Integral Attacks on Round-Reduced Bel-T-256

Muhammad ElSheikh, Mohamed Tolba, and Amr M. Youssef(✉)

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

Abstract. Bel-T is the national block cipher encryption standard of the Republic of Belarus. It has a 128-bit block size and a variable key length of 128, 192 or 256 bits. Bel-T combines a Feistel network with a Lai-Massey scheme to build a complex round function with 7 S-box layers per round then iterate this round function 8 times to construct the whole cipher. In this paper, we present integral attacks against Bel-T-256 using the propagation of the bit-based division property. Firstly, we propose two 2-round integral characteristics by employing a Mixed Integer Linear Programming (MILP) (Our open source code to generate the MILP model can be downloaded from https://github.com/mhgharieb/Bel-T-256) approach to propagate the division property through the round function. Then, we utilize these integral characteristics to attack $3\frac{2}{7}$ rounds (out of 8) Bel-T-256 with data and time complexities of 2^{13} chosen plaintexts and $2^{199.33}$ encryption operations, respectively. We also present an attack against $3\frac{6}{7}$ rounds with data and time complexities of 2^{33} chosen plaintexts and $2^{254.61}$ encryption operations, respectively. To the best of our knowledge, these attacks are the first published theoretical attacks against the cipher in the single-key model.

Keywords: Bel-T · Integral attacks · Bit-based division property MILP

1 Introduction

In 2011, the Republic of Belarus, formerly known by its Russian name Byelorussia, has approved the Bel-T block cipher family as the state standard cryptographic encryption algorithm [1]. The Bel-T family consists of three block ciphers, denoted as Bel-T-k, with the same block size of 128 bits and key length $k = 128$, 192 or 256 bits. Bel-T merges a Lai-Massey scheme [8] with a Feistel network [5]. To the authors' knowledge, there are only two published cryptanalysis results on Bel-T's; fault-based attacks are considered in [6], and related-key differential attack on round-reduced Bel-T-256 are presented in [2]. In this paper, we present the first published single-key attack against Bel-T-256. Table 1 contrasts the result of our attacks with the related-key differential attack in [2].

C. Cid and M. J. Jacobson, Jr. (Eds.): SAC 2018, LNCS 11349, pp. 73–91, 2019.
https://doi.org/10.1007/978-3-030-10970-7_4

Table 1. Attack results on Bel-T-256

Model	Attack	#Round	Data	Time	Reference
Related key	Differential	$5\frac{6}{7}$	$2^{123.28}$	$2^{228.4}$	[2]
Single key	Integral	$3\frac{2}{7}$	2^{13}	$2^{199.33}$	Sect. 3.3
		$3\frac{6}{7}$	2^{33}	$2^{254.61}$	Sect. 3.4

Integral Attacks. In [4], Daemen *et al.* proposed a new cryptanalysis technique to analyze the security of the block cipher SQUARE. Subsequently, Knudsen and Wagner [7] formalized this technique and called it *integral attack*. The integral attack is a chosen-plaintext attack where the set of plaintext used in the attack is chosen to have XOR sum of 0. Firstly, the cryptanalyst constructs a multiset of plaintext such that it has a constant value at some bits while the other bits vary through all possible values. After that, the cryptanalyst calculates the XOR sum of all bits (or some of them) on the corresponding ciphertext after r rounds. If it is always 0 irrespective of the used secret key, we conclude that the cipher under test has an integral distinguisher.

The major techniques used to construct an integral characteristic include estimating the algebraic degree of the nonlinear parts of the cipher, and evaluating the propagation characteristic of the following integral properties [7]: ALL ($\mathcal{A}$) where every member appears the same number in the multiset; BALANCE ($\mathcal{B}$) where the XOR sum of all members in the multiset is 0; CONSTANT ($\mathcal{C}$) where the value is fixed to a constant for all members in the multiset; and UNKNOWN ($\mathcal{U}$) where the multiset is indistinguishable from one of n-bit random values.

Recently, Todo and Morii [16] proposed a generalization of the integral property called *bit-based integral property*. Unfortunately, the searching algorithm which they proposed to construct the integral distinguisher is restricted to ciphers whose block size is less than 32 bits due to its exponential time and memory complexities. To overcome this problem, Xiang *et al.* [17] proposed systematic rules to easily search for such integral distinguishers by employing a Mixed Integer Linear Programming (MILP) approach.

The rest of this paper is organized as follows. In Sect. 2, we briefly revisit the bit-based division property and summarize how to present its propagation through the basic cipher operations with MILP models. We also describe our approach to model the modular subtraction operation. In Sect. 3, we investigate the security of Bel-T block cipher against the integral attacks utilizing this MILP approach, finally, the conclusion is presented in Sect. 4.

2 Bit-Based Division Property

The division property, introduced by Todo [14], is a generalization of the integral property to utilize the hidden relations between the traditional $\mathcal{A}$ and $\mathcal{B}$ properties by exploiting the algebraic degree of the nonlinear components of the block cipher. Later, Todo in [15] proposed the first theoretical attack against the

full round MISTY1 based on a 6-round integral distinguisher. To construct this distinguisher, Todo utilized an improved version of the division property after analyzing the Algebraic Normal Form (ANF) of the S-boxes.

Recently, Todo and Morii [16] proposed a special case of the division property, called *bit-based division property*, in which each bit is traced independently. The bit-based division property allows us to exploit both of the algebraic degree and the details of the round function's structure. The bit-based division property is defined as follows:

Definition 1 (Bit-based Division Property [14]). *Let $\mathbb{X}$ be a multiset whose elements take a value of $\mathbb{F}_2^n$. When the multiset $\mathbb{X}$ has the division property $\mathcal{D}_{\mathbb{K}}^{1^n}$, where $\mathbb{K}$ denotes a set of n-dimensional vectors whose i-th element takes 0 or 1, it fulfills the following conditions:*

$$\bigoplus_{x \in \mathbb{X}} \boldsymbol{x}^{\boldsymbol{u}} = \begin{cases} \textit{unknown} & \textit{if there exists } \boldsymbol{k} \in \mathbb{K} \textit{ s.t. } \boldsymbol{u} \succeq \boldsymbol{k}, \\ 0 & \textit{otherwise.} \end{cases}$$

where $\boldsymbol{x}^{\boldsymbol{u}} = \prod_{i=1}^{n} x[i]^{u[i]}$, $\boldsymbol{u} \succeq \boldsymbol{k}$ if $u[i] \geq k[i]$ $\forall i$, and $x[i]$, $u[i]$ are the i-th bits of $\boldsymbol{x}$ and $\boldsymbol{u}$, respectively.

In the following, we present some propagation rules of the division property and show how to utilize MILP for automating the search for integral distinguishers based on the bit-based division property.

2.1 MILP Modeling for Propagation Rules of the Bit-Based Division Property

The advantage of the bit-based division property, over the traditional one, is its ability to exploit both the algebraic degree and the details of the round function structure by tracing each bit independently. The technique presented in [16] to find such distinguishers, however, is restricted to primitives whose block sizes are less than 32 bits due to its time and memory complexities. As mentioned above, to overcome this limitation, Xiang *et al.* [17] defined a new notation called *Division Trail*. With the division trail, it becomes easy to employ MILP for constructing the integral distinguisher. Later, Sun *et al.* complemented this work by handling ARX-based ciphers (`modulo` operations) [10] and ciphers with non-bit-permutation linear layers [11].

