1 , ? , 0)
0	1	0	1	1	(? , 0 , ?)	(? , 0 , ?)
0	1	1	0	0	(1 , ? , ?)	(1 , ? , ?)
0	1	1	0	1	(0 , ? , ?)	(? , 0 , 0)
1	0	0	1	0	(0 , ? , ?)	(? , 0 , 0)
1	0	0	1	1	(1 , ? , ?)	(1 , ? , ?)
1	0	1	0	0	(? , 0 , ?)	(? , 0 , ?)
1	0	1	0	1	(? , 1 , ?)	(1 , ? , 0)
1	1	0	0	0	(? , ? , 0)	(? , ? , 0)
1	1	0	0	1	(? , ? , 1)	(1 , 0 , ?)
1	1	1	1	1	(? , ? , ?)	(? , ? , ?)

Table 4: Part of the truth table $\mathbb{T}$.

δz_{i-1}	δy_{i-1}	δx_{i-1}	δz_i	δy_i	δx_i	δc_{i+1}	$z_i^{\oplus}$	$y_i^{\oplus}$	$x_i^{\oplus}$
				....					
0	0	1	0	0	1	0	?	?	1
0	0	1	0	0	1	1	?	?	0
0	0	1	0	1	1	0	0	?	?
0	0	1	0	1	1	1	1	?	?
0	0	1	1	0	1	0	?	0	?
0	0	1	1	0	1	1	?	1	?
0	1	0	0	1	0	0	?	1	?
0	1	0	0	1	0	1	?	0	?
0	1	0	0	1	1	0	0	?	?
0	1	0	0	1	1	1	1	?	?
0	1	0	1	1	0	0	?	?	0
0	1	0	1	1	0	1	?	?	1
0	1	1	0	0	1	0	?	?	1
0	1	1	0	0	1	1	?	?	0
0	1	1	0	1	0	0	?	1	?
0	1	1	0	1	0	1	?	0	?
0	1	1	0	1	1	0	0	?	?
				....					

keys as shown in Table 5. The encryption key is extracted from the master key as follows:

- Bel-T-256: the encryption key is identical to the master key.
- Bel-T-192: the master key is formatted as $K_1||\dots||K_6$ and K_7, K_8 are set to $K_7 := K_1 \oplus K_2 \oplus K_3$ and $K_8 := K_4 \oplus K_5 \oplus K_6$.
- Bel-T-128: the master key is formatted as $K_1||\dots||K_4$ and K_5, K_6, K_7, K_8 are set to $K_5 := K_1$, $K_6 := K_2$, $K_7 := K_3$ and $K_8 := K_4$.

4.2 MILP-based Search for Differential Characteristic of Bel-T

To search for differential characteristics in a block cipher using MILP, the difference propagation through its components is described using a set of linear constraints. In Bel-T, this means generating a set of linear inequalities to describe how an XOR difference would propagate through a bit-wise XOR, an addition/subtraction modulo 2^{32}, and an 8-bit S-box. As the difference propagates with probability through the non-linear components, its associated probability is incorporated in the corresponding linear inequalities. The objective function of the MILP model would be to maximize this probability, which we do by minimizing the negative of the base-2 logarithm of this probability.

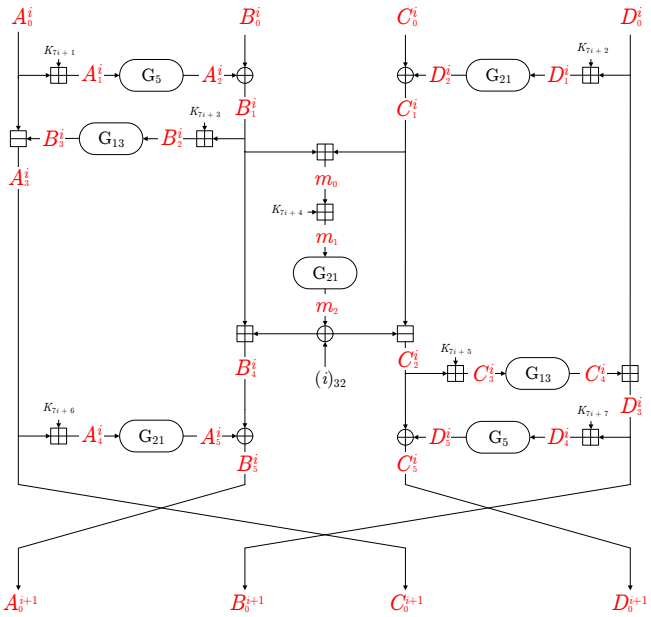

Fig. 2: Bel-T round function. $\oplus, \boxplus, \boxminus$ denote bit-wise XOR, arithmetic addition and subtraction modulo 2^{32} respectively, and $(i)_{32}$ denotes the round number represented as 32-bit word.

