Optimal Lower Bounds on the Number of Queries for Solving Differential Equations of Addition

Souradyuti Paul and Bart Preneel

Katholieke Universiteit Leuven, Dept. ESAT/COSIC,
Kasteelpark Arenberg 10,
B–3001 Leuven-Heverlee, Belgium
{Souradyuti.Paul, Bart.Preneel}@esat.kuleuven.ac.be

Abstract. Equations that mix addition modulo 2^n $(+)$ and exclusive-or $(\oplus)$ have a host of applications in design and cryptanalysis of symmetric ciphers. In this paper we study two basic equations of the form $(x+y) \oplus (x+(y \oplus \beta)) = \gamma$ and $(x+y) \oplus ((x \oplus \alpha)+(y \oplus \beta)) = \gamma$, which are termed *differential equations of addition*. Firstly, the paper presents formal proofs for the number of solutions for (x, y) in the above equations. Secondly, we give an algorithm that solves the first equation with $(n-t-1)$ queries in the worst case, where n is the input size and t is a non-negative parameter depending on the input, when the previous best known algorithm required $3(n-1)$ queries. For the other equation, the number of queries required by our algorithm is 3 in the worst case for all $n > 2$, i.e., the number of queries is constant asymptotically. The most important contribution of the paper is that, using simple combinatorial relations among carry bits and input bits, we also show that, for our algorithms, the upper bounds on the required number of queries match worst case lower bounds. This, in effect, closes further research in this direction as our lower bounds are *optimal*. Finally, as an example of practical use of our results, we show that these results alone improve the data complexity of a differential attack on the Helix stream cipher by a factor of 3 in the worst case and by a factor of 46.5 in the best case.

Keywords: Input and output differentials, Lower bound, Upper bound, Optimal bound, Asymptotic Complexity, Query.

1 Introduction

The *arithmetic addition of two n-bit integers modulo* 2^n is a nonlinear transformation when considered over $GF(2)$. Equations that mix addition with other Boolean operations such as *exclusive-or* $(\oplus)$, *or* $(\vee)$ and/or *and* $(\wedge)$ are interesting research subjects in their own right. However, many cryptographic primitives, such as Helix [4], IDEA [9], Mars [7], RC6 [8], and Twofish [10] mix modular addition with exclusive-or operations

to achieve nonlinearity through the propagation of the carry-bits. The list is by no means exhaustive as the equations of the above types have applications outside the scope of cryptography too (e.g. optimization of circuit complexities). However, in the present context, we will take a closer look at the combination of *addition modulo* 2^n and *bitwise exclusive-or* as it is extensively used as one of the basic building blocks to generate modern symmetric ciphers.

There is a large body of literature that studies equations involving *addition* from many different angles. Staffelbach and Meier investigated the probability distribution of the carry for integer addition [3]. Wallen explained the linear approximations of modular addition [14]. In the most recent development of stream ciphers, Klimov and Shamir also used an update function for internal state, known as T-function, where *modular addition* and *OR* are mixed in a certain fashion to achieve many useful properties of a secure stream cipher [12, 13].

Differential cryptanalysis, introduced by Biham and Shamir [1], is one of the most powerful attacks against symmetric ciphers. Immunity against differential cryptanalysis is a prime factor in the evaluation of the security of a cipher. The interplay between addition ($+$) and exclusive-or ($\oplus$) against differential cryptanalysis has been studied in depth by Lipmaa and Moriai [2]. In particular, the equation they investigated to determine the differential probabilities is $(x+y) \oplus ((x \oplus \alpha)+(y \oplus \beta)) = \gamma$, where x, y, α, β, $\gamma \in \mathbb{Z}_2^n$, α, β are the input differentials and γ is the output differential. They have shown that the probability of a triplet (α, β, γ) satisfying the above equation on a randomly chosen pair of n-bit integers (x, y) can be computed with an asymptotic time complexity of $O(\log n)$. Many other useful differential properties of addition (e.g. maximal differentials) can also be determined with the same asymptotic time complexity [2].

Another way of mixing *addition* and *exclusive-or* is to use the dual of the above case where differences are expressed using *addition modulo* 2^n, that is, employing equations of the form $(x \oplus y) + ((x+\alpha) \oplus (y+\beta)) = \gamma$. The differential probabilities of this case has been investigated in detail by Lipmaa *et al.* [6].

In this paper we explore two basic addition equations where differences of inputs and output are expressed in terms of *exclusive-or*. In particular, we study the following two equations separately:

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma \tag{1}$$

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma \tag{2}$$

with an objective to determine (x, y), which satisfies the equation for all triples (α, β, γ)[1], using a minimum number of queries (α, β) where an adversary is *only* allowed to supply (α, β) to an oracle and receive the corresponding γ ($\alpha = 0$ for Eqn. (1)). Note that Eqn. (2) has already been studied by Lipmaa and Moriai [2] to compute many differential properties. Our focus is on recovering secret information instead of calculating differential probabilities. These methods can be used to reduce the data complexity of adaptively chosen plaintext/ciphertext attacks that attempt to recover secret information of a cipher. As a direct example, we apply our results to the Helix cipher. We term the equations of the above types *differential equations of addition* to be consistent with the existing body of literature as the corresponding *differential probabilities* derived from such equations are known as *differential probabilities* of *addition* [2].

Our Main Contributions: The aim of this paper is fourfold. First, we determine the number of all solutions for Eqn. (1) and (2) in a general framework. A claim, on the number of solutions for Eqn. (1), has been made in [5], for a specific case of $n = 32$, without any *formal proof* which is non-trivial. Secondly, we show that a worst case lower bound on the required number of queries $(0, \beta)$ to solve Eqn. (1) for (x, y) is $(n-t-1)$ where $(n-t) > 1$ with t being the bit-position of the least significant '1' of x. A worst case lower bound on the number of queries (α, β) required to solve Eqn. (2) is 3 for $n > 2$. Most importantly, for solving the above equations we also design algorithms whose upper bounds on the number of queries match worst case lower bounds.

Our algorithm to solve Eqn. (1) records an improvement over the previous best known algorithm which required $3(n-1)$ queries [5] (note that our algorithm takes $(n-t-1)$ queries with $t \geq 0$). Furthermore, our results essentially close further investigation in this particular direction as the equations are solved with an *optimal* number of queries in the worst case. We directly use these results to reduce the data complexity of an attack on the Helix stream cipher by a factor of 3 in the worst case (a factor of 46.5 in the best case), without exploring any other possibilities for improvement [5]. In addition to that, our solution techniques, which make use of simple combinatorial relations among carry bits and input bits, leave open the possibility of solving more complex equations (e.g., combination of *addition*, *XOR*, *multiplication* and T-functions) efficiently and also computing differential probabilities of addition with improved

[1] The number of all possible triplets equals 2^n for Eqn. (1) and 2^{2n} for (2).

complexities.

2 The Problem and an Adversarial Model

The aim of an adversary is to solve the following equation for fixed unknown integers x and y,

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma \tag{3}$$

using triplets (α, β, γ). A pair (α, β) is defined to be a *query*. In Eqn. (3), $x, y, \alpha, \beta, \gamma \in \mathbb{Z}_2^n$ and the symbols '$+$', '$\oplus$', '$\wedge$' denote the binary operations *addition modulo* 2^n, *bit-wise exclusive-or* and *bit-wise and* on $\mathbb{Z}_2^n$ respectively. We will denote $x \wedge y$ by xy. Throughout the paper, unless otherwise stated, n denotes a positive integer.

2.1 The Power of the Adversary

The power of an adversary that solves Eqn. (3) is defined as follows.

1. An adversary has unrestricted computational power.
2. An adversary has infinite amount of memory.
3. An adversary can *only* make queries $(\alpha, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ to an oracle which computes γ using fixed $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ in Eqn. (3) and returns the value to the adversary. We will often refer to that fixed (x, y) as the *seed* of the oracle.