In the following subsection, we briefly describe how to model the division trail through several operations using MILP constraints. We firstly start by introducing the notation of a division trail.

Definition 2 (Division Trail [17]). *Let f_r denote the round function of an iterated block cipher. Assume that the input multiset to the block cipher has the initial division property $\mathcal{D}_{\{\boldsymbol{k}\}}^{1^n}$, and denote the division property after i-round propagation through f_r by $\mathcal{D}_{\mathbb{K}_i}^{1^n}$. Thus, we have the following chain of division property propagations:*

$$\{\boldsymbol{k}\} \stackrel{\text{def}}{=} \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} \cdots \xrightarrow{f_r} \mathbb{K}_r.$$

Moreover, for any vector $\boldsymbol{k}_i^* \in \mathbb{K}_i (i \geq 1)$*, there must exist a vector* $\boldsymbol{k}_{i-1}^* \in \mathbb{K}_{i-1}$ *such that* $\boldsymbol{k}_{i-1}^*$ *can propagate to* $\boldsymbol{k}_i^*$ *by the division property propagation rules. Furthermore, for* $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \cdots \times \mathbb{K}_r$*, if* $\boldsymbol{k}_{i-1}$ *can propagate to* $\boldsymbol{k}_i$ *for all* $i \in \{1, 2, \ldots, r\}$*, we call* $(\boldsymbol{k}_0, \boldsymbol{k}_1, \ldots, \boldsymbol{k}_r)$ *an r-round division trail. Thus, the set of the last vectors of all r-round division trails which start with* $\{\boldsymbol{k}\}$ *is equal to* $\mathbb{K}_r$*. Then, the i-th bit of r-round ciphertext is balanced if* e_i *(a unit vector whose i-th element is 1) does not exist in* $\mathbb{K}_r$*.*

The propagation rules of the bit-based division property through basic operations in block ciphers can be found in [15]. In here, we only summarize the MILP models associated with such rules.

Model for COPY [11]**.** Let $(a) \xrightarrow{COPY} (b_1, b_2, \ldots, b_m)$ denote the division trail through COPY function, where one bit is copied to m bits. Then, it can be described using the following MILP constraints:

$$\begin{cases} a - b_1 - b_2 - \cdots - b_m = 0, \\ a, b_1, b_2, \ldots, b_m \text{ are binary variables} \end{cases}$$

Model for XOR [11]**.** Let $(a_1, a_2, \ldots, a_m) \xrightarrow{XOR} (b)$ denote the division trail through an XOR function, where m bits are compressed to one bit using an XOR operation. Then, it can be described using the following MILP constraints:

$$\begin{cases} a_1 + a_2 + \cdots + a_m - b = 0, \\ a_1, a_2, \ldots, a_m, b \text{ are binary variables} \end{cases}$$

Model for AND [17]**.** Let $(a_0, a_1) \xrightarrow{AND} (b)$ denote the division trail though an AND function, where two bits are compressed using an AND operation. Then, it can be described using the following MILP constraints:

$$\begin{cases} b - a_0 \geq 0, \\ b - a_1 \geq 0, \\ a_0, a_1, b \text{ are binary variables} \end{cases}$$

MILP Model for S-Boxes. The original version of the bit-based division introduced in [16] is limited to bit-oriented ciphers and cannot be applied to ciphers with S-boxes. Xiang *et al.* overcome this problem by representing the S-Box using its algebraic normal form (ANF) (Algorithm 2 in [17]), also see [9].

The division trail though an n-bit S-box can be represented as a set of $2n$-dimensional binary vectors $\in \{0,1\}^{2n}$ which has a convex hull. The H-Representation of this convex hull can be computed using readily available functions such as `inequality_generator()` function in Sage[1] which returns a set of linear inequalities that describe these vectors. We use this set of inequalities as MILP constraints to present the division trail though the S-box.

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

MILP Model for Modular Addition. In [10], Sun *et al.* proposed a systematic method to deduce an MILP model for the modular addition operation of 4-bit variables by expressing the operation at the bit-level. Then this method is generalized for n-bit variables in [12].

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

$$z_{n-1} = x_{n-1} \oplus y_{n-1} \oplus c_{n-1},\ c_{n-1} = 0,$$
$$z_i = x_i \oplus y_i \oplus c_i,\ c_i = x_{i+1}y_{i+1} \oplus (x_{i+1} \oplus y_{i+1})c_{i+1},\quad i = n-2, n-3, \ldots, 0.$$

Consequently, the division trail through the modular addition can be deduced in terms of COPY, AND, and XOR operations [12].

MILP Model for Modular Addition with a Constant. In [10], Sun *et al.* explain how to deduce an MILP model for the modular addition of a 4-bit variable with a constant. The authors expressed the operation at the bit-level and exploited that the operations of XOR/AND with a constant do not influence the division property [10]. We can generalize this method for n-bit variables as follows. Let $(a_0, a_1, \ldots, a_{n-1}) \rightarrow (d_0, d_1, \ldots, d_{n-1})$ denote the division trail through n-bit modular addition with a constant, the division property propagation can be decomposed as COPY, AND, and XOR operations as follows:

$$\begin{cases} (a_{n-1}) \xrightarrow{COPY} (d_{n-1}, f_0, g_0) \\ (a_{n-2}) \xrightarrow{COPY} (a_{n-2,0}, a_{n-2,1}, a_{n-2,2}) \\ (a_{n-2,0}, f_0) \xrightarrow{XOR} (d_{n-2}) \\ (a_{n-2,1}, g_0) \xrightarrow{AND} (e_0) \\ (a_{n-2,2}, e_0) \xrightarrow{XOR} (v_0) \\ \left.\begin{array}{l} (v_{i-1}) \xrightarrow{COPY} (f_i, g_i) \\ (a_{n-2-i}) \xrightarrow{COPY} (a_{n-2-i,0}, a_{n-2-i,1}, a_{n-2-i,2}) \\ (a_{n-2-i,0}, f_i) \xrightarrow{XOR} (d_{n-2-i}) \\ (a_{n-2-i,1}, g_i) \xrightarrow{AND} (e_i) \\ (a_{n-2-i,2}, e_i) \xrightarrow{XOR} (v_i) \end{array}\right\} \textit{iterated for } i = 1, ..., n-3 \\ (a_0, v_{n-3}) \xrightarrow{XOR} (d_0) \end{cases}$$

where the intermediate variables $a_{i,0}$, $a_{i,1}$, $a_{i,2}$, f_i, g_i, e_i, and v_i are as shown in Table 2.

MILP Model for Modular Subtraction. In this section, we present an approach to deduce an MILP model for the modular subtraction operation using the same methodology used for Modular Addition. For consistency, we use the same notation as in [10].

[2] Big-endian representation.