Table 5: Encryption Key schedule of Bel-T, where i and K_{7i+j} denote the round number and the round key, respectively.

i	K_{7i+1}	K_{7i+2}	K_{7i+3}	K_{7i+4}	K_{7i+5}	K_{7i+6}	K_{7i+7}
0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
1	K_8	K_1	K_2	K_3	K_4	K_5	K_6
2	K_7	K_8	K_1	K_2	K_3	K_4	K_5
3	K_6	K_7	K_8	K_1	K_2	K_3	K_4
4	K_5	K_6	K_7	K_8	K_1	K_2	K_3
5	K_4	K_5	K_6	K_7	K_8	K_1	K_2
6	K_3	K_4	K_5	K_6	K_7	K_8	K_1
7	K_2	K_3	K_4	K_5	K_6	K_7	K_8

Bit-wise XOR. If $\delta x_i, \delta y_i$ and δz_i represent the bit-level differences, then the difference propagation through the bit-wise XOR operation $\delta x_i \oplus \delta y_i = \delta z_i$ can be represented by 5 linear inequalities [23]. Using the truth table of the XOR operation, these can be further reduced to the following 4 linear inequalities:

$$\delta x_i + \delta y_i - \delta z_i \geq 0,\ \delta x_i - \delta y_i + \delta z_i \geq 0,\ -\delta x_i + \delta y_i + \delta z_i \geq 0,\ -\delta x_i - \delta y_i - \delta z_i \geq -2.$$

Modular Addition and Subtraction. We use the new MILP model described in Section 3 to propagate the input differences $(\boldsymbol{\Delta x}, \boldsymbol{\Delta y})$ to an output difference

($\Delta \boldsymbol{z}$) through the addition modulo 2^{32} such that $\boldsymbol{x} \boxplus \boldsymbol{y} = \boldsymbol{z}$ using $32 \times (32-2)+4 = 964$ inequalities. Since the subtraction modulo 2^n, $\boldsymbol{x} \boxminus \boldsymbol{y} = \boldsymbol{z}$ is equivalent to $\boldsymbol{x} = \boldsymbol{y} \boxplus \boldsymbol{z}$, the difference propagation through modular subtraction can be described in a similar way as that used to describe modular addition.

Modular Addition with a Secret Key. The Bel-T round function encompasses a modular addition with a secret key which has zero difference in a single-key differential attack. This operation can then be expressed as $\boldsymbol{x} \boxplus \boldsymbol{k} = \boldsymbol{z}$ and the differential characteristic as $(\Delta \boldsymbol{x}, 0) \rightarrow \Delta \boldsymbol{z}$. Therefore, the difference propagation through this operation can be described in a similar way as that used to describe modular addition by inserting 32 more constraints to explicitly set $\Delta \boldsymbol{y} = 0$. The number of required constraints will be $964 + 32 = 996$. Indeed, we can improve this description by decreasing the number of MILP constraints to roughly half as follows. We repeat the steps described in Section 3 using the rows of the truth table $\mathbb{T}$ that have $\delta y_{i-1} = \delta y_i = 0$ and also $\delta y_{i+1} = 0$. Consequently, the number of MILP constraints decreases to $(13 + 4)(32 - 2) + 2 = 512$.

8-bit S-box. Using the Sage `inequality_generator()` function to model the DDT of an 8-bit S-box is computationally infeasible. Therefore, the use of MILP to search for differential characteristics was restricted to block ciphers that do not include 8-bit S-boxes. Abdelkhalek *et al.* [2] have put forward an approach to model the DDT of an 8-bit S-box efficiently. First, the DDT is split into several tables corresponding to unique probability values. After assigning binary variables to each unique probability value, these binary variables are represented as Boolean functions in the input and output difference bits, i.e., each Boolean function is 1 when the input difference is propagated to the output difference with the corresponding probability value, and 0 otherwise. Next, the Quine-McCluskey algorithm [18,16] was used to transform the Boolean functions to their reduced Product of Sum (PoS) which can then be described by a set of linear inequalities. To describe the deterministic propagation of the zero-difference, an additional binary variable was used as a sort of flag, i.e., when it is 0, the S-box is inactive and therefore both the input and output differences are set to 0. When it is 1, the S-box is active and one probability value along with input difference and corresponding output difference are chosen. As in ARX block ciphers, the probability of the differential characteristic gets lower when more bits are active, we decided to follow the approach in [3] in which we do not use the high probability entries in the DDT, but rather the entries with low Hamming weight in the input and output differences. Throughout our experiments, we have limited the Hamming weight of the input and output difference not to exceed 3. However, the partial DDT was still too large to be handled directly using the `inequality_generator()` function and hence we augmented our approach with the approach proposed by Abdelkhalek *et al.* for handling the DDT of large S-boxes to describe the partial DDT using linear inequalities. Based on our implementation, $1,660$ linear inequalities are needed to describe this Hamming weight-based partial DDT.

Lai-Massey Scheme. Since the Lai-Massey scheme is invertible, the following constraints are added to our model to enforce the output of the Lai-Massey scheme (B_4^i, C_2^i) to be non-zero when its input (B_1^i, C_1^i) is non-zero, see Fig. 2.

$$\begin{aligned}
\sum_{j=0}^{n-1} B_{1,j}^i + \sum_{j=0}^{n-1} C_{1,j}^i + LM_i &\geq 1,\\
\sum_{j=0}^{n-1} B_{4,j}^i + \sum_{j=0}^{n-1} C_{2,j}^i + 2n \times LM_i &\leq 2n,\\
\sum_{j=0}^{n-1} B_{1,j}^i + \sum_{j=0}^{n-1} C_{1,j}^i + 2n \times LM_i &\leq 2n,\\
\sum_{j=0}^{n-1} B_{4,j}^i + \sum_{j=0}^{n-1} C_{2,j}^i + LM_i &\geq 1.
\end{aligned}$$

In these constraints, LM_i is a dummy binary variable. If the input difference is zero, the first equation enforces LM_i to be 1 which enforces the output difference to be zero in the second equation. If the input difference is non-zero, the third equation enforces LM_i to be 0 which enforces the output difference to be non-zero in the fourth equation.