Such an oracle seeded with $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ (which is unknown to adversary) can be viewed as a mapping $O_{xy} : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ and defined by

$$\begin{aligned} O_{xy} = \{(\alpha, \beta, \gamma) \,|\, (\alpha, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n, \\ \gamma = (x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta))\} \,. \end{aligned} \tag{4}$$

An adversarial model, similar to the one described above for Eqn. (3), can be constructed for the following equation:

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma \,, \tag{5}$$

by setting $(\alpha, \beta) \in \{0\}^n \times \mathbb{Z}_2^n$ and the mapping $O_{xy} : \{0\}^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$.

The model described above represents a practical adaptively chosen message attack scenario where the adversary makes queries to an oracle adaptively. Based on the replies from the oracle, the adversary computes one or more unknown parameters.

2.2 The Problem

O_{xy}, defined in Eqn. (4), generates a family of mappings $\mathcal{F} = \{O_{xy} \,|\, (x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n\}$. Let a mapping $f : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathcal{F}$ be defined by

$$f(x,\, y) = O_{xy}\,. \tag{6}$$

In the adversarial framework described in Sect. 2.1, solving Eqn. (3) or (5) is understood to be solving the corresponding equation:

$$f(x,\, y) = D \in \mathcal{F} \quad \text{for } (x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n\,. \tag{7}$$

Let D-satisfiable denote the solution set for Eqn. (7). Therefore,

$$D\text{-satisfiable} = \{(x,\, y) \,|\, (x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n,\, f(x,\, y) = D\}\,. \tag{8}$$

The task of an adversary is to determine D-satisfiable when the maximum information she can extract from the oracle is the set D.

Example 1. (D-satisfiable) Suppose $n = 2$ and therefore, x, y, α, β, $\gamma \in \mathbb{Z}_2^2 \times \mathbb{Z}_2^2$. The oracle receives $(\alpha,\, \beta)$ and computes $\gamma = (x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta))$ and returns γ to the adversary. For example, let the oracle return $\gamma = (1,\, 0)$ for $(\alpha,\, \beta) = ((0,\, 0),\, (0,\, 1))$. There are at most 16 values of $(\alpha,\, \beta)$ (therefore, at most 16 *queries* an adversary can submit to the oracle) and for each $(\alpha,\, \beta)$ the oracle returns a γ. Now, the set D (as defined in Eqn. 7) contains all 16 triplets $(\alpha,\, \beta,\, \gamma)$. Therefore, the set D-satisfiable (as defined in Eqn. 8) contains all possible values of $(x,\, y)$ such that each $(x,\, y)$ generates the same set D. □

Rules of the Game: Now we lay down the rules followed by the adversary to determine the set D-satisfiable that, in turn, gives the essence of the whole problem.

1. The adversary starts with no information about $(x,\, y)$.
2. The adversary can choose any $(\alpha,\, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ as the *first query*. The *first query* remains same for any seed $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ since the adversary has no information about the seed when she submits the *first query*.
3. Using a strategy, the adversary computes queries adaptively, i.e., based on the previous queries and the corresponding oracle outputs the next query is determined.

4. Suppose, for the seeds (a, b) and (a', b'), the first t queries submitted by the adversary and the corresponding oracle outputs are $(Q1, O1)$, $(Q2, O2), \ldots (Qt, Ot)$ then the $(t+1)$th query for both (a, b) and (a', b') will be the same. Note that the adversary can not distinguish between (a, b) and (a', b') from the first t outputs of the oracle.
5. The game stops the moment the adversary constructs D-satisfiable.

Against this scenario we are going to address the following questions in the subsequent sections.

1. What is the size of D-satisfiable?
2. Is it possible to determine D-satisfiable when D is *entirely* known?
3. Is it possible to determine D-satisfiable when D is *partly* known? By *partly known* we mean that the adversary submits queries fewer than the maximum possible queries. Note, if this case is possible then without submitting any extra query the adversary can always complete the construction of D using an element of D-satisfiable. In such case, how far can the number of submitted queries be reduced to determine D-satisfiable in the worst case?

So far, we hope to have explained enough about the problem that we are going to solve and the challenges associated with it.

Organization: The rest of the paper is organized as follows. Sect. 3.1 elaborates on the relations among different quantities which are used throughout the paper to establish most of the important results. Sect. 3.2 gives formal proofs for the number of solutions for the equations in discussion. Sect. 3.3 determines lower bounds on the number of queries to solve the equations. In Sect. 3.4, we design algorithms that solve the equations with an optimal number of queries. A practical cryptographic application is presented in Sect. 4. Finally, in Sect. 5, we sum up possible extensions of our work.

3 Towards the Solution

3.1 Relations Among Different Quantities

Let an oracle seeded with $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ generate $D \in \mathcal{F}$ (see Sect. 2 for notations and definitions). Let the binary representation of x be $(x_{n-1}, x_{n-2}, \ldots, x_2, x_1, x_0)$. Let $(\alpha, \beta, \gamma) \in D$. Therefore,

$$\gamma_i = x_i \oplus y_i \oplus c_i \oplus \tilde{x}_i \oplus \tilde{y}_i \oplus \tilde{c}_i \tag{9}$$

for all $i \in \mathbb{Z}_n$ where $\tilde{x_i} = x_i \oplus \alpha_i$ and $\tilde{y_i} = y_i \oplus \beta_i$ and the carry bits c_i and $\tilde{c_i}$ are computed recursively in the following way,

$$c_0 = \tilde{c_0} = 0 \tag{10}$$

$$c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i \tag{11}$$

$$\tilde{c_{i+1}} = \tilde{x_i}\tilde{y_i} \oplus \tilde{x_i}\tilde{c_i} \oplus \tilde{y_i}\tilde{c_i} \tag{12}$$

Now we construct a set $\tilde{D}$ in the following fashion,

$$\tilde{D} = \{(\alpha, \beta, \tilde{\gamma} = \alpha \oplus \beta \oplus \gamma) \,|\, (\alpha, \beta, \gamma) \in D\}. \tag{13}$$

Note, $\tilde{\gamma_i} = c_i \oplus \tilde{c_i}$ for all $i \in \mathbb{Z}_n$ (compare with Eqn. (9)). It is easy to identify a bijection between D and $\tilde{D}$ where $(\alpha, \beta, \gamma) \in D$ is mapped to $(\alpha, \beta, \alpha \oplus \beta \oplus \gamma) \in \tilde{D}$. We will, henceforth, use either D or $\tilde{D}$ as the oracle output according to whichever suits our analysis best.

Definition 1. *(A-compatible) Let $\phi \subset A \subseteq \mathbb{Z}_2^n \times \mathbb{Z}_2^n \times \mathbb{Z}_2^n$. An element $(a, b) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ is A-compatible if $(a+b) \oplus ((a \oplus p) + (b \oplus q)) \oplus p \oplus q = r$ for all $(p, q, r) \in A$.*

Definition 2. *(A-consistent) Let $\phi \subset A \subseteq \mathbb{Z}_2^n \times \mathbb{Z}_2^n \times \mathbb{Z}_2^n$. A set $S \subseteq \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ is such that an element $s \in S$ if and only if s is A-compatible. Then the set S is called A-consistent.*

Theorem 1. *D-satisfiable = $\tilde{D}$-consistent.*

Proof. A proof is immediate from the construction of D and $\tilde{D}$.

Suppose $n > 1$. For any $(\alpha, \beta, \tilde{\gamma}) \in \tilde{D}$, $\tilde{\gamma}_{i+1}$ can be computed using x_i, y_i, c_i, α_i, $\beta_i, \tilde{\gamma_i}$ for $i \in \mathbb{Z}_{n-1}$. Table 1 lists the values of $\tilde{\gamma}_{i+1}$ as computed from all possible values of x_i, y_i, c_i, α_i, β_i, $\tilde{\gamma_i}$.

Computation of Important Parameters G_i, $S_{i,0}$ and $S_{i,1}$: We now determine an important quantity, denoted by G_i, from the set of queries and its results. Let the adversary submit a few queries (either in batch or adaptively) to the oracle and construct a nonempty set $A \subseteq \tilde{D}$ (see the beginning Sect. 3.1 for a definition of $\tilde{D}$). Suppose $n > 1$. Now, we construct G_i as follows,

$$G_i = \{(\alpha_i, \beta_i, \tilde{\gamma_i}, \tilde{\gamma_{i+1}}) \,|\, (\alpha, \beta, \tilde{\gamma}) \in A\} \quad \text{for each } i \in \mathbb{Z}_{n-1}. \tag{14}$$

Now, for each $i \in \mathbb{Z}_{n-1}$, from Table 1, we identify (x_i, y_i, c_i) that corresponds to *every* element $(\alpha_i, \beta_i, \tilde{\gamma_i}, \tilde{\gamma_{i+1}}) \in G_i$. $S_{i,j}$ denotes the number of solutions for (x_i, y_i) that correspond to *every* element in G_i and $c_i = j$.