Table 2. The intermediate variables for modular addition with a constant

$\underbrace{z_{n-1}}_{d_{n-1}}$	$\underbrace{x_{n-1}}_{a_{n-1}}$		
$\underbrace{z_{n-2}}_{d_{n-2}}$	$\underbrace{x_{n-2}}_{a_{n-2,0}} \oplus \underbrace{c_{n-2}}_{f_0}$	c_{n-2}	x_{n-1}
$\underbrace{z_{n-3}}_{d_{n-3}}$	$\underbrace{x_{n-3}}_{a_{n-3,0}} \oplus \underbrace{c_{n-3}}_{f_1}$	$\underbrace{c_{n-3}}_{v_0}$	$\underbrace{x_{n-2}}_{a_{n-2,2}} \oplus \overbrace{\underbrace{x_{n-2}}_{a_{n-2,1}}\ \underbrace{c_{n-2}}_{g_0}}^{e_0}$
$\underbrace{z_{n-4}}_{d_{n-4}}$	$\underbrace{x_{n-4}}_{a_{n-4,0}} \oplus \underbrace{c_{n-4}}_{f_2}$	$\underbrace{c_{n-4}}_{v_1}$	$\underbrace{x_{n-3}}_{a_{n-3,2}} \oplus \overbrace{\underbrace{x_{n-3}}_{a_{n-3,1}}\ \underbrace{c_{n-3}}_{g_1}}^{e_1}$
	$\cdots$	$\cdots$	
$\underbrace{z_1}_{d_1}$	$\underbrace{x_1}_{a_{1,1}} \oplus \underbrace{c_1}_{f_{n-3}}$	$\underbrace{c_1}_{v_{n-4}}$	$\underbrace{x_2}_{a_{2,2}} \oplus \overbrace{\underbrace{x_2}_{a_{2,1}}\ \underbrace{c_2}_{g_{n-4}}}^{e_{n-4}}$
$\underbrace{z_0}_{d_0}$	$\underbrace{x_0}_{a_0} \oplus c_0$	$\underbrace{c_0}_{v_{n-3}}$	$\underbrace{x_1}_{a_{1,2}} \oplus \overbrace{\underbrace{x_1}_{a_{1,1}}\ \underbrace{c_1}_{g_{n-3}}}^{e_{n-3}}$

Let $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ be n-bit vectors where $\boldsymbol{z} = \boldsymbol{x} \boxminus \boldsymbol{y}$. This relation can be rewritten as $\boldsymbol{z} = \boldsymbol{x} \boxplus$ (2's complement of $\boldsymbol{y}$) $= \boldsymbol{x} \boxplus (\bar{\boldsymbol{y}} \boxplus 1)$, where $\bar{\boldsymbol{y}}$ is the 1's complement of $\boldsymbol{y}$. Therefore, the division trail through the modular subtraction can be modelled as a division trail through a modular addition followed by a modular addition with a constant. This representation has two issues. The first issue is that two operations are used to present one operation which requires the use of more MILP constraints and variables, and consequently slowing down the search process. The second issue is that the information about the value of the constant, which is 1, in the modular addition with a constant is not utilized. This may lead the search process to conclude that some bits are not balanced even that they are balanced, as we show in Appendix A. Instead, at the bit level implementation, the modular subtraction operation is handled as a modular addition operation with two modifications: the first carry to the modular addition will be 1 instead of 0 ($c_{n-1} = 1$), and the second input to the modular addition will be the 1's complement of the second operand ($\bar{\boldsymbol{y}}$).

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

$$z_{n-1} = x_{n-1} \oplus \bar{y}_{n-1} \oplus c_{n-1},\ c_{n-1} = 1,$$
$$z_i = x_i \oplus \bar{y}_i \oplus c_i,\ c_i = x_{i+1}\bar{y}_{i+1} \oplus (x_{i+1} \oplus \bar{y}_{i+1})c_{i+1},\quad \forall i = n-2, n-3, \ldots, 0.$$
$$\textit{where } \bar{y}_i = y_i \oplus 1$$

The operation of XOR/AND with a constant does not influence the division property [10]. Therefore, the division property of $\bar{\boldsymbol{y}}$ is the same of $\boldsymbol{y}$. Consequently, we can generalize the modular subtraction operation for n-bit variables as follows:

Proposition 1. *Let* $((a_0, a_1, \ldots, a_{n-1}), (b_0, b_1, \ldots, b_{n-1})) \rightarrow (d_0, d_1, \ldots, d_{n-1})$ *be a division trail through n-bit modular subtraction operation. The division property propagation can be decomposed as COPY, AND, and XOR operations as follows:*

$$
\begin{cases}
(a_{n-1}) \xrightarrow{COPY} (a_{n-1,0}, a_{n-1,1}, a_{n-1,2}) \\
(b_{n-1}) \xrightarrow{COPY} (b_{n-1,0}, b_{n-1,1}, b_{n-1,2}) \\
(a_{n-1,0}, b_{n-1,0}) \xrightarrow{XOR} (d_{n-1}) \\
(a_{n-1,2}, b_{n-1,2}) \xrightarrow{XOR} (t_0) \\
(a_{n-1,1}, b_{n-1,1}) \xrightarrow{AND} (t_1) \\
(t_0, t_1) \xrightarrow{XOR} (v_0) \\
(v_0) \xrightarrow{COPY} (g_0, r_0) \\
(a_{n-2}) \xrightarrow{COPY} (a_{n-2,0}, a_{n-2,1}, a_{n-2,2}) \\
(b_{n-2}) \xrightarrow{COPY} (b_{n-2,0}, b_{n-2,1}, b_{n-2,2}) \\
\left.\begin{array}{l}
(a_{n-i,0}, b_{n-i,0}, g_{i-2}) \xrightarrow{XOR} (d_{n-i}) \\
(a_{n-i,1}, b_{n-i,1}) \xrightarrow{AND} (v_{i-1}) \\
(a_{n-i,2}, b_{n-i,2}) \xrightarrow{XOR} (m_{i-2}) \\
(m_{i-2}, r_{i-2}) \xrightarrow{AND} (q_{i-2}) \\
(v_{i-1}, q_{i-2}) \xrightarrow{XOR} (w_{i-2}) \\
(w_{i-2}) \xrightarrow{COPY} (g_{i-1}, r_{i-1}) \\
(a_{n-i-1}) \xrightarrow{COPY} (a_{n-i-1,0}, a_{n-i-1,1}, a_{n-i-1,2}) \\
(b_{n-i-1}) \xrightarrow{COPY} (b_{n-i-1,0}, b_{n-i-1,1}, b_{n-i-1,2})
\end{array}\right\} \textit{iterated for } i = 2, ..., n-2 \\
(a_{1,0}, b_{1,0}, g_{n-3}) \xrightarrow{XOR} (d_1) \\
(a_{1,1}, b_{1,1}) \xrightarrow{AND} (v_{n-2}) \\
(a_{1,2}, b_{1,2}) \xrightarrow{XOR} (m_{n-3}) \\
(m_{n-3}, r_{n-3}) \xrightarrow{AND} (q_{n-3}) \\
(v_{n-2}, q_{n-3}) \xrightarrow{XOR} (w_{n-3}) \\
(a_0, b_0, w_{n-3}) \xrightarrow{XOR} (d_0)
\end{cases}
$$

where the intermediate variables $a_{i,0}$, $a_{i,1}$, $a_{i,2}$, t_0, t_1, v_i, g_i, r_i, m_i, q_i, and w_i are as shown in Table 3.