4.3 3-round Differential Characteristic

Using the above derived MILP model of the different components of the Bel-T, we are able to build a model of the whole round of Bel-T using $55,641$ linear inequalities and $2,647$ binary variables. Then, we used the Gurobi[3] optimizer on a server of two Xeon Processors E5-2697 ($2 \times 12 = 24$ cores in total) with 125 GB RAM to search for a differential characteristic of Bel-T. Consequently, we found a 2-round differential characteristic with probability 2^{-54} after about 4.5 hours. We use this characteristic as an initial solution for the optimizer in order to extend the characteristic to 3 rounds. After running the search process for 36 days, we were not able to find a 3-round differential characteristic better than the one that holds with probability 2^{-111}. The 3-round differential characteristic we use in our attack is shown in Fig. 3 in which 0 denotes a 32-bit difference of all zeros, e_i, e_{i-j} and $e_{i,j,k,\cdots}$ denote 32-bit difference of all 0's and 1 at bit i, bits i to j, and bits $i, j, k, \cdots$, respectively.

4.4 Validity of The Differential Characteristic

In this section, we show that Bel-T block cipher is not a Markov cipher and the differential characteristic depends on the used secret key. Consequently, we propose a systematic way to obtain the ratio of the keys that can be attacked using our distinguisher.

[3] http://www.gurobi.com/

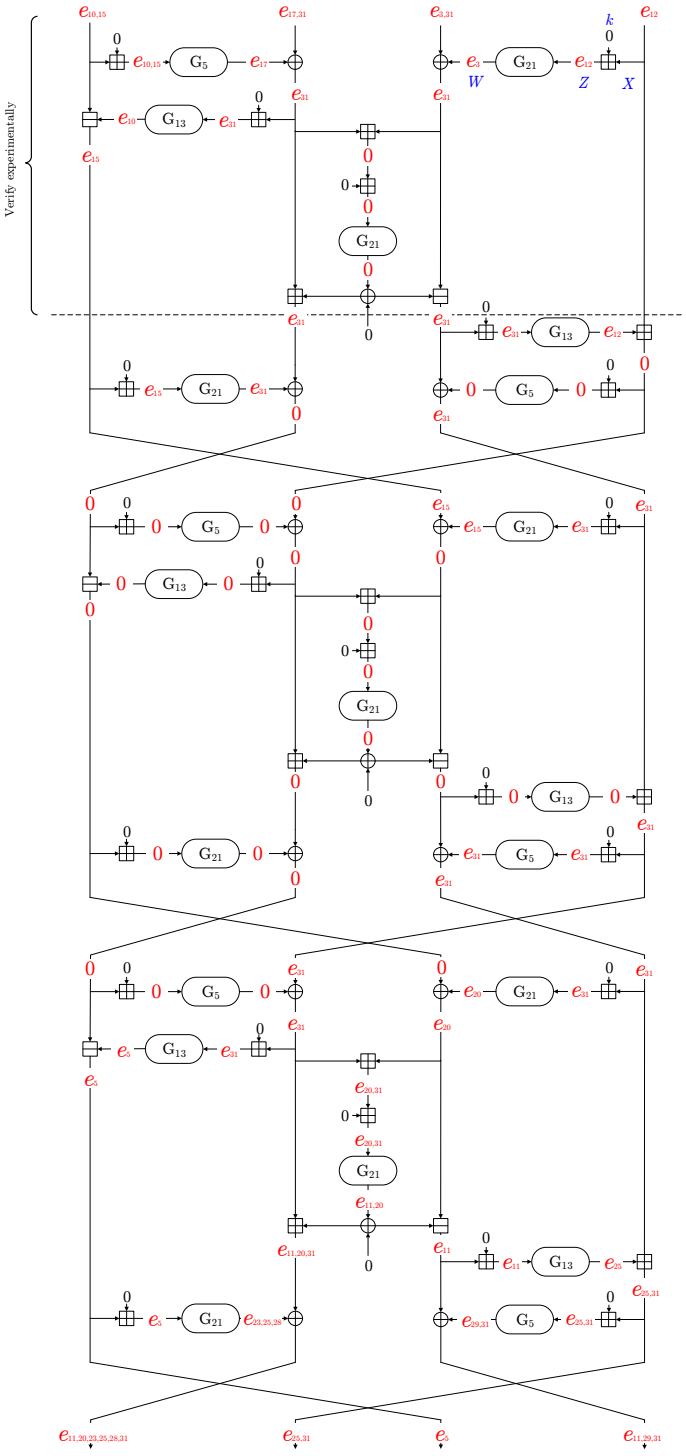

Fig. 3: 3-round Differential Characteristic of Bel-T with Probability 2^{-111}. 0 denotes a 32-bit difference of all zeros, e_i, e_{i-j} and $e_{i,j,k}$ denote a 32-bit difference of 0's and 1 at bit i, bits i to j, and bits i, j, and k, respectively

Recall that a Markov cipher [13] is an iterated block cipher in which the probability of the difference *e.g.,* the XOR difference through the individual operations of the round function is independent of the corresponding plaintext values of its input, if the round keys applied to each round are independent and chosen in a uniformly random manner. In the case of Bel-T, the secret key is mixed via modular addition operations, therefore the XOR difference propagation through these operations is probabilistic and depends on the used key. Additionally, the hypothesis of independent round keys does not hold due to the simple key schedule of Bel-T. Moreover, there are many two or more successive modular additions, which are not independent as shown in Section 2. For these reasons, we can conclude that Bel-T is not a Markov cipher.