Table 1. All possible values of γ_{i+1} are plotted against all possible values of x_i, y_i, c_i, α_i, β_i, γ_i. A row and a column are denoted by Row(l) (where $l \in \mathbb{Z}_4$) and Col(k) (where $k \in \mathbb{Z}_8$). Row(l)×Col(k) denotes the value on Row(l) and Col(k).

(x_i, y_i, c_i)	$(\alpha_i, \beta_i, \gamma_i)$								
	(0, 0, 0)	(0, 0, 1)	(0, 1, 0)	(0, 1, 1)	(1, 0, 0)	(1, 0, 1)	(1, 1, 0)	(1, 1, 1)	
(0, 0, 0)	0	0	0	1	0	1	1	1	Row(0)
(1, 1, 1)									
(0, 0, 1)	0	0	1	0	1	0	1	1	Row(1)
(1, 1, 0)									
(0, 1, 0)	0	1	0	0	1	1	0	1	Row(2)
(1, 0, 1)									
(1, 0, 0)	0	1	1	1	0	0	0	1	Row(3)
(0, 1, 1)									
	Col(0)	Col(1)	Col(2)	Col(3)	Col(4)	Col(5)	Col(6)	Col(7)	

Example 2. $(S_{i,0}, S_{i,1})$ Suppose, after submission of a few queries to the oracle the adversary constructs a nonempty set $A \subseteq \tilde{D}$. Let $n = 3$ and $A = \{((0, 1, 0), (1, 0, 1), (0, 0, 0)), ((0, 0, 0), (1, 1, 1), (1, 0, 0)), ((0, 0, 1), (0, 1, 1), (1, 1, 0))\}$. Therefore, $G_0 = \{(0, 1, 0, 0), (1, 1, 0, 1)\}$, $G_1 = \{(1, 0, 0, 0), (0, 1, 0, 1), (0, 1, 1, 1)\}$ (see Eqn. 14). Now, from Table 1, G_0 and G_1 correspond to Row(0) and Row(3) respectively. Thus, $S_{0,0} = S_{0,1} = 1, S_{1,0} = S_{1,1} = 1$. □

3.2 Number of Solutions

Before we compute the number of solutions for the equations, we establish three fundamental results in the following propositions and theorem.

Proposition 1. *(Equality of $S_{i,0}$ and $S_{i,1}$) For any nonempty set $A \subseteq \tilde{D}$ and $n > 1$, $S_{i,0} = S_{i,1}$ for all $i \in \mathbb{Z}_{n-1}$ (see Sect. 3.1 for notations and definitions).*

Proof. Let the size of G_i be k_i where $i \in \mathbb{Z}_{n-1}$. From Table 1, it is easy to see that, for any nonempty set $A \subseteq \tilde{D}$, for any $i \in \mathbb{Z}_{n-1}$, $k_i \in \mathbb{Z}_9 \setminus \mathbb{Z}_1$ and the exact value of k_i depends on the set A. Now, from Table 1, for any $i \in \mathbb{Z}_{n-1}$ and for any $k_i \in \mathbb{Z}_9 \setminus \mathbb{Z}_1$, $S_{i,0} = S_{i,1}$. □

We set,

$$S_{i,0} = S_{i,1} = S_i \quad \text{for all } i \in \mathbb{Z}_{n-1}. \tag{15}$$

Theorem 2. *(An equivalence between A and G_i's) Let $\phi \subset A \subseteq \tilde{D}$ and $n > 1$. The following two statements are equivalent.*

1. $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ *is A-compatible.*
2. $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ *is such that, for each* $i \in \mathbb{Z}_{n-1}$*, the triple* $(x_i,\, y_i,\, c_i)$ *corresponds to* every *element* $(\alpha_i,\, \beta_i,\, \tilde{\gamma}_i,\, \tilde{\gamma_{i+1}}) \in G_i$ *in Table 1, where* $c_0 = 0$ *and* $c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i$*.*

Proof. From the construction of G_i, it can be shown that (1) $\Leftrightarrow$ (2). □

Proposition 2. *(Size of A-consistent) Let* $\phi \subset A \subseteq \tilde{D}$ *and* S *denote the size of A-consistent. Then,*

$$S = \begin{cases} 4 \cdot \prod_{i=0}^{n-2} S_i & \text{if } n > 1, \\ 4 & \text{if } n = 1 \end{cases}$$

Proof. Case 1: When $n > 1$. From Theorem 2, S is the number of all possible solutions for $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ that correspond to $G_0,\, G_1,\, \cdots,\, G_{n-2}$ (i.e., $(x_i,\, y_i,\, c_i)$ corresponds to every element in $G_i,\ i \in \mathbb{Z}_{n-1}$). Let M_k denote the number of all possible solutions for $((x_k,\, \cdots, x_0),\, (y_k,\, \cdots, y_0))$ that correspond to $G_0,\, G_1,\, \cdots,\, G_k$ where $k \in \mathbb{Z}_{n-1}$.
Case 1(a): When $n > 2$. We determine the size of the set A-consistent recursively. Let $M_i = M_{i,0} + M_{i,1}$ such that $M_{i,0}$ solutions produce $c_{i+1} = 0$ and $M_{i,1}$ solutions produce $c_{i+1} = 1$. Therefore, for all $i \in \mathbb{Z}_{n-2}$,

$$\begin{aligned} M_{i+1} &= M_{i,0} \cdot S_{i+1,0} + M_{i,1} \cdot S_{i+1,1} \\ &= S_{i+1} \cdot M_i \,. \end{aligned} \tag{16}$$

as $S_{i,0} = S_{i,1}$ for all $i \in \mathbb{Z}_{n-1}$ (see Proposition 1). It is easy to show (a proof is by contradiction) that M_{i+1}, so calculated, gives the number of all possible solutions for $((x_{i+1},\, \cdots, x_0), (y_{i+1},\, \cdots, y_0))$ that correspond to $G_0,\, G_1,\, \cdots G_{i+1}$. From Eqn. (16),

$$M_{n-2} = \prod_{i=0}^{n-2} S_i \tag{17}$$

as $M_0 = S_0$. Note that, for all $(\alpha,\, \beta,\, \tilde{\gamma}) \in A$, r is independent of $(x_{n-1},\, y_{n-1})$. Therefore,

$$S = 4 \cdot \prod_{i=0}^{n-2} S_i \quad \text{if } n > 2. \tag{18}$$

Case 1(b): When $n = 2$. It is easy to show that $S = 4 \cdot S_0$ if $n = 2$.
Case 2: When $n = 1$. It is trivial to show that $S = 4$ if $n = 1$ since for all $(\alpha,\, \beta,\, \tilde{\gamma}) \in A$, $\tilde{\gamma}$ is independent of $(x_{n-1},\, y_{n-1})$. □

As explained in Sect. 3.1, the number solutions for Eqn. (3) or (5) is the size of the set D-satisfiable (see (7)). From this point onwards, we will treat these two equations separately. The set $D \in \mathcal{F}$ (and consequently the corresponding $\tilde{D}$) may correspond to either Eqn. (3) or (5). The relevant equation should be understood according to the context. Note, only Col(0), Col(1), Col(2) and Col(3) of Table 1 are relevant for Eqn. (5) because $\alpha = (0, 0, \cdots 0)_n$ for all queries. Therefore, $(\alpha, \beta, \gamma) \in D$ and $(\alpha, \beta, \tilde{\gamma}) \in \tilde{D}$ will be denoted by $(0, \beta, \gamma)$ and $(0, \beta, \tilde{\gamma})$ in case of Eqn. (5). The following theorem determines the number of solutions for Eqn. (5). A claim similar to that of the following theorem has been made in [5] for a specific case of $n = 32$, which discusses a *differential attack* on Helix [4], without any formal proof. We prove it in a general framework.