In Appendix A, we present the results of an experiment we performed on a toy cipher to validate the model of the modular subtraction and show the effect of the first carry.

Table 3. The intermediate variables for modular subtraction

$\underbrace{z_{n-1}}_{d_{n-1}} = \underbrace{x_{n-1}}_{a_{n-1,0}} \oplus \underbrace{\bar{y}_{n-1}}_{b_{n-1,0}} \oplus 1$	
$\underbrace{z_{n-2}}_{d_{n-2}} = \underbrace{x_{n-2}}_{a_{n-2,0}} \oplus \underbrace{\bar{y}_{n-2}}_{b_{n-2,0}} \oplus \overbrace{c_{n-2}}^{g_0}$	$\overbrace{c_{n-2}}^{v_0} = \overbrace{\underbrace{x_{n-1}}_{a_{n-1,1}} \underbrace{\bar{y}_{n-1}}_{b_{n-1,1}}}^{t_1} \oplus (\overbrace{\underbrace{x_{n-1}}_{a_{n-1,2}} \oplus \underbrace{\bar{y}_{n-1}}_{b_{n-1,2}}}^{t_0})$
$\underbrace{z_{n-3}}_{d_{n-3}} = \underbrace{x_{n-3}}_{a_{n-3,0}} \oplus \underbrace{\bar{y}_{n-3}}_{b_{n-3,0}} \oplus \overbrace{c_{n-3}}^{g_1}$	$\overbrace{c_{n-3}}^{w_0} = \overbrace{\underbrace{x_{n-2}}_{a_{n-2,1}} \underbrace{\bar{y}_{n-2}}_{b_{n-2,1}}}^{v_1} \oplus \overbrace{(\overbrace{\underbrace{x_{n-2}}_{a_{n-2,2}} \oplus \underbrace{\bar{y}_{n-2}}_{b_{n-2,2}}}^{m_0}) \oplus \overbrace{c_{n-2}}^{r_0}}^{q_0}$
$\underbrace{z_{n-4}}_{d_{n-4}} = \underbrace{x_{n-4}}_{a_{n-4,0}} \oplus \underbrace{\bar{y}_{n-4}}_{b_{n-4,0}} \oplus \overbrace{c_{n-4}}^{g_2}$	$\overbrace{c_{n-4}}^{w_1} = \overbrace{\underbrace{x_{n-3}}_{a_{n-3,1}} \underbrace{\bar{y}_{n-3}}_{b_{n-3,1}}}^{v_2} \oplus \overbrace{(\overbrace{\underbrace{x_{n-3}}_{a_{n-3,2}} \oplus \underbrace{\bar{y}_{n-3}}_{b_{n-3,2}}}^{m_1}) \oplus \overbrace{c_{n-3}}^{r_1}}^{q_1}$
...	...
$\underbrace{z_1}_{d_1} = \underbrace{x_1}_{a_{1,0}} \oplus \underbrace{\bar{y}_1}_{b_{1,0}} \oplus \overbrace{c_1}^{g_{n-3}}$	$\overbrace{c_1}^{w_{n-4}} = \overbrace{\underbrace{x_2}_{a_{2,1}} \underbrace{\bar{y}_2}_{b_{2,1}}}^{v_{n-3}} \oplus \overbrace{(\overbrace{\underbrace{x_2}_{a_{2,2}} \oplus \underbrace{\bar{y}_2}_{b_{2,2}}}^{m_{m-4}}) \oplus \overbrace{c_2}^{r_{n-4}}}^{q_{n-4}}$
$\underbrace{z_0}_{d_0} = \underbrace{x_0}_{a_0} \oplus \underbrace{\bar{y}_0}_{b_0} \oplus c_0$	$\overbrace{c_0}^{w_{n-3}} = \overbrace{\underbrace{x_1}_{a_{1,1}} \underbrace{\bar{y}_1}_{b_{1,1}}}^{v_{n-2}} \oplus \overbrace{(\overbrace{\underbrace{x_1}_{a_{1,2}} \oplus \underbrace{\bar{y}_1}_{b_{1,2}}}^{m_{m-3}}) \oplus \overbrace{c_1}^{r_{n-3}}}^{q_{n-3}}$

3 Integral Attack on Bel-T-256

In this Section, we investigate the security of the Bel-T block cipher against the integral attack based on the bit-based division property.

3.1 Bel-T Specification

The official Bel-T specification is available only in Russian and the only version of the specification available in English is the one provided in its fault-based attacks analysis [6]. Bel-T has a 128-bit block size and a variable key length of 128, 192 or 256 bits. The 128-bit plaintext is divided into 4 32-bit words, i.e., $P = A^0||B^0||C^0||D^0$. Then, the round function illustrated in Fig. 1, is repeated eight times for all versions of Bel-T. Three mappings G_5, G_{13} and G_{21}: $\{0,1\}^{32} \rightarrow \{0,1\}^{32}$ are used, where G_r maps a 32-bit word $u = u_1||u_2||u_3||u_4$, with $u_i \in \{0,1\}^8$, as follows: $G_r(u) = (H(u_1)||H(u_2)||H(u_3)||H(u_4)) \lll r$. Here, H is an 8-bit S-box and $\lll r$ stands for left shift rotation by r positions. The specification of the 8-bit S-box can be found in [6].

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 keys as shown in Table 4. 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 $K_1||\ldots||K_6$ 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 $K_1||\ldots||K_4$ 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 $K_8 := K_4$.

3.2 Integral Distinguishers of Bel-T

As shown in Fig. 1, the Bel-T round function includes 7 S-boxes, modular additions, modular additions with key and modular subtractions. We construct an MILP model for the bit-based division property through Bel-T as follows. Firstly, we generate the division trail of the S-box using Algorithm 2 in [17]. Then, we deduce the inequalities of the S-box using `inequality_generator()` function in Sage. In the case of the Bel-T S-box, the number of generated inequalities is 71736, which is very large set to be handled by any MILP optimizer. Therefore, we reduce this set using a Greedy Algorithm which is proposed by Sun *et al.* in [13]. The size of the reduced set of the S-box representation inequalities is 28 and can be found in Appendix B.