Since the secret key is mixed via modular addition operations, Bel-T is not a *key-alternating* cipher [7] and the probability of the XOR difference of these modular operations may drop to zero due to the used key [5] and we therefore cannot use our distinguisher in this case. In the remaining of this section, we obtain the ratio of the keys (valid keys) which we can use the distinguisher with. We define the S-box layer to include the modular addition with a key followed by the G-box mapping (G_r). We consider a 32-bit key as an invalid key when the probability of the XOR difference through its S-box layer drops to zero independent of the other input of the modular addition.

Let us consider, *e.g.,* the S-box layer of K_2 in round 0 (see Fig. 3) in which the key K_2 has a specific value k, $Z = X \boxplus k$ and $W = G_{21}(Z)$ where $\Delta X = \Delta Z = 0x00001000$, $\Delta k = 0x00000000$ and $\Delta W = 0x00000008$. Therefore, we are looking for the values of k that cannot give the output difference ΔW for any value of X.

For each value of k, we can exhaustively search over all possible values of the pair $(X, X \oplus \Delta X)$ to check if there is a value of X that leads to the output difference ΔW. If there is no such value, we consider k as invalid. The complexity of search for all possible values of K_2 will be roughly $\mathcal{O}(2^{64})$ which is computationally hard because we will repeat this search for all modular additions with keys.

Alternatively, we can obtain from Table 2 that the condition $k_{12} = c_{12}$, where k_{12} and c_{12} are the bit number 12 of the key and the carry respectively, is the only constraint that has to be checked to verify whether the key k is an invalid key or not. Also from the DDT of the G-box, the second byte of Z (bits from Z_8 to Z_{15}) in hexadecimal has to be one of $\{0x02, 0x12, 0x4C, 0x5C\}$ to satisfy the output difference ΔW. Accordingly, the following constraints have to be satisfied:

$$k_{12} = c_{12}, \qquad Z_8 = 0, \qquad Z_{13} = 0, \qquad Z_{15} = 0, \qquad \bar{Z_9} = Z_{10} = Z_{11} = Z_{14}.$$

For each value of k, there is a value X that gives $Z_8 = 0$ with probability 1 because there are no conditions on k nor Z from bit 0 to 7. Given this fact and by using equations (1) and (2), we can prove that the carry bits $c_9 = c_{10} = c_{11} = c_{12} = 0$ if the key bits $k_8 = 1$ and $k_9 = k_{10} = k_{11} = 0$ independently

of the corresponding bits of X. Therefore, if the key bit $k_{12} = 1$, the condition $k_{12} = c_{12}$ will be impossible. As a result, if the key k has the pattern $k_8 = k_{12} = 1$ and $k_9 = k_{10} = k_{11} = 0$, it will be an invalid key irrespective of the value X due to the contradiction between the two constraints $Z_8 = 0$ and $k_{12} = c_{12}$. We can manually search for such patterns but this process is very difficult, time-consuming, and error-prone.

Observation 1 *Consider a modular addition $\boldsymbol{z} = \boldsymbol{x} \boxplus \boldsymbol{y}$ where the bit z_i has a specific value. Then, the carry bit c_j (for $j > i$) depends on the input bits from i to $j-1$ and is independent of the input bits from 0 to $i-1$.*

The dependency between a carry bit c_j and the input bits from 0 to $j-1$ is due to the carry chain (see equation 2). If we know that the output bit z_i has a specific value, we can evaluate the carry bit c_i as $c_i = z_i \oplus x_i \oplus y_i$ instead of evaluating it using the value of x_{i-1}, y_{i-1} and c_{i-1}. Thus, the carry chain and dependency are broken. Back to our example, given that $Z_8 = 0$, the carry bit c_{12} will depend on the bits from 8 to 11 of the inputs X and k based on the observation. Therefore, considering the key k as an invalid will depend on its bits from 8 to 12. In general, given a key k, if we exhaustively search over all possible values of the pair $(X, X \oplus \Delta X)$ and there is no value X that can lead to the difference ΔW, then the byte of the key containing the conditional bits is the reason for invalidating ΔW. We therefore can repeat the search for all possible value of these bytes. Consequently, the exhaustive search complexity in our example will be reduced roughly to $\mathcal{O}(2^{40})$ which is feasible.

The above approach can be generalized to determine the set of the byte values $\mathbb{K}$ leading to invalid keys as shown in Procedure (Obtain Invalid Key Set).

Table 6 summarizes the ratio of valid keys of each key K_i that has conditions in our distinguisher. It should be noted that the key K_2 is used in two rounds but the bytes that have the conditions are in different positions. Accordingly, the total ratio of the valid keys can be evaluated as the multiplication of all ratios of the valid keys which will be $2^{-3.8}$ corresponding to $2^{252.2}$ keys. In order to validate this result, we have experimentally verified the differential characteristic. In particular, we have opted the first four S-box layers of the differential characteristic of probability 2^{-24} (see Fig. 3) and have found that the experimental probability matches on average the theoretical one for 4426 of 10000 randomly generated keys. Comparing with Table 6, this ratio is very close to the ratio of the valid keys for this part of the distinguisher.