Theorem 3. *(Number of Solutions for Eqn. 5) Let the position of the least significant '1' of x in the following equation,*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma$$

be t (following the convention that the position of the least significant bit is zero and the positions of the successive higher order bits are incremented by 1 successively) and $x, y, \beta, \gamma \in \mathbb{Z}_2^n$. Let $f(x, y) = D$ be given. Then the size of D-satisfiable is,
(i) 2^{t+3} when $n - 1 > t \geq 0$. ,
(ii) 2^{n+1} otherwise.

Proof. We consider the set $\tilde{D}$ corresponding to D (see Proposition 1).
(i) When $n-1 > t \geq 0$. We prove it by dividing it into two disjoint cases.
Case 1: When $n - 2 > t \geq 0$. First, we prove the following two lemmas.

Lemma 1. *For any $(0, \beta, \tilde{\gamma}) \in \tilde{D}$, $\tilde{\gamma}_i = 0$ for all $i \in \mathbb{Z}_{t+1}$.*

Proof. If the position of the least significant '1' of x is t then $c_i = \tilde{c}_i = 0$ for all $i \in \mathbb{Z}_{t+1}$ and for all $\beta \in \mathbb{Z}_2^n$ (see Eqn. (10), (11) and (12)). Recall $\tilde{\gamma}_i = c_i \oplus \tilde{c}_i$. This proves the lemma. □

Lemma 2. *For any $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{t+1}$, there exists $(0, \beta, \tilde{\gamma}) \in \tilde{D}$ with $\tilde{\gamma}_i = 1$.*

Proof. We prove the lemma by induction on i. Suppose, there exists $(0, a, b) \in \tilde{D}$ with $b_{i=k} = 1$ for some $k \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$ (induction hypothesis). The statement is true when $i = t + 1$. Select $(0, m, n) \in \tilde{D}$ with $m_t = 1$. Now, the carry bits, as defined in Sect. 3.1, $c_t = \tilde{c}_t = 0$ and $x_t = 1$ which implies $n_{t+1} = 1$. We construct three n-bit integers from a,

1. $a' = (a_{n-1}, a_{n-2}, \cdots, a_{k+1}, 0, a_{k-1}, \cdots, a_0)$
2. $a'' = (a_{n-1}, a_{n-2}, \cdots, a_{k+1}, 1, a_{k-1}, \cdots, a_0)$
3. $a''' = (a_{n-1}, a_{n-2}, \cdots, a_{k+1}, 1, 0, 0, \cdots, 0)$.

Now we select three elements $(0, a', b')$, $(0, a'', b'')$, $(0, a''', b''') \in \tilde{D}$ (such elements exist since, for any $p \in \mathbb{Z}_2^n$, there exists $(0, p, q) \in \tilde{D}$ for some $q \in \mathbb{Z}_2^n$). Note that $b'_k = b''_k = b_k = 1$ and $b'''_k = 0$. From Table 1, at least one of b'_{k+1}, b''_{k+1} and b'''_{k+1} is 1. This proves the lemma. □

Now, we construct $G_i = \{(0, \beta_i, \tilde{\gamma_i}, \tilde{\gamma_{i+1}}) \mid (0, \beta, \tilde{\gamma}) \in \tilde{D}\}$ for each $i \in \mathbb{Z}_{n-1}$ (see Sect. 3.1). From Lemma 1, for each $i \in \mathbb{Z}_{t+1}$,

$$G_i = \{(0, 0, 0, e_{i+1}), (0, 1, 0, f_{i+1})\} \tag{19}$$

for some $e_{i+1}, f_{i+1} \in \mathbb{Z}_2$. Note that, for any two elements $(0, a, b)$, $(0, a', b') \in \tilde{D}$, $b_0 = b'_0 = 0$ and $b_{i+1} = b'_{i+1}$ if $a_i = a'_i$ and $b_i = b'_i$ for any $i \in \mathbb{Z}_{n-1}$ (see Sect. 3.1). Also note that, for any $(0, a, b) \in \tilde{D}$ and any $i \in \mathbb{Z}_n$, one can select $(0, a', b')$, $(0, a'', b'') \in \tilde{D}$ with $(a'_i, b'_i) = (0, b_i)$, $(a''_i, b''_i) = (1, b_i)$. Therefore, from Lemma 2, for each $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$,

$$\begin{aligned} G_i = \{&(0, 0, 0, e_{i+1}), (0, 0, 1, f_{i+1}), \\ &(0, 1, 0, g_{i+1}), (0, 1, 1, h_{i+1})\} \end{aligned} \tag{20}$$

for some $f_{i+1}, g_{i+1}, h_{i+1} \in \mathbb{Z}_2$.

Let $S_{i,j}$ (see Sect. 3.1), denote the number of solutions for (x_i, y_i) that correspond to G_i and $c_i = j$. From Table 1, for each $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$, $S_{i,0} = 1$, $S_{i,1} = 1$. Similarly, $S_{i,0} = 2$, $S_{i,1} = 2$ for each $i \in \mathbb{Z}_{t+1}$.

Let S denote the size of $\tilde{D}$-consistent. From Proposition 2,

$$S = 4 \cdot \prod_{i=0}^{n-2} S_i = 4 \cdot \underbrace{1 \cdot 1 \cdots 1}_{n-t-2 \text{ times}} \cdot \underbrace{2 \cdot 2 \cdots 2}_{(t+1) \text{ times}} = 2^{t+3}.$$

From Proposition 1 the size of D-satisfiable is 2^{t+3}.
Case 2: When $n = t + 2$ and $t \geq 0$. Following a similar way as in *Case 1*, it can be shown that $S = 2^{t+3}$ when $n = t + 2$.
(ii) A proof is similar to the above using Proposition 2. □

Theorem 4. *(Number of Solutions for Eqn. 3) Let $f(x, y) = D$ be given for the equation,*

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$$

$x, y, \alpha, \beta, \gamma \in \mathbb{Z}_2^n$. Then the size of D-satisfiable is 4.

Proof. Case 1: When $n > 2$. We construct $G_i = \{(\alpha_i, \beta_i, \tilde{\gamma_i}, \tilde{\gamma_{i+1}}) \mid (\alpha, \beta, \tilde{\gamma}) \in \tilde{D}\}$ for all $i \in \mathbb{Z}_{n-1}$. Using a similar technique used in Theorem 3, it is easy to show that

$$\begin{aligned} G_i = \{ &(0, 0, 0, e_{i+1}), (0, 0, 1, f_{i+1}), (0, 1, 0, g_{i+1}), \\ &(0, 1, 1, h_{i+1}), (1, 0, 0, m_{i+1}), (1, 0, 1, n_{i+1}), \\ &(1, 1, 0, p_{i+1}), (1, 1, 1, q_{i+1})\} \end{aligned} \tag{21}$$

for some $e_{i+1}, f_{i+1}, g_{i+1}, h_{i+1}, m_{i+1}, n_{i+1}, p_{i+1}, q_{i+1} \in \mathbb{Z}_2$ and $i \in \mathbb{Z}_{n-1} \setminus \{0\}$. Note that G_0 does not contain any member $(\alpha_0, \beta_0, \tilde{\gamma_0}, \tilde{\gamma_1})$ with $\tilde{\gamma_0} = 1$. Exactly the same way as in Theorem 3, for all $i \in \mathbb{Z}_{n-1}$, $S_{i,0}$ and $S_{i,1}$ can be determined from Table 1 that correspond to G_i. We see that $S_{i,0} = S_{i,1} = 1$ for all $i \in \mathbb{Z}_{n-1}$. Therefore, the size of D-satisfiable is 4 (see Proposition 2 and Proposition 1).
Case 2: When $n = 2$. A proof is similar to the above.
Case 3: When $n = 1$. A proof is trivial using Proposition 2. □

3.3 Lower Bounds

Now, we go back to our adversarial framework described in Sect. 2.1. An adversary supplies $(\alpha, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ to the oracle and receives the corresponding γ. Then she calculates $\tilde{\gamma} = \alpha \oplus \beta \oplus \gamma$ and constructs a nonempty set $A \subseteq \tilde{D}$.