Then, we implement the MILP model for modular addition and deduce the model for subtraction. Finally, we use the Gurobi[3] optimizer to search for the longest integral distinguisher for Bel-T. Based on our implementation, we found several 2-round integral distinguishers. Our code that is used to generate the MILP model for Bel-T and to search for an integral distinguisher can be downloaded from github.[4]

In here, we present two such distinguishers which are chosen in order to minimize the attack data and time complexities.

$$\begin{aligned} IC1 &: ((\mathcal{C}_{0-31}), (\mathcal{C}_{0-31}), (\mathcal{C}_{0-17}||\mathcal{A}_{18-18}||\mathcal{C}_{19-31}), (\mathcal{A}_{0-7}||\mathcal{C}_{8-31})) \\ &\xrightarrow{2R} ((\mathcal{U}_{0-31}), (\mathcal{U}_{0-31}), (\mathcal{U}_{0-26}||\mathcal{B}_{27-31}), (\mathcal{U}_{0-31})) \\ IC2 &: ((\mathcal{C}_{0-31}), (\mathcal{C}_{0-31}), (\mathcal{C}_{0-10}||\mathcal{A}_{11-26}||\mathcal{C}_{27-31}), (\mathcal{A}_{0-15}||\mathcal{C}_{16-31})) \\ &\xrightarrow{2R} ((\mathcal{U}_{0-26}||\mathcal{B}_{27-31}), (\mathcal{U}_{0-31}), (\mathcal{B}_{0-31}), (\mathcal{U}_{0-31})) \end{aligned}$$

where $\mathcal{C}_{i-j}/\mathcal{A}_{i-j}/\mathcal{B}_{i-j}/\mathcal{U}_{i-j}$ denote CONSTANT/ALL/BALANCE/UNKNOWN from bit number i to bit number j respectively counting from the most significant bit of the branch. Both of these integral distinguishers have been verified experimentally using a set of 256 randomly generated keys.

[3] http://www.gurobi.com/.
[4] https://github.com/mhgharieb/Bel-T-256.

Table 4. Encryption key schedule of Bel-T, where i and K_{7i-j} denote the round number and the round key, respectively.

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

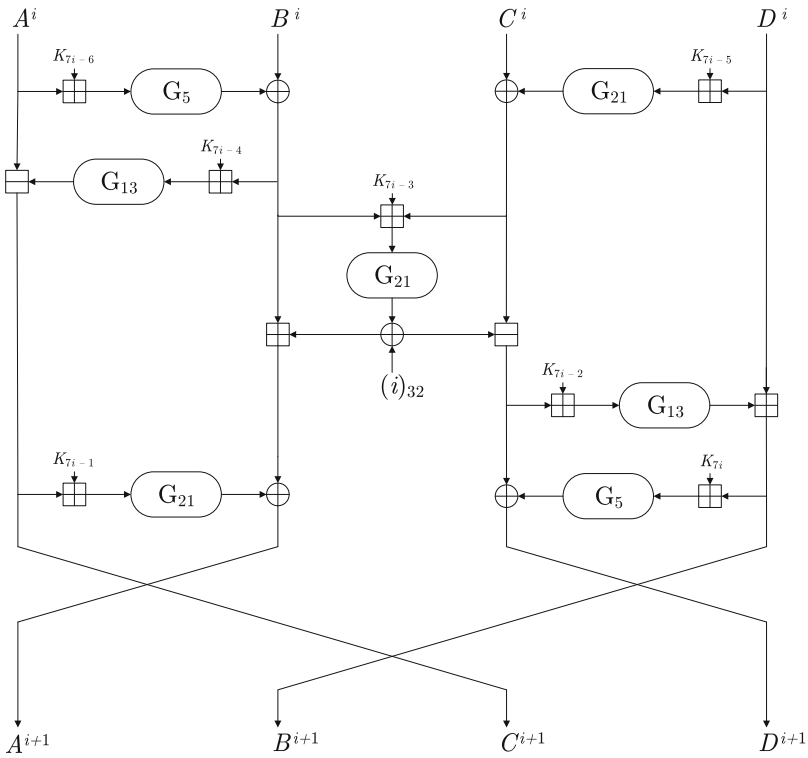

Fig. 1. Bel-T round function, where $\oplus, \boxplus, \boxminus$ denote bit-wise XOR, arithmetic addition and subtraction modulo 2^{32} respectively, and $(i)_{32}$ denotes the round number represented as 32-bit word.

3.3 Integral Cryptanalysis of $3\frac{2}{7}$-Round Bel-T-256

In this section, we present our Integral attack on $3\frac{2}{7}$-round Bel-T-256 by appending one round and two S-box layers on the above derived integral distinguisher $IC1$ as illustrated in Fig. 2.

Data Collection. We select m structures of plaintexts. In each structure, the 9 bits (bit number 18 in branch C^0 and bits 0-7 in branch D^0) vary through all 2^9 possible values and all other bits are fixed to an arbitrary constant value.

This ensures that each structure satisfies the required input division property of the integral distinguisher $IC1$. After that, we query the encryption oracle to obtain the corresponding ciphertexts. Subsequently, we apply the following key recovery procedure.

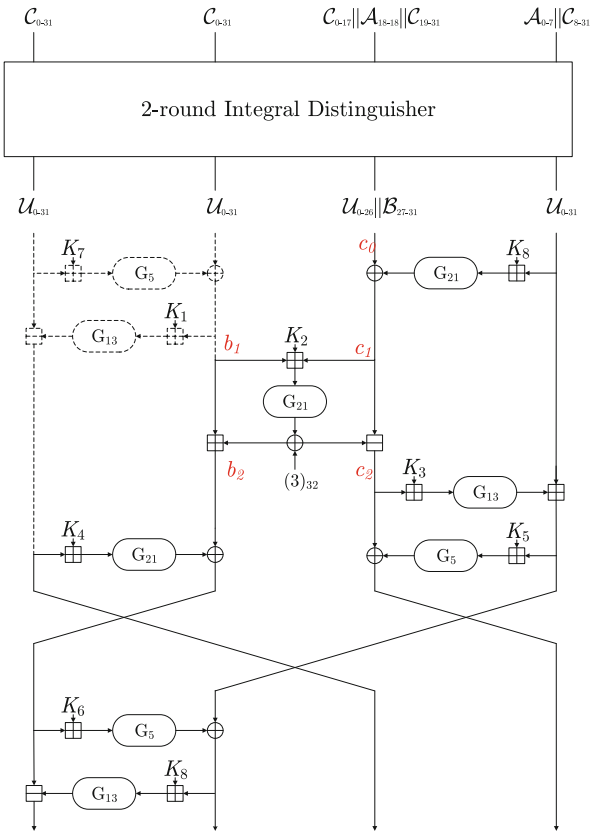

Fig. 2. $3\frac{2}{7}$-round attack on Bel-T-256

Key Recovery. For ciphertexts in each structure obtained in the data collection phase, we apply the following procedure:

1. Guess K_8 and K_4 and partially decrypt the ciphertext to obtain b_2.
2. Guess K_6 and K_5 and partially decrypt the ciphertext to obtain c_2.
3. Recall that $b_1 = b_2 - G_{21}(b_1 + c_1 + K_2) \oplus (3)_{32}$ and $c_1 = c_2 + G_{21}(b_1 + c_1 + K_2) \oplus (3)_{32}$. Hence $b_1 + c_1 = b_2 + c_2$. Therefore, by guessing K_2, we can deduce $G_{21}(b_1 + c_1 + K_2) = G_{21}(b_2 + c_2 + K_2)$ and then compute c_1 from b_2 and c_2.
4. Guess K_3 and use the previous guessed value of K_8 to compute c_0 from c_1 and c_2.
5. For each bit in the 5 least significant bits of the 32-bit word c_0, check that its XOR sum over the structure is zero. The probability that all these 5 bits are balanced is 2^{-5}. Therefore the probability that a key is survived after this test is also 2^{-5}. This means that the number of 192-bit key candidates passed this check is $2^{192} \times 2^{-5}$.