5 Differential Attack on $4\frac{1}{7}$-Round Reduced Bel-T-256

In this section, we present a differential attack on $4\frac{1}{7}$-round reduced Bel-T-256 by appending one round and one S-box layer on the above derived differential distinguisher as illustrated in Fig. 4. Our differential characteristic ends at A_0^3, B_0^3, C_0^3 and D_0^3 with values $e_{11,20,23,25,28,31}$, $e_{25,31}$, e_5 and $e_{11,29,31}$, respectively. Therefore, by propagating the differences at A_0^3 and D_0^3 through the S-box layers, we

Procedure Obtain Invalid Key Set

Input : $\Delta X, \Delta W$
Output: $\mathbb{K}$
begin
  $\mathbb{K} = \emptyset$
  Determine `PosOfBytes` and `NBytes` which are the position and the number of bytes that have XOR difference in ΔX
  for $2^{8\times \texttt{NBytes}}$ possible values of `Bytes` **do**
    Generate k randomly such that the concatenation of the bytes in the position `PosOfBytes` has the value `Bytes`
    `invalid` = True
    for 2^{32} possible values of X **do**
      if $G(X \boxplus k) \oplus G((X \oplus \Delta X) \boxplus k) = \Delta W$ **then**
        `invalid` = False
        break
      end
    end
    if `invalid` **then**
      $\mathbb{K} \leftarrow \mathbb{K} \cup \{\texttt{Bytes}\}$
    end
  end
  return $\mathbb{K}$
end

Table 6: Ratio of valid keys

Round	Key	Ratio of valid keys
0	K_1	136/256
	K_2	216/256
	K_6	129/256
2	K_2	216/256
	K_3	144/256
	K_4	228/256
	K_5	192/256

Table 7: The difference at the points used in the attack

Point label	The difference in Binary
A_0^3	10010010 10010000 00001000 00000000
B_0^3	10000010 00000000 00000000 00000000
C_0^3	00000000 00000000 00000000 00100000
D_0^3	10100000 00000000 00001000 00000000
B_1^3	???00000 000????? ???????? ????????
C_1^3	???????? ???????? ???00000 000?????

obtain the corresponding 32-bit difference at B_1^3 and C_1^3. Table 7 summarizes the difference in Binary at some points that we will use during the attack. Our attack has two phases: pre-computation phase and an online phase.

5.1 Pre-computation Phase

In this phase, we create 4 hash tables (H_1, H_2, H_3, H_4) corresponding to the S-box layers shown in Fig. 4 as follows:

H_1 : For all $2^{5\times32=160}$ possible values of $x, \Delta x, y, \Delta y$ and K_2, we obtain the corresponding values of z and Δz such that $z = y \boxminus G_{13}(x \boxplus K_2)$. If the value of Δz is equal to the difference at D_0^3, we store the values of K_2 and z in the hash table H_1 indexed by the values of $x, \Delta x, y$ and Δy. The probability that the value of Δz is equal to the difference at D_0^3 is equal to 2^{-32}. Therefore, Table H_1 has on average $2^{160} \times 2^{-32} = 2^{128}$ entries. As a result, we have, on average, $\frac{2^{128}}{2^{4\times32}} = 1$ value for K_2 per row.

H_2 : For the value of Δx equal to the difference at D_0^3 and all 2^{24} possible value of Δy in form of the difference at C_1^3 combined with all $2^{3\times32=96}$ possible values of x, y and K_7, we obtain the corresponding values of z and Δz such that $z = y \oplus G_{21}(x \boxplus K_7)$. Then, we store the value of K_7 in the hash table H_2 indexed by the values of x, y and Δy, if the value of Δz is equal to the difference at C_0^3 which has a probability equal to 2^{-24}. Therefore, Table H_2 has on average $2^{96+24} \times 2^{-24} = 2^{96}$ entries. Thus, we have, on average, $\frac{2^{96}}{2^{2\times32+24}} = 2^8$ value for K_7 per row.

H_3 : For all 2^{24} possible value of Δx in form of the difference at B_1^3 combined with all $2^{4\times32=128}$ possible values of $x, y, \Delta y$ and K_8, we obtain the corresponding values of z and Δz such that $z = y \boxplus G_{13}(x \boxplus K_8)$. If the value of Δz is in the form of the difference at A_0^3, we store the values of K_8 and z in the hash table H_3 indexed by the values of $x, \Delta x, y$ and Δy. The probability that the value of Δz is in the form of the difference at A_0^3 is equal to 2^{-32}. Therefore, Table H_3 has on average $2^{128+24} \times 2^{-32} = 2^{120}$ entries. As a result, we have, on average, $\frac{2^{120}}{2^{3\times32+24}} = 1$ values for K_8 per row.

H_4 : Initialize a hash table of $2^{3\times32+24=120}$ rows with binary value 0. Then, for the value of Δx equal to the difference at A_0^3 and all 2^{24} possible values of Δy in the form of the difference at B_1^3 combined with all $2^{3\times32=96}$ possible values of x, y, and K_6, we obtain the corresponding values of z and Δz such that $z = y \oplus G_5(x \boxplus K_6)$. If the value of Δz is equal to the difference at B_0^3, we store a binary value 1 in the hash table H_4 indexed by the values of $x, y, \Delta y$ and K_6. Here, the binary values 1 and 0 denote a valid entry and an invalid entry. The probability of finding a valid entry in H_4, equivalent to the probability that the value Δz is equal to the difference at B_0^3, is equal to 2^{-24}. Consequently, we have one valid entry for every 2^{24} accesses to H_4.