Theorem 5. *(Relation between G_i and A-consistent) We consider the equation,*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma\,,$$

where the position of the least significant '1' of x is t and $n - 2 > t \geq 0$. Let $\phi \subset A \subseteq \tilde{D}$ and, for some $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$, G_i contain no element $(0, \beta_i, \tilde{\gamma_i}, \tilde{\gamma_{i+1}})$ with $\tilde{\gamma_i} = 1$. Then the size of A-consistent is 2^{t+3+k} where $k > 0$ (see Sect. 3.1 for a definition of G_i).

Proof. Without loss of generality, assume G_k contains no element $(0, q_k, r_k, r_{k+1})$ with $r_k = 1$ for some $k \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$. Therefore, the set G_k is of one of the following forms,

$$G_k = \{(0, 0, 0, a)\} \quad \text{or } \{(0, 0, 0, a), (0, 1, 0, b)\}.$$

Now, from Table 1, $S_k = 2^k$ for either of the cases, where $k > 0$. Similarly, using Lemma 1, $S_i \geq 2$ for all $i \in \mathbb{Z}_{t+1}$. Also $S_i \geq 1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$. Therefore, from Proposition 2, the size of A-consistent is 2^{t+3+k} where $k > 0$. □

Theorem 6. *A lower bound on the number of queries* $(0, \beta)$ *to solve,*

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma\,,$$

in the worst case of $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ *is*
(i) $(n - t - 1)$*, when* $n - 1 > t \geq 0$*,*
(ii) 1 when $n = 1 + t$ *with* $t > 0$*,*
(iii) 1 when $x = 0$ *and* $n > 1$*,*
(iv) 0 otherwise, i.e., when $n = 1$*,*
where t *is the position of the least significant '1' of* x*.*

Proof. (i) When $n - 1 > t \geq 0$. We first divide it into four disjoint cases. *Case 1.* When $n > 4 + t$. By Theorem 5, a *necessary* condition is that G_{n-2}, constructed for $A \subseteq \tilde{D}$, must have an element $(0, q_{n-2}, r_{n-2}, r_{n-1})$ with $r_{n-2} = 1$ otherwise the number of solutions for (x, y) is 2^{t+3+k} where $k > 0$. But, from Theorem 3, the number of solutions is 2^{t+3}. To have G_{n-2} having an element $(0, q_{n-2}, r_{n-2}, r_{n-1})$ with $r_{n-2} = 1$, G_{n-3} must contain an element $(0, q_{n-3}, r_{n-3}, r_{n-2})$ with $r_{n-2} = 1$.

Let $l(k)$ denote a lower bound on the number of adaptively chosen queries to construct $A \subseteq \tilde{D}$ such that G_i contains an element $(0, q_i, r_i, r_{i+1})$ with $r_{i+1} = 1$ for some $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_k$ in the worst case, where, $k \in \mathbb{Z}_{n-2} \setminus \mathbb{Z}_{t+1}$. Let $p \in \mathbb{Z}_{n-3} \setminus \mathbb{Z}_{t+1}$. Therefore, a worst case lower bound $l(p)$ means, for *any adaptively chosen sequence* of $l(p) - 1$ queries $(l(p) > 0)$, there exists $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ such that G_i contains no element $(0, q_i, r_i, r_{i+1})$ with $r_{i+1} = 1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_p$. Now, for *each adaptively chosen sequence* of $l(p) - 1$ queries we *always* identify an $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ for which all queries produce $r_{i+1} = 0$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_p$. From Table 1, we construct (a, b), $(a', b') \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ for *each* (x, y) in the following fashion. The carry c_j is computed from the preceding j bits of (x, y).

1. (Construction of a and b) $a_i = x_i$ and $b_i = y_i$ for all $i \in \mathbb{Z}_{p+1}$. If $c_i = 0$ set $a_i = 0$, $b_i = 0$ for each $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{p+1}$. If $c_i = 1$ set $a_i = 1$, $b_i = 1$ for each $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{p+1}$.
2. (Construction of a' and b') $a'_i = x_i$ and $b'_i = y_i$ for all $i \in \mathbb{Z}_{p+1}$. If $c_i = 0$ set $a'_i = 0$, $b'_i = 1$ for each $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{p+1}$. If $c_i = 1$ set $a'_i = 1$, $b'_i = 0$ for each $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{p+1}$.

The values of (a_i, b_i) and (a'_i, b'_i) for each $i \in \mathbb{Z}_n$ are chosen from Table 1 in order to have both (a, b) and (a', b') produce the same sequence of oracle outputs as (x, y) does on the selected sequence $l(p) - 1$ queries. A reason is that the least significant $(p+1)$ bits of both (a, b) and (a', b') are the same as that of (x, y). Therefore, on any query, the least significant

$(p+2)$ bits of the oracle output, for both $(a,\, b)$ and $(a',\, b')$ are the same as for $(x,\, y)$. As a result, each of the $l(p)-1$ queries produces oracle output $\tilde{\gamma}$ with $\tilde{\gamma}_{p+1}=0$ for all of $(a,\, b)$, $(a',\, b')$ and $(x,\, y)$. The rest of the $(n-p-1)$ bits of $(a,\, b)$ and $(a',\, b')$ are chosen in a way such that, for each of the $l(p)-1$ queries, $\tilde{\gamma}$ has the most significant $(n-p-2)$ bits zero. Thus, we prove that both $(a,\, b)$ and $(a',\, b')$ produce the same sequence of oracle outputs as $(x,\, y)$ does on the selected sequence $l(p)-1$ queries.

Additionally, if $(\beta_{p+1},\, \tilde{\gamma_{p+1}}) = (0,\, 1)$ for the $l(p)$th query, then $(a,\, b)$ produces $\tilde{\gamma_{p+2}} = 0$ and therefore all other higher order bits of $\tilde{\gamma}$ are also zero. Similarly, if $(\beta_{p+1},\, \tilde{\gamma_{p+1}}) = (1,\, 1)$ then $(a',\, b')$ produces $\tilde{\gamma_{p+2}} = 0$ and consequently all other higher order bits of $\tilde{\gamma}$ are also zero. Therefore, for the chosen sequence of $l(p)$ queries, either $(a,\, b)$ or $(a',\, b')$ produces oracle outputs such that G_i contains all elements with $r_{i+1}=0$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{p+1}$. Now, from Rule 4 in Sect. 2.2, the first $l(p)-1$ queries submitted by the adversary for both $(a,\, b)$ and $(a',\, b')$ are the same as the queries she submits for $(x,\, y)$ and either $(a,\, b)$ or $(a',\, b')$ produces $\tilde{\gamma_i}=0$ for all $i \in \mathbb{Z}_n \setminus \mathbb{Z}_{p+2}$ for each of the $l(p)$ queries. Therefore, we establish that, for *any adaptively chosen* sequence of $l(p)$ queries, there exists $(m,\, n) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ such that G_i contains no element $(0,\, q_i,\, r_i,\, r_{i+1})$ with $r_{i+1}=1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{p+1}$. Therefore, a lower bound on the number of queries to construct $A \subseteq \tilde{D}$ such that G_i contains an element $(0,\, q_i,\, r_i,\, r_{i+1})$ with $r_{i+1}=1$ for some $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{p+1}$ is $l(p)+1$ in the worst case. Therefore,

$$l(p+1) = l(p) + 1\,.$$

Following the recursion,

$$l(n-3) = n-t-4+l(t+1) \tag{22}$$

The following lemma computes a value of $l(t+1)$.

Lemma 3. *Let $n-3 > t \geq 0$. For any adaptively selected sequence of two queries, there exists $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ such that G_i contains no element $(0,\ q_i, r_i,\, r_{i+1})$ with $r_{i+1}=1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_{t+1}$.*

Proof. Let the first two queries and the corresponding oracle outputs be be $(0,\, \beta)$, $(0,\, \beta')$, $\tilde{\gamma}$ and $\tilde{\gamma}'$. Depending *only* on the tth bit of β and β', the oracle returns outputs (i.e., $\tilde{\gamma}$ and $\tilde{\gamma}'$) according to the following rules.