After repeating the above procedure for m structures, the number of surviving 192-bit key candidates will be $2^{192} \times (2^{-5})^m = 2^{192-5m}$. After that, we recover the 256-bit master key by testing the 2^{192-5m} 192-bit surviving key candidates along with the remaining 2^{64} values for K_1 and K_7 using 2 plaintext/ciphertext pairs.

Attack Complexity. The data complexity of the above attack is $m \times 2^9$ chosen plaintexts. The dominant part of time complexity is coming from deducing 192-bit key candidates after checking m structures. This part is equal to $\frac{7}{23} \times 2^9 \times 2^{192} \times [1 + 2^{-5} + (2^{-5})^2 + \cdots + (2^{-5})^{m-1}] = \frac{7}{23} \times 2^{201} \times \frac{1 - (2^{-5})^m}{1 - 2^{-5}}$. Additionally, the part due to exhaustively searching for the master key which is equal to $2 \times 2^{64} \times 2^{192-5m} = 2^{257-5m}$. To balance the attack between data and time complexities, we take $m = 16$. This means that the data complexity will be $16 \times 2^9 = 2^{13}$ chosen plaintexts and the time complexity will be $\frac{7}{23} \times 2^{201} \times \frac{1 - 2^{-80}}{1 - 2^{-5}} + 2^{177} \approx 2^{199.33}$ encryption operations.

It should be noted that other choices of m can lead to possible data and time trade-off. For example, if we set $m = 1$, the data complexity will be reduced to 2^9 chosen plaintexts at the expense of increasing the time complexity to 2^{252}.

3.4 Integral Cryptanalysis of $3\frac{6}{7}$-Round Bel-T-256

In this section, we present our integral attack on $3\frac{6}{7}$-round Bel-T-256 by appending one round and six S-box layers on the above derived integral distinguisher $IC2$, which is the only distinguisher makes the attack feasible, as illustrated in Fig. 3.

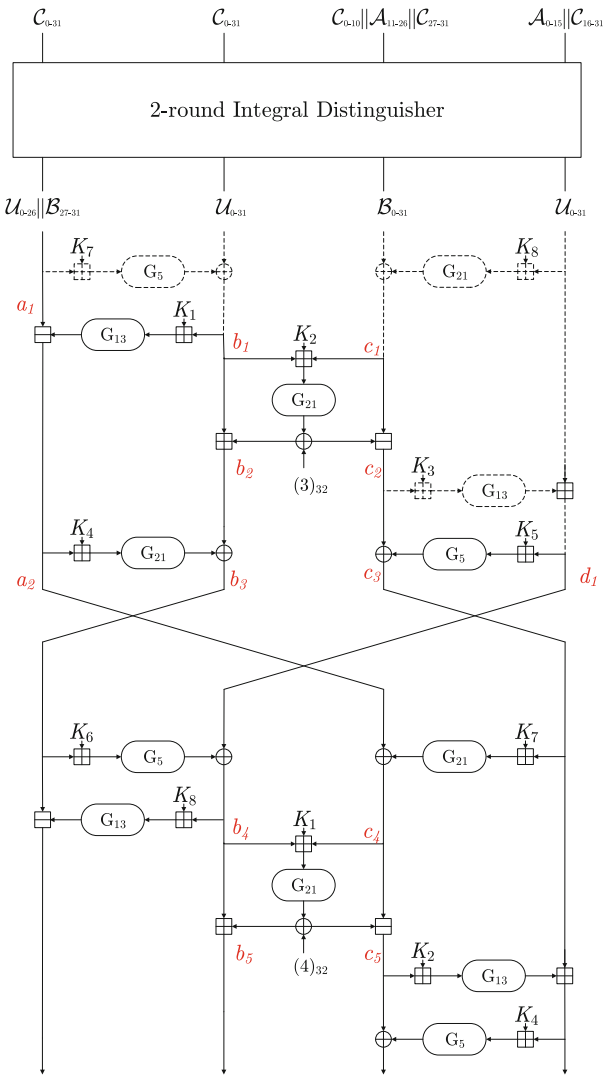

Fig. 3. $3\frac{6}{7}$-round attack on Bel-T-256

Data Collection. We select m structures of plaintexts. In each structure, the 32 bits (bits 11–26 in branch C^0 and bits 0–15 in branch D^0) vary through all 2^{32} possible values and all other bits are fixed to an arbitrary constant value. This ensures that each structure satisfies the required input division property of the integral distinguisher $IC2$. After that, we query the encryption oracle to obtain the corresponding ciphertexts. Subsequently, we apply the following key recovery procedure.

Key Recovery. For ciphertexts in each structure obtained in the data collection, we apply the following procedure:

1. Guess K_4 and partially decrypt the ciphertext to obtain c_5.
2. Recall that $b_4 = b_5 - G_{21}(b_4 + c_4 + K_1) \oplus (4)_{32}$ and $c_4 = c_5 + G_{21}(b_4 + c_4 + K_1) \oplus (4)_{32}$, hence $b_4 + c_4 = b_5 + c_5$. Therefore, by guessing K_1, we can deduce $G_{21}(b_4 + c_4 + K_1) = G_{21}(b_5 + c_5 + K_1)$ and then compute b_4 and c_4 from b_5 and c_5.
3. Guess K_2, K_6, K_7 and K_8 and deduce each 32-bit words a_2, b_3, c_3 and d_1.
4. Use the previous guessed value of K_4 to get the value of b_2 from a_2 and b_3.
5. Guess K_5 and get the value of c_2 from c_3 and d_1.
6. Recall that $b_1 = b_2 - G_{21}(b_1 + c_1 + K_2) \oplus (3)_{32}$ and $c_1 = c_2 + G_{21}(b_1 + c_1 + K_2) \oplus (3)_{32}$, hence $b_1 + c_1 = b_2 + c_2$. Therefore, by guessing K_2, we can deduce $G_{21}(b_1 + c_1 + K_2) = G_{21}(b_2 + c_2 + K_2)$ and then compute b_1 from b_2 and c_2.
7. Use the previous guessed value of K_1 to compute a_1 from a_2 and b_1.
8. For each bit in the 5 least significant bits of 32-bit word a_1, check that the XOR sum of it over the structure is zero. The probability that all these 5 bits are balanced is 2^{-5}. Therefore the probability that a key is survived after this test is also 2^{-5}. This means that the number of 224-bit key candidates passed this check is $2^{224} \times 2^{-5}$.

After repeating the above procedure for m structures, the number of surviving 224-bit key candidates will be $2^{224} \times (2^{-5})^m = 2^{224-5m}$. After that we recover the 256-bit master key by testing the 2^{224-5m} 192-bit surviving key candidates along with the remaining 2^{32} values for K_3 using 2 plaintext/ciphertext pairs.