Table 8 summarizes the time and memory complexities of the pre-computation phase. It should be noted that the memory required by the tables H_1 and H_4 can be slightly reduced to $2^{128.51}$ and $2^{119.01}$ 32-bit words respectively, if we store only the valid candidates of K_2 and K_6 based on the ratio of the valid keys form Table 6.

5.2 Online Phase

In this phase, we collect a set of plaintext/ciphertext pairs. Then, we utilize the pre-computation tables and key guessing to obtain right candidate keys and then recover the correct master key.

Table 8: The time and memory complexities of the pre-computation phase

Table	Time (S-box layer Encryption)	Memory (32-bit word)
H_1	2^{160}	$2^{160} \times 2^{-32} \times 2 = 2^{129}$
H_2	2^{120}	$2^{120} \times 2^{-24} \times 1 = 2^{96}$
H_3	2^{152}	$2^{152} \times 2^{-32} \times 2 = 2^{121}$
H_4	2^{120}	2^{120} §

Data Collection. We select a set of 2^m 128-bit plaintexts that can take any arbitrary values then we compute another set of 2^m plaintexts by XORing each plaintext in the first set with the input of the differential distinguisher (*i.e.,* $A_0^0||B_0^0||C_0^0||D_0^0$). After that, we query the encryption oracle and compute the corresponding ciphertext difference. Here, we use 2^{m+1} plaintexts to generate 2^m plaintext/ciphertext pairs satisfying the input difference of our differential distinguisher (the value of m will be determined below).

Key Recovery. We first prepare $2^{7\times 32} = 2^{224}$ counters corresponding to the 2^{224} keys involved in the analysis. After that, for each ciphertext pair in 2^m pairs obtained in the data collection phase, we apply the following procedure:

1. Guess K_4 and partially decrypt the ciphertext to get the value and the difference at C_2^3. The average number of keys suggested by a pair after this step is 2^{32}.
2. Access the hash table H_1 to get, on average, 1 value of K_2 and D_0^3.
3. Guess K_6 and partially decrypt the ciphertext to get the value and the difference at A_3^3. The average number of keys suggested by a pair after this step will increase to 2^{64}.
4. Guess K_3 and partially decrypt the ciphertext combined with the value and the difference from the previous step to get the value and the difference at B_4^3. The average number of keys suggested by a pair after this step is 2^{96}.
5. Recall that $B_1^3 = B_4^3 \boxminus G_{21}(B_1^3 \boxplus C_1^3 \boxplus K_1) \oplus (3)_{32}$ and $C_1^3 = C_2^3 \boxplus G_{21}(B_1^3 \boxplus C_1^3 \boxplus K_1) \oplus (3)_{32}$. Hence $B_1^3 \boxplus C_1^3 = B_4^3 \boxplus C_2^3$. Therefore, by guessing K_1, we can deduce $G_{21}(B_1^3 \boxplus C_1^3 \boxplus K_1) = G_{21}(B_4^3 \boxplus C_2^3 \boxplus K_1)$ and then use the values obtained in steps 1 and 4 to compute the value and the difference at B_1^3 and C_1^3 and discard the key if the differences are not in the required form. This step filters out the suggested keys by 2^{16}. Thus, the average number of keys suggested by a pair after this step is 2^{112}.
6. Use the values and the differences form steps 3 and 5 to access the hash table H_3 and get, on average, 1 values of K_8 and A_0^3.
7. Access the hash table H_4 using the previously guessed value of K_6 in step 3 and the values and the differences from steps 5 and 7 to check if it is a valid

§ For simplicity, we store the binary values 0 and 1 as 32-bit words.

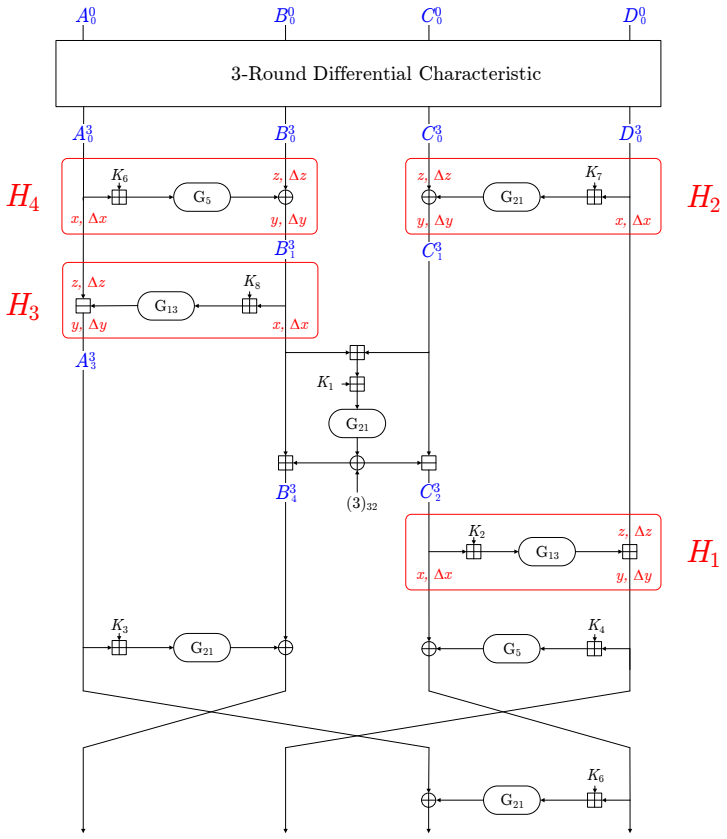

Fig. 4: $4\frac{1}{7}$-Round Attack on Bel-T-256

entry or not. This step will filter out the suggested keys by 2^{24}. Thus, the average number of keys suggested by a pair after this filtration will be 2^{88}.