1. If $\beta_t = 0$ then the oracle returns $\tilde{\gamma} = (0,\, 0,\, \cdots,\, 0)_n$.
2. If $\beta_t = 1$ then $\tilde{\gamma}_{t+1}=1$ and all other bits of $\tilde{\gamma}$ are zero.
3. If $\beta'_t = 0$ then the oracle returns $\tilde{\gamma}' = (0,\, 0,\, \cdots,\, 0)_n$.

4. If $\beta'_t = 1$ then $\tilde{\gamma}'_{t+1} = 1$ and all other bits of $\tilde{\gamma}'$ are zero.

Under any of the above input-output combinations one can find from Table 1 that $S_i \geq 1$ for all $i \in \mathbb{Z}_{n-1}$. Therefore, from Proposition 2, the number of solutions for (x, y) under any of the above input-output combinations is *at least* 4. This proves the lemma. □

From Lemma 3, $l(t+1) = 3$. Therefore, from Eqn. 22,

$$l(n-3) = n - t - 1\,.$$

Case 2: When $n = t + 4$, a worst case lower bound is 3. A proof is using Lemma 3.
Case 3: When $n = t + 3$, a worst case lower bound is 2. A reason is, from Table 1 it is clear that, with only one query $S_{n-2} > 1$ which makes the number of solutions for this case greater than 2^{t+3} which is impossible from Theorem 3.
Case 4: When $n = t + 2$, a worst case lower bound on the number of queries is 1. For $n > 1$, this lower bound is trivial.
(ii) When $n = 1 + t$ and $t > 0$, a worst case lower bound is 1. A proof is trivial.
(iii) When $x = 0$ and $n > 1$, a worst case lower bound is 1. A proof is easy.
(iv) A proof is trivial. □

Theorem 7. *A lower bound on the number of queries* (α, β) *to solve,*

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma\,,$$

in the worst case of $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ *is*
(i) 3 *when* $n > 2$,
(ii) 2 *when* $n = 2$,
(iii) 0 *when* $n = 1$.

Proof. (i) When $n > 2$. Let the first two queries and the corresponding oracle outputs be (α, β), (α', β'), $\tilde{\gamma}$ and $\tilde{\gamma}'$. Depending on the two least significant bits of α, β, α' and β', the oracle returns outputs (i.e., $\tilde{\gamma}$ and $\tilde{\gamma}'$) according to the following rules.

1. If $(\alpha_0, \beta_0) = (0, 0)$ then $\tilde{\gamma} = (0, 0, \cdots, 0)_n$.
2. If $(\alpha_0, \beta_0) \neq (0, 0)$ and $(\alpha_1, \beta_1) = (1, 1)$ then $\tilde{\gamma} = (1, 1, \cdots, 1, 0)_n$.
3. If $(\alpha_0, \beta_0) \neq (0, 0)$ and $(\alpha_1, \beta_1) \neq (1, 1)$ then $\tilde{\gamma} = (0, 0, \cdots, 0)_n$.
4. If $(\alpha_0, \beta_0) = (\alpha'_0, \beta'_0)$ then $\tilde{\gamma} = \tilde{\gamma}'$.

5. If $(\alpha_0, \beta_0) \neq (\alpha'_0, \beta'_0) = (0, 0)$ then $\tilde{\gamma}' = (0, 0, \cdots, 0)_n$.
6. If $(\alpha_0, \beta_0) \neq (\alpha'_0, \beta'_0) \neq (0, 0)$ and $(\alpha'_1, \beta'_1) = (0, 0)$ then $\tilde{\gamma}' = (0, 0, \cdots, 0)_n$.
7. If $(\alpha_0, \beta_0) \neq (\alpha'_0, \beta'_0) \neq (0, 0)$ and $(\alpha'_1, \beta'_1) = (1, 1)$ then $\tilde{\gamma}' = (1, 1, \cdots, 1, 0)_n$.
8. If $(\alpha_0, \beta_0) \neq (\alpha'_0, \beta'_0) \neq (0, 0)$, $(\alpha'_1, \beta'_1) \in \{(0, 1), (1, 0)\}$ and $(\alpha_1, \beta_1) = (\alpha'_1, \beta'_1)$ then $\tilde{\gamma}' = (0, 0, \cdots, 0)_n$.
9. If $(\alpha_0, \beta_0) \neq (\alpha'_0, \beta'_0) \neq (0, 0)$, $(\alpha'_1, \beta'_1) \in \{(0, 1), (1, 0)\}$ and $(\alpha_1, \beta_1) \neq (\alpha'_1, \beta'_1)$ then $\tilde{\gamma}'_0 = 0$ and $\tilde{\gamma}'_i = 1 \oplus \tilde{\gamma}_i$ for all $i \in \mathbb{Z}_n \setminus \mathbb{Z}_1$.

From the oracle outputs produced according to the above rules on the first two queries, one can show, using Table 1, that one of the following cases occurs.

1. $S_0 \geq 2$ and $S_i \geq 1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_1$.
2. $S_1 \geq 2$ and $S_i \geq 1$ for all $i \in \mathbb{Z}_{n-1} \setminus \{1\}$.
3. $S_0 \geq 2$, $S_1 \geq 2$ and $S_i \geq 1$ for all $i \in \mathbb{Z}_{n-1} \setminus \mathbb{Z}_2$.

Clearly, for any of the above cases, the number of valid solutions S, derived from the results of the queries, is *at least* 8 which is not the case with this equation (see Theorem 4). Therefore, a lower bound on the number of queries in the worst case is 3.
(ii) When $n = 2$. Using Table 1, a proof is similar to the proof for *(i)*.
(iii) When $n = 1$. A proof is trivial. □

3.4 Optimal Algorithms

We design two algorithms Algo1 and Algo2, described in Fig. 1 and Fig. 2 respectively, to show that our lower bounds on the number of queries, computed in Sect. 3.3, are optimal. Notations used are consistent with the present analysis. The oracle O returns $\tilde{\gamma}$ on input (α, β). The variable T denotes Table 1. In Algo1, Least-Significant-one(p) computes the least significant '1' of p. The following proposition will be used to prove the correctness of Algo1 and Algo2.

Proposition 3. *Let G_i, constructed from the oracle output $A = \tilde{D}$, be known for all $i \in \mathbb{Z}_{n-1}$ ($n > 1$). Let L_i contain all triples (x_i, y_i, c_i) such that each triple corresponds to all elements in G_i (as computed in Table 1). Let a set M be constructed from L_i's in the following way,*

$$\begin{aligned} M = \{&((x_{n-1}, x_{n-2}, \cdots, x_0), (y_{n-1}, y_{n-2}, \cdots, y_0)) \\ &|(x_{n-1}, y_{n-1}) \in \mathbb{Z}_2^2, (x_i, y_i, c_i) \in L_i, i \in \mathbb{Z}_{n-1}, c_0 = 0, \\ &c_{j+1} = x_j y_j \oplus x_j c_j \oplus y_j c_j, j \in \mathbb{Z}_{n-1}\}. \end{aligned}$$

Algo1(**Input**: Oracle O, n, Table T; **Output**: a set of lists)