Attack Complexity. The data complexity is $m \times 2^{32}$ chosen plaintexts. The dominant part of time complexity is coming from deducing 224-bit key candidates after checking m structure. This part is equal to $\frac{10}{27} \times 2^{32} \times 2^{224} \times [1 + 2^{-5} + (2^{-5})^2 + \cdots + (2^{-5})^{m-1}] = \frac{10}{27} \times 2^{256} \times \frac{1 - (2^{-5})^m}{1 - 2^{-5}}$. Additionally, the part due to exhaustively searching for the master key which is equal to $2 \times 2^{32} \times 2^{224-5m} = 2^{257-5m}$. To balance the attack between data and time complexities, we take $m = 2$. This means that the data complexity will be $2 \times 2^{32} = 2^{33}$ chosen plaintexts and the time complexity will be $\frac{10}{27} \times 2^{256} \times \frac{1 - 2^{-10}}{1 - 2^{-5}} + 2^{247} \approx 2^{254.61}$ encryption.

4 Conclusion

In this paper, we investigated the security of Bel-T-256 against integral attacks based on the bit-based division property. In particular, we have built a MILP model for the Bel-T round function to automate the search for integral distinguishers based on the bit-based division property. Using two of the obtained integral distinguishers, we presented attacks on $3\frac{2}{7}$ and $3\frac{6}{7}$ rounds of Bel-T-256 with data and time complexities of 2^{13}, 2^{33} chosen plaintexts and $2^{199.33}$, $2^{254.61}$ encryption operations, respectively.

A Validation of the MILP Model for the Division Trail Through a Modular Subtraction Operation

In this appendix, we provide the result of our experiments on a toy cipher in order to validate the MILP model for the division trail through a modular subtraction operation. Moreover, we show that the proposed model of the division trail through the modular subtraction at the bit-level ($\boldsymbol{z} = \boldsymbol{x} \boxminus \boldsymbol{y}$) gives better results than modelling it as a division trail through a modular addition followed by a modular addition with a constant ($\boldsymbol{z} = \boldsymbol{x} \boxplus \bar{\boldsymbol{y}} \boxplus 1$).

The round function of the toy cipher used during the experiments is a small version of the SPECK round function [3] with modular subtraction instead of modular addition as shown in Fig. 4 where the block size is 8 bits, (X_L^i, X_R^i) is the input of the i-th round, and k_i is the subkey used in the i-th round.

We follow the same approach used in [10] to validate their MILP model for modular addition. The experimental procedure is as follows:

1. For an initial division property, use our MILP model for the modular subtraction at the bit-level ($\boldsymbol{z} = \boldsymbol{x} \boxminus \boldsymbol{y}$) to find the set of balanced bits at the output of the toy cipher.
2. Use the other MILP model ($\boldsymbol{z} = \boldsymbol{x} \boxplus \bar{\boldsymbol{y}} \boxplus 1$) to find the balanced bits corresponding the same initial division property.
3. Exhaustively search for the balanced bits as follows:
   (a) Divide the space of the plaintexts (2^8 plaintexts) to a group of multi-sets of plaintexts. Each one of these multi-sets satisfies the initial division property.
   (b) Encrypt each multi-set of the plaintexts using a randomly chosen key and find the bits with zero-sum over all the corresponding ciphertexts of that multi-set, and then find the common zero-sum bits over all the multi-sets.
   (c) Repeat the previous step 2^{10} iterations and find the common zero-sum bits at the output of the toy cipher over all the iterations.

4. Compare the results from the previous three steps for the same initial division property.
5. Repeat the previous steps for all possible values of the initial division property and for a toy cipher consists of up to 6 rounds similar to the one in the Fig. 4.

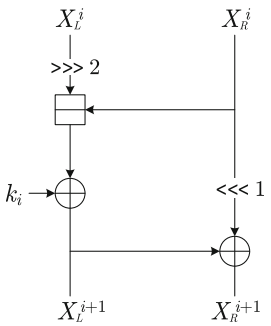

Fig. 4. The round function of the toy cipher.

Table 5. Comparison of zero-sum bits found by using three methods for the toy cipher, where #{Bits} is the number of balanced bits and 'Bits' is the position of these bits counted from the most significant bit.

Input Division property	Rounds	Exhaustive search		MILP-aided Bit-based Division property			
				$z = x \boxminus y$		$z = x \boxplus \bar{y} \boxplus 1$	
		#{Bits}	Bits	#{Bits}	Bits	#{Bits}	Bits
$\mathcal{D}^{1^8}_{\{[01111111]\}}$	1	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	2	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	3	6	$1 \sim 3, 5 \sim 7$	6	$1 \sim 3, 5 \sim 7$	4	$2 \sim 3, 6 \sim 7$
	4	1	3	1	3	0	-
	5	0	-	0	-	0	-
$\mathcal{D}^{1^8}_{\{[11111110]\}}$	1	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	2	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	3	6	$1 \sim 3, 5 \sim 7$	6	$1 \sim 3, 5 \sim 7$	6	$1 \sim 3, 5 \sim 7$
	4	3	$2 \sim 3, 6$	1	3	1	3
	5	0	-	0	-	0	-
$\mathcal{D}^{1^8}_{\{[00001111]\}}$	1	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	2	4	$2 \sim 3, 6 \sim 7$	4	$2 \sim 3, 6 \sim 7$	4	$2 \sim 3, 6 \sim 7$
	3	0	-	0	-	0	-
$\mathcal{D}^{1^8}_{\{[11110000]\}}$	1	8	$0 \sim 7$	8	$0 \sim 7$	8	$0 \sim 7$
	2	8	$0 \sim 7$	8	$0 \sim 7$	6	$1 \sim 3, 5 \sim 7$
	3	2	$3, 7$	2	$3, 7$	1	3
	4	0	-	0	-	0	-

From the result of the experiments, we can conclude that the balanced bits found by the MILP-aided bit-based division property are indeed balanced. Moreover, the MILP model for the division trail through the modular subtraction at the bit-level ($\boldsymbol{z} = \boldsymbol{x} \boxminus \boldsymbol{y}$) also uses less number of constraints and gives same or better results (in terms of number of the balanced bits) than modelling it as a division trail through a modular addition followed by a modular addition with a constant ($\boldsymbol{z} = \boldsymbol{x} \boxplus \bar{\boldsymbol{y}} \boxplus 1$). A sample of our results can be found in Table 5 and the mismatch between the two approaches for modelling the division trail through a modular subtractions is summarized in Table 6.

Table 6. Mismatch between the two approaches for modelling the division trail through a modular subtraction.