8. Use the value from step 2 combined with the value and the difference from step 5 to access the hash table H_2 and get, on average, 2^8 value of K_7. Consequently, the average number of keys suggested by a pair after this procedure will be increased to 2^{96}. Thus, we increment the corresponding 2^{96} counters.

After repeating the above procedure for 2^m pairs, we select the key corresponding to the highest counter as a 224-bit right key. After that, we recover the 256-bit master key by testing the 224-bit right key along with the remaining 2^{32} values for K_5 using 2 plaintext/ciphertext pairs.

Table 9 summarizes the above steps, whereas the second column presents the average number of keys suggested by a pair after each step. The third and fourth

columns present the time complexity of each step in form of memory accesses and single S-box layer encryption in terms of m.

Table 9: Key recovery process of the attack on $4\frac{1}{7}$-round Bel-T-256

Step	# of suggested keys by a pair	Time Complexity	
		32-bit word memory Access	S-box layer Encryption
1	2^{32}	-	$2^m \times 2^{32} \times 2 = 2^{m+33}$
2	$2^{32} \times 1 = 2^{32}$	$2^m \times 2^{32} \times 2 = 2^{m+33}$	-
3	$2^{32} \times 2^{32} = 2^{64}$	-	$2^m \times 2^{64} \times 2 = 2^{m+65}$
4	$2^{64} \times 2^{32} = 2^{96}$	-	$2^m \times 2^{96} \times 2 = 2^{m+97}$
5	$2^{96} \times 2^{32} \times 2^{-16} = 2^{112}$	-	$2^m \times 2^{128} \times 2 = 2^{m+129}$
6	$2^{112} \times 1 = 2^{112}$	$2^m \times 2^{112} \times 2 = 2^{m+113}$	-
7	$2^{112} \times 2^{-24} = 2^{88}$	$2^m \times 2^{112} \times 1 = 2^{m+112}$	-
8	$2^{88} \times 2^8 = 2^{96}$	$2^m \times 2^{96} \times 1 = 2^{m+96}$	-

5.3 Attack Complexity and Success Probability

In this section, we present the complexity analysis of our attack in order to determine the required number of chosen plaintexts and the memory required to launch this attack. Also, we compute the success probability of the attack. Finally, we calculate its time complexity to compare our attack against the exhaustive search attack.

Data Complexity. For the differential attack to succeed with a high probability, we have to determine an appropriate value for the number of required plaintext/ciphertext pairs. To do so, we utilize the concept of signal-to-noise ratio (S/N) [4], which is calculated using the following formula:

$$S/N = \frac{2^k \times p}{\alpha \times \beta}$$

where k is the number of key bits involved in the analysis, p is the probability of the differential characteristic, α is the number of guessed keys by a pair, and β is the ratio of the pairs that are not discarded. In our analysis, $k = 224$, $p = 2^{-111}$, $\alpha = 2^{96}$ from table 9, and $\beta = 1$. Therefore, we have $S/N = \frac{2^{224} \times 2^{-111}}{2^{96} \times 1} = 2^{17}$. Due to this high S/N, we can use the recommendation of Biham and Shamir [4] that $3 \sim 4$ right pairs are sufficient enough to mount a successful differential attack. Therefore, we select the number of plaintext/ciphertext pairs

(2^m) equal to $4 \times p^{-1} = 2^{113}$. Consequently, the data complexity will be 2^{114} chosen plaintexts.

According to [20] and due to the high S/N, the success probability of the attack (P_s) can be calculated as $P_s \approx \Phi(\sqrt{p \times 2^m})$ where Φ is the cumulative distribution function of the standard normal distribution. Therefore, our differential attack will succeed with probability $P_s \approx 0.9772$.

Time Complexity. During the attack procedure, we make 32-bit word memory accesses in some steps and partially decrypt single S-box layers in other steps. Each S-box layer can be considered as a 32-bit big S-box with one or two modulo operations. Therefore, the time of single S-box layer will be slightly higher than the time of 32-bit word memory access. For simplicity, we assume that the time of 32-bit word memory access is the same as the time of a single S-box layer lookup which is roughly equal to $\frac{1}{7}$ of the time of one round encryption.

From Table 8, the time complexity of the pre-computation phase is dominated by the time required to construct the hash table H_1 which is equal to $\frac{1}{7} \times \frac{1}{4\frac{1}{7}} \times 2^{160} \approx 2^{155.14}$ $4\frac{1}{7}$-round encryptions. Similarly, from Table 9, the dominant part of the time complexity in the online phase comes from steps 5 which is $\frac{1}{7} \times \frac{1}{4\frac{1}{7}} \times (2^{m+129}) = 2^{m+124.14}$ $4\frac{1}{7}$-round encryptions. Therefore, the total time complexity of the online phase will be $2^{113+124.14} + 2 \times 2^{32} = 2^{237.14}$ $4\frac{1}{7}$-round encryptions.

Memory Complexity. The memory complexity of the pre-computation phase can be determined from Table 8 in which we need $2^{129} + 2^{96} + 2^{121} + 2^{120} \approx 2^{129}$ 32-bit word $= 2^{127}$ 128-bit blocks of memory. During the online phase, we have prepared 2^{224} counters corresponding to 2^{224} keys involved in the analysis. Since the upper limit of each counter depends on the number of plaintext/ciphertext pairs $(2^m = 2^{113})$, we can declare each counter as an unsigned 128-bit integer variable. Consequently, we need 2^{224} 128-bit blocks of memory in total.

6 Conclusion

In this paper, we studied the propagation of the XOR difference through modular addition. We showed that the independency assumption between two or more consecutive modular addition operations does not necessarily hold, and we constructed a more accurate MILP model for the differential trail through the modular addition taking into account the dependency between the consecutive modular additions. Then, we utilized the developed MILP model to automate the search process for the differential characteristics for Bel-T cipher. Up to the authors' knowledge, this is the best published theoretical attack against Bel-T-256 in the single-key setting.

References

1. Preliminary State Standard of Republic of Belarus (STBP 34.101.312011) (2011), http://apmi.bsu.by/assets/files/std/belt-spec27.pdf
2. Abdelkhalek, A., Sasaki, Y., Todo, Y., Tolba, M., Youssef, A.: MILP modeling for (large) s-boxes to optimize probability of differential characteristics. IACR Transactions on Symmetric Cryptology **2017**(4), 99–129 (Dec 2017)
3. Abdelkhalek, A., Tolba, M., Youssef, A.M.: Related-key Differential Attack on Round-Reduced Bel-T-256. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **101**(5), 859–862 (2018)
4. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer (1993)
5. Biryukov, A., Velichkov, V.: Automatic Search for Differential Trails in ARX Ciphers. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. LNCS, vol. 8366, pp. 227–250. Springer (2014)
6. Cui, T., Jia, K., Fu, K., Chen, S., Wang, M.: New Automatic Search Tool for Impossible Differentials and Zero-Correlation Linear Approximations. Cryptology ePrint Archive, Report 2016/689 (2016), https://eprint.iacr.org/2016/689
7. Daemen, J., Rijmen, V.: Probability distributions of Correlation and Differentials in Block Ciphers. Journal of Mathematical Cryptology JMC **1**(3), 221–242 (2007)
8. ElSheikh, M., Tolba, M., Youssef, A.M.: Integral Attacks on Round-Reduced Bel-T-256. In: Selected Areas in Cryptography – SAC 2018. LNCS, vol. 11349, pp. 73–91. Springer (2019)
9. Feistel, H., Notz, W.A., Smith, J.L.: Some cryptographic techniques for machine-to-machine data communications. Proceedings of the IEEE **63**(11), 1545–1554 (1975)
10. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In: Fast Software Encryption–FSE. LNCS, vol. 9783, pp. 268–288. Springer (2016)
11. Jovanovic, P., Polian, I.: Fault-based attacks on the Bel-T block cipher family. In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. pp. 601–604. EDA Consortium (2015)
12. Lai, X., Massey, J.L.: A Proposal for a New Block Encryption Standard. In: EUROCRYPT 1990. pp. 389–404. LNCS, Springer (1991)
13. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: Advances in Cryptology — EUROCRYPT '91. LNCS, vol. 547, pp. 17–38. Springer (1991)
14. Leurent, G.: Analysis of Differential Attacks in ARX Constructions. In: Advances in Cryptology – ASIACRYPT 2012. LNCS, vol. 7658, pp. 226–243. Springer (2012)
15. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Fast Software Encryption–FSE. LNCS, vol. 2355, pp. 336–350. Springer (2002)
16. McCluskey Jr, E.J.: Minimization of boolean functions. Bell system technical Journal **35**(6), 1417–1444 (1956)
17. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In: Information Security and Cryptology – Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer (2012)
18. Quine, W.V.O.: A way to simplify truth functions. The American Mathematical Monthly **62**(9), 627–631 (1955), http://www.jstor.org/stable/2307285
19. Sasaki, Y., Todo, Y.: New Impossible Differential Search Tool from Design and Cryptanalysis Aspects. In: Advances in Cryptology – EUROCRYPT 2017. LNCS, vol. 10212, pp. 185–215. Springer (2017)

20. Selçuk, A.A.: On Probability of Success in Linear and Differential Cryptanalysis. Journal of Cryptology **21**(1), 131–147 (Jan 2008)
21. Sun, L., Wang, W., Liu, R., Wang, M.: MILP-Aided Bit-Based Division Property for ARX-Based Block Cipher. Cryptology ePrint Archive, Report 2016/1101 (2016), https://eprint.iacr.org/2016/1101
22. Sun, L., Wang, W., Wang, M.: MILP-Aided Bit-Based Division Property for Primitives with Non-Bit-Permutation Linear Layers. Cryptology ePrint Archive, Report 2016/811 (2016), https://eprint.iacr.org/2016/811
23. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties (2014), https://eprint.iacr.org/2014/747
24. Wang, G., Keller, N., Dunkelman, O.: The Delicate Issues of Addition with Respect to XOR Differences. In: Selected Areas in Cryptography–SAC. LNCS, vol. 4876, pp. 212–231. Springer (2007)
25. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In: Advances in Cryptology – ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer (2016)