1. If $n \leq 0$ then exit with a comment {"Invalid Input"}.
2. If $n = 1$ then return an empty set {} and exit.
3. $\beta = (111 \cdots 11)_n$
4. $\tilde{\gamma} = O(\beta)$
5. if $\tilde{\gamma} = 0$
    6. For each $i \in \mathbb{Z}_{n-1}$
        7. $G_i = \{(0, 0, 0, 0), (0, 1, 0, 0)\}$
    8. Go to Step 28.
9. $t =$ Least-Significant-one$(\tilde{\gamma})$
10. $t = t - 1$
11. For each $i \in \mathbb{Z}_t$
    12. $G_i = \{(0, 0, 0, 0), (0, 1, 0, 0)\}$
13. $G_t = \{(0, 0, 0, 0), (0, 1, 0, 1)\}$
14. if $t = n - 2$, Go to Step 28.
15. $\beta' = (111 \cdots, \beta'_{t+1} = 1, 00 \cdots 0)$
16. $\tilde{\gamma}' = O(\beta')$
17. $G_{t+1} = \{(0, 0, 0, 0), (0, 1, 1, \tilde{\gamma}_{t+2}), (0, 1, 0, \tilde{\gamma}'_{t+2})\}$
18. if $t = n - 3$, Go to 28.
19. For each $i \in \mathbb{Z}_{n-t-1} \setminus \mathbb{Z}_2$, in increasing order
    20. if $\tilde{\gamma}_{t+i} == \tilde{\gamma}'_{t+i} == 1$
        21. $\beta' = (111 \cdots \beta'_{t+i-1} = 0, 00 \cdots 0)$
        22. $\tilde{\gamma}' = O(\beta')$, Go to Step 27
    23. if $\tilde{\gamma}_{t+i} == \tilde{\gamma}'_{t+i} == 0$
        24. if $\tilde{\gamma}_{t+i-1} = 1$ swap $((\beta, \tilde{\gamma}), (\beta', \tilde{\gamma}'))$
        25. $\beta' = (\cdots, \beta'_{t+i}, \beta'_{t+i-1} = 0, \beta'_{t+i-2}, \cdots)$
        26. $\tilde{\gamma}' = O(\beta')$
    27. $G_{t+i} = \{(0, 0, 0, 0), (0, 1, \tilde{\gamma}_{t+i}, \tilde{\gamma}_{t+i+1}), (0, 1, \tilde{\gamma}'_{t+i}, \tilde{\gamma}'_{t+i+1})\}$
28. Using the table T, collect all (x_i, y_i, c_i) corresponding to G_i in list L_i.
29. Return the set $\{L_i | i \in \mathbb{Z}_{n-1}\}$.

Fig. 1. An Algorithm to solve the equation $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ with an optimal number of queries.

Then,
(i) $M = D$*-satisfiable;*
(ii) there exists an algorithm such that D*-satisfiable can be constructed from* L_i*'s with memory* $n \cdot 2^{O(n)}$ *and time* $n \cdot 2^{O(n)}$*.*

Proof. (i) From Theorem 2 and 1,

$$(a,\, b) \in M \Rightarrow (a,\, b) \in \tilde{D}\text{-compatible} \Rightarrow (a,\, b) \in D\text{-satisfiable}. \quad (23)$$

Now from Lemma 1, 2, Proposition 2 and Table 1 it is easy to see that the size of M is
1) 2^{t+3} if $n - 1 > t \geq 0$,
2) 2^{n+1} otherwise,
where t denotes the position of the least significant '1' of x and the pair $(x,\, y)$ is the seed of the oracle which outputs $\tilde{D}$.

The above results together with Theorem 3 show that $M = D$-satisfiable.
(ii) We construct a set M_k recursively where $k \in \mathbb{Z}_n \setminus \mathbb{Z}_2$ and $M_1 = \{((x_0),\, (y_0))|(x_0,\, y_0,\, c_0) \in L_0\}$,

$$\begin{aligned} M_k = \{&((x_{k-1},\, \cdots,\, x_0),\, (y_{k-1},\, \cdots,\, y_0))|(x_{k-1},\, y_{k-1},\, c_{k-1}) \in L_{k-1}, \\ &((x_{k-2},\, \cdots,\, x_0),\, (y_{k-2},\, \cdots,\, y_0)) \in M_{k-1}\}. \end{aligned}$$

Now, we construct

$$\begin{aligned} M_n = \{&((x_{n-1},\, \cdots,\, x_0),\, (y_{n-1},\, \cdots,\, y_0))|(x_{n-1},\, y_{n-1}) \in \mathbb{Z}_2^2, \\ &((x_{n-2},\, \cdots,\, x_0),\, (y_{n-2},\, \cdots,\, y_0)) \in M_{n-1}\}. \end{aligned}$$

Note that the size of each L_i is $O(1)$ since the size of the Table 1 is $O(1)$. Also note that the size of M_n is $2^{O(n)}$ and therefore the asymptotic memory requirement to construct M_n recursively following the above algorithm is $n \cdot 2^{O(n)}$ since $k = O(n)$ and M_{k+1} can be constructed from M_k only. It is trivial to show that the time T_1 to construct M_n from L_i's is also $2^{O(n)}$ (assuming asymptotic time to copy and delete an n-bit string $O(1)$). Note that $M \subseteq M_n$. To compute M from M_n we have to filter out the false candidates from M_n by performing the following steps for each member $(x,\, y) \in M_n$,

1. If $(x_0,\, y_0,\, 0)$ does not belong to L_0 then $(x,\, y)$ is a false candidate otherwise set $i = 1$ and proceed to the next step.
2. Compute the carry bit $c_i = x_{i-1}y_{i-1} \oplus x_{i-1}c_{i-1} \oplus y_{i-1}c_{i-1}$; then check whether $(x_i,\, y_i,\, c_i)$ belongs to L_i. If the answer is 'yes' then increment

i by one; if $i < n-1$ carry out this step once more, if $i = n-1$, then (x, y) is a member of M. If the answer is 'no' then (x, y) is a false candidate.

The time T_2, required to perform the above steps for all members of M_n, is $n \cdot 2^{O(n)}$. Therefore, $T = T_1 + T_2 = n \cdot 2^{O(n)}$. Thus, the set M can be constructed from L_i's with memory $n \cdot 2^{O(n)}$ and time $n \cdot 2^{O(n)}$. From the first part of the proposition we already know that $M = D$-satisfiable. □

Correctness of Algo1: To show that Algo1 (see Fig. 1) is correct we *only* need show that the computed L_i corresponds to oracle output $A = \tilde{D}$ for all $i \in \mathbb{Z}_{n-1}$, since there exists an algorithm to determine the corresponding D-satisfiable from the computed L_i's with finite time and memory without any extra query (see Proposition 3). We prove the correctness of Algo1 by considering all possible cases individually.

1. When $n \leq 0$, Algo1 correctly returns "Invalid Input" (see line 1).
2. When $n = 1$, Algo1 returns an empty set (see line 2). This empty set is an indicator showing D-satisfiable= $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$. Therefore, Algo1 is correct for $n = 1$ (see Theorem 3). We will later see that the algorithm never returns empty set for $n > 1$.
3. When $n > 1$ and the oracle output $\tilde{\gamma} = 0$ on query $\beta = (11 \cdots 11)_n$ then the seed of the oracle (x, y) is such that the least $(n-1)$ bits of x are each zero. Then from Lemma 1 and Eqn. (19), $G_i = \{(0, 0, 0, 0), (0, 1, 0, 0)\}$ for all $i \in \mathbb{Z}_{n-1}$. Therefore, Algo1 determines G_i's that correspond to oracle output $A = \tilde{D}$ (line 7). Hence, L_i's, computed in line 28, also correspond to oracle output $A = \tilde{D}$.
4. When $n > 1$ and the oracle output $\tilde{\gamma} \neq 0$ on query $\beta = (11 \cdots 11)_n$, then line 9 and 10 compute the position of the least significant '1' of x (denoted by t). Line 12 and 13 compute G_i for all $i \in \mathbb{Z}_{t+1}$. Using Table 1, Lemma 1 and 2 it can be shown that G_i for all $i \in \mathbb{Z}_{t+1}$ correspond to oracle output $A = \tilde{D}$. Hence, L_i for all $i \in \mathbb{Z}_{t+1}$ are also correct.
   – If $t = n-2$ then the construction of L_i for all $i \in \mathbb{Z}_{n-1}$ is complete (line 14).
   – If $n-2 > t$ then a second query $\beta' = (111 \cdots, \beta'_{t+1} = 1, 00 \cdots 0)$ is submitted (line 15 and 16). Now, $G_{t+1} = \{(0, 0, 0, 0), (0, 1, 1, \tilde{\gamma}_{t+2}), (0, 1, 0, \tilde{\gamma}'_{t+2})\}$ (line 17). Note that the size of G_{t+1} is less than what it should be if constructed from the oracle output $A = \tilde{D}$ (see Eqn.(20)). Now, we observe an interesting property of Table 1. If we choose any two columns from Col(1), Col(2) and Col(3), we see that each row of

the partially specified table is unique. Therefore, G_{t+1} corresponds to exactly one row in Table 1. Note that Eqn. (20) requires that G_{t+1} correspond to a single row in Table 1. Therefore, L_{t+1}, constructed from G_{t+1}, also corresponds to the oracle output $A = \tilde{D}$. If $t = n-3$, then the construction of L_i for all $i \in \mathbb{Z}_{n-1}$ is complete (see line 18).
– If $n-3 > t$ then a loop between lines 19 and 27 is executed. The jth iteration of the loop determines G_{t+2+j} (iterations are numbered 0, 1, 2, and so on). The execution continues till G_{n-2} is evaluated. At the start of every iteration, oracle outputs on exactly two queries are known. At the start of jth iteration let the queries and the corresponding outputs be $(\beta, \tilde{\gamma})$ and $(\beta', \tilde{\gamma}')$. Note that $\beta_{t+1+j} = \beta'_{t+1+j} = 1$ and $\tilde{\gamma}_{t+1+j} \neq \tilde{\gamma}'_{t+1+j}$. Now there are three possible cases. Case 1: If $\tilde{\gamma}_{t+2+j} = \tilde{\gamma}'_{t+2+j} = 1$ then a new query $\beta' = (111 \cdots \beta'_{t+1+j} = 0, 00 \cdots 0)$ is submitted and the corresponding oracle output $\tilde{\gamma}'$ is collected (see line 21 and 22). It is clear from β' that $\tilde{\gamma}'_{t+2+j} = 0$. As a result, we get $\beta_{t+2+j} = \beta'_{t+2+j} = 1$ and $\tilde{\gamma}_{t+2+j} \neq \tilde{\gamma}'_{t+2+j}$. Now, we determine $G_{t+2+j} = \{(0, 0, 0, 0), (0, 1, 1, \tilde{\gamma}_{t+3+j}), (0, 1, 0, \tilde{\gamma}'_{t+3+j})\}$ (see line 27). As argued in the earlier case L_{t+2+j}, corresponding to G_{t+2+j}, also corresponds to the entire set of oracle outputs $A = \tilde{D}$. Case 2: If $\tilde{\gamma}_{t+2+j} = \tilde{\gamma}'_{t+2+j} = 0$ then a new query $\beta' = (111, \cdots, \beta'_{t+1+j} = 0, \beta'_{t+j}, \cdots)$ is submitted (the corresponding output is $\tilde{\gamma}'$), assuming $\tilde{\gamma}_{t+1+j} = 0$ without loss of generality (see line 25). Now, from Row(2) of Table 1 (consider only the first four columns as they are only relevant for the equation in discussion) $\tilde{\gamma}'_{t+2+j} = 1$. Therefore, $G_{t+2+j} = \{(0, 0, 0, 0), (0, 1, 1, \tilde{\gamma}'_{t+3+j}), (0, 1, 0, \tilde{\gamma}_{t+3+j})\}$ and L_{t+2+j} is correct (argument is similar as before that G_{t+2+j} refers to a unique row in Table 1). Case 3: If $\tilde{\gamma}_{t+2+j} \neq \tilde{\gamma}'_{t+2+j}$ then the execution jumps to line 27 and the computed $G_{t+j+2} = \{(0, 0, 0, 0), (0, 1, \tilde{\gamma}_{t+j+2}, \tilde{\gamma}_{t+i+3}), (0, 1, \tilde{\gamma}'_{t+j+2}, \tilde{\gamma}'_{t+j+3})\}$. Therefore, L_{t+2+j} corresponds to $\tilde{D}$.

Thus, for any n and t, Algo1 constructs L_i, for all $i \in \mathbb{Z}_{n-1}$ (as defined in Proposition 3), that corresponds to the entire set of oracle output $A = \tilde{D}$. Therefore, Algo1 correctly solves Eqn.(5).

Correctness of Algo2: The correctness of Algo2 (see Fig. 2) is proved the same way as Algo1 is proved. We will only verify whether Algo2 computes L_i corresponding to oracle output $A = \tilde{D}$ for any $i \in \mathbb{Z}_{n-1}$ (see Proposition 3 for a method to construct L_i). The solutions for $n \leq 1$ are given in line 1 and 2 (an explanation is similar to that for Algo1). Now we take a closer look at the first two queries $(a, b) = ((111, \cdots 11)_n,$

Algo2(**Input**: Oracle O, n, Table T; **Output**: a set of lists)

1. If $n \leq 0$ then exit with a comment {"Invalid Input"}.
2. If $n = 1$ then return an empty set {} and exit.
3. $(a, b) = ((111, \cdots 11)_n, (000, \cdots 00)_n)$
4. $\tilde{\gamma} = O(a, b)$
5. $(c, d) = ((\cdots 101010)_n, (\cdots 010101)_n)$
6. $\tilde{\gamma}' = O(c, d)$
7. For each $i \in \mathbb{Z}_{n-1}$
    8. $G_i = \{(a_i, b_i, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (c_i, d_i, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}$
9. For each $i \in \mathbb{Z}_{n-1}$
    10. Using table T, extract all possible (x_i, y_i, c_i) corresponding to G_i and store it in L_i.
11. If $|L_i| = 2$ for all $i \in \mathbb{Z}_{n-1}$ then Go to step 27.
12. For each $i \in \mathbb{Z}_{n-1}$ and i even
    13. if $|L_i| = 4$ then collect $(x_{i-1}, y_{i-1}, 0)$ from L_{i-1} and $(1, 0, \tilde{\gamma}_{i-1}, \tilde{\gamma}_i) \in G_{i-1}$
        14. Select $(\alpha_{i-1}, \beta_{i-1})$ from T such that $(x_{i-1}, y_{i-1}, 0)$ corresponds to both $(\alpha_{i-1}, \beta_{i-1}, 0, \tilde{\gamma}_i)$ and $(\alpha_{i-1}, \beta_{i-1}, 1, \tilde{\gamma}_i)$.
        15. $(c_{i-1}, d_{i-1}) = (\alpha_{i-1}, \beta_{i-1})$
16. For each $i \in \mathbb{Z}_{n-1}$ and i odd
    17. if $|L_i| = 4$ then collect $(x_{i-1}, y_{i-1}, 0)$ and $(1, 0, \tilde{\gamma}_{i-1}, \tilde{\gamma}_i) \in G_{i-1}$
        18. Select $(\alpha_{i-1}, \beta_{i-1})$ from T such that $(x_{i-1}, y_{i-1}, 0)$ corresponds to both $(\alpha_{i-1}, \beta_{i-1}, 0, 1 \oplus \tilde{\gamma}_i)$ and $(\alpha_{i-1}, \beta_{i-1}, 1, 1 \oplus \gamma_i)$
        19. $(c_{i-1}, d_{i-1}) = (\alpha_{i-1}, \beta_{i-1})$
20. $\tilde{\gamma}' = O(c, d)$
21. For each $i \in \mathbb{Z}_{n-1}$
    22. $G_i = \{(a_i, b_i, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (c_i, d_i, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}$
23. For each $i \in \mathbb{Z}_{n-1}$
    24. Using table T, extract all possible (x_i, y_i, c_i) corresponding to G_i and store it in L'_i.
25. For each $i \in \mathbb{Z}_{n-1}$
    26. If $|L_i| = 4$, then assign $L_i = L'_i$.
27. Return the set $\{L_i | i \in \mathbb{Z}_{n-1}\}$.

Fig. 2. An Algorithm to solve the equation $(x + y) \oplus ((x + \alpha) + (y \oplus \beta)) = \gamma$ with an optimal number of queries.

$(000, \cdots 00)_n$) and $(c, d) = (\ (\cdots 101010)_n,\ (\cdots 010101)_n)$ and their corresponding outputs $\tilde{\gamma}$ and $\tilde{\gamma}'$ (see line 2 and 3). Note that if i is even then G_i is of the following form,

$$G_i = \{(1, 0, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (0, 1, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}. \tag{24}$$

If i is odd then G_i is of the following form,

$$G_i = \{(1, 0, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (1, 0, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}. \tag{25}$$

From Eqn. (21), G_i for any $i \in \mathbb{Z}_{n-1}$ should correspond to exactly one row in Table 1. Now, we observe two