Rounds	Inputs Division property	MILP-aided Bit-based Division property			
		$z = x \boxminus y$		$z = x \boxplus \bar{y} \boxplus 1$	
		#{Bits}	Bits	#{Bits}	Bits
1	{[10000011]}, {[11000010]}, {[11000011]}	8	$0 \sim 7$	6	$1 \sim 3, 5 \sim 7$
2	{[01101101]}, {[01111001]}, {[10100101]}, {[10101100]}, {[10110001]}, {[10111000]}, {[11100100]}, {[11110000]}	8	$0 \sim 7$	6	$1 \sim 3, 5 \sim 7$
	{[10001111]}, {[10011011]}, {[11001110]}, {[11001111]}, {[11011010]}, {[11011011]}	6	$1 \sim 3, 5 \sim 7$	4	$2 \sim 3, 6 \sim 7$
	{[10000011]}, {[11000010]}	2	3,7	1	3
	{[11000011]}	4	$2 \sim 3, 6 \sim 7$	1	3
3	{[01110111]}, {[01111111]}, {[10110110]}, {[10110111]}, {[10111101]}, {[10111110]}, {[11110110]}, {[11111100]}	6	$1 \sim 3, 5 \sim 7$	4	$2 \sim 3, 6 \sim 7$
	{[01101101]}, {[01111001]}, {[10100101]}, {[10101100]}, {[10110001]}, {[10111000]}, {[11100100]}, {[11110000]}	2	3,7	1	3
	{[10001111]}, {[10011011]}, {[11001110]}, {[11001111]}, {[11011010]}, {[11011011]}	1	3	0	-
4	{[11111011]}	6	$1 \sim 3, 5 \sim 7$	4	$2 \sim 3, 6 \sim 7$
	{[01110111]}, {[01111111]}, {[10110110]}, {[10110111]}, {[10111101]}, {[10111110]}, {[11110110]}, {[11111100]}	1	3	0	-
5	{[11111011]}	1	3	0	-

B Division Trail Representation of Bel-T S-Box

$$
\begin{bmatrix}
1 & 1 & 1 & 36 & 1 & 1 & 1 & 1 & -6 & -6 & -6 & -6 & -6 & -6 & -6 & -6 & 5 \\
1 & 1 & 39 & 1 & 1 & 1 & 1 & 1 & -7 & -7 & -7 & -7 & -6 & -6 & -6 & -6 & 6 \\
2 & 68 & 2 & 2 & 2 & 2 & 2 & 2 & -11 & -11 & -12 & -12 & -12 & -11 & -12 & -11 & 10 \\
1 & 1 & 1 & 1 & 1 & 1 & 35 & 1 & -6 & -6 & -6 & -6 & -6 & -6 & -6 & -5 & 5 \\
14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -3 & -3 & -3 & -2 & -3 & -1 & 3 \\
0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & -2 & -2 & -1 & 0 & -2 & -1 & -2 & -1 & 2 \\
-6 & -2 & 0 & 0 & -5 & 2 & 2 & 1 & -7 & -6 & -6 & -6 & 30 & -5 & -7 & -5 & 20 \\
-8 & -12 & -6 & -11 & -8 & -6 & -13 & -2 & -6 & -10 & -5 & -5 & -12 & -2 & 32 & 4 & 70 \\
-1 & -2 & -2 & -3 & -2 & -2 & -2 & -4 & 16 & 16 & 17 & 17 & 17 & 14 & 17 & 15 & 0 \\
0 & -1 & -2 & 0 & 0 & 1 & 0 & 0 & -2 & 2 & 5 & -3 & -3 & -2 & -3 & 2 & 6 \\
1 & 1 & -1 & 2 & 0 & 2 & 0 & 2 & -1 & -6 & -7 & 19 & 1 & -7 & -7 & -7 & 8 \\
-9 & -1 & 0 & -1 & -1 & 3 & 3 & -2 & -7 & -7 & -6 & -7 & -2 & 29 & -5 & -8 & 21 \\
0 & -3 & -1 & -1 & -3 & -1 & 0 & 2 & -4 & 10 & -4 & -3 & -1 & -1 & 2 & -5 & 13 \\
-2 & -1 & -8 & -2 & -5 & -5 & -8 & -12 & 10 & -15 & 3 & 3 & -11 & 3 & 4 & -7 & 53 \\
-23 & -24 & -23 & -21 & -21 & -23 & -22 & -26 & 4 & 5 & 2 & 3 & 4 & -1 & 4 & 4 & 158 \\
-2 & -4 & 0 & -5 & -2 & -1 & -6 & -1 & -4 & -4 & -4 & -5 & -6 & 3 & 14 & 2 & 25 \\
-10 & 1 & 3 & 2 & 2 & 0 & 3 & 3 & -13 & -12 & -13 & 47 & 0 & -12 & -12 & -12 & 23 \\
-1 & -2 & 0 & -2 & -2 & -1 & 0 & -2 & 3 & 3 & 3 & 3 & 2 & 3 & 3 & 1 & 7 \\
-6 & -3 & -5 & -7 & -6 & -6 & -6 & -4 & 4 & 3 & 4 & 5 & 5 & 4 & 2 & 1 & 35 \\
-1 & -2 & -3 & -2 & 0 & -1 & -3 & -3 & 15 & 15 & 14 & 14 & 15 & 12 & 15 & 13 & 0 \\
-9 & 3 & -9 & -9 & 0 & 1 & -9 & -9 & -2 & -2 & 8 & -3 & -2 & 8 & -2 & -12 & 48 \\
-1 & -1 & -3 & 0 & 0 & -1 & -1 & -1 & -2 & 1 & 3 & -4 & -3 & -2 & -2 & 6 & 11 \\
-2 & -2 & 0 & -2 & -2 & 3 & -2 & -2 & -1 & -1 & -1 & -1 & 2 & -3 & -1 & 2 & 13 \\
-1 & -2 & -1 & 0 & -1 & -1 & -1 & 1 & -2 & -3 & 0 & 0 & -2 & 0 & 3 & 1 & 9 \\
0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -3 & 3 & 1 & -2 & -1 & 1 & 1 & -2 & 8 \\
-1 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 6 \\
-1 & -1 & 0 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 6 \\
-2 & -1 & -2 & -2 & -2 & -2 & 0 & -2 & 2 & 2 & 0 & 2 & 2 & 1 & 2 & 1 & 11
\end{bmatrix}
\begin{bmatrix}
a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ 1
\end{bmatrix}
\geq
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{bmatrix}
$$

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., Tolba, M., Youssef, A.M.: Related-key differential attack on round-reduced Bel-T-256. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **101**(5), 859–862 (2018)
3. Beaulieu, R., Treatman-Clark, S., Shors, D., Weeks, B., Smith, J., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE (2015)
4. Daemen, J., Knudsen, L., Rijmen, V.: The block cipher square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0052343
5. Feistel, H., Notz, W.A., Smith, J.L.: Some cryptographic techniques for machine-to-machine data communications. Proc. IEEE **63**(11), 1545–1554 (1975)
6. 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)

7. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45661-9_9
8. Lai, X., Massey, J.L.: A proposal for a new block encryption standard. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46877-3_35
9. Sun, L., Wang, M.: Toward a further understanding of bit-based division property. Sci. China Inf. Sci. **60**(12), 128101 (2017)
10. 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
11. 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
12. Sun, L., Wang, W., Wang, M.: Automatic search of bit-based division property for ARX ciphers and word-based division property. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 128–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_5
13. Sun, S., et al.: 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
14. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_12
15. Todo, Y.: Integral cryptanalysis on full MISTY1. J. Cryptol. **30**(3), 920–959 (2017)
16. Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-52993-5_18
17. 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: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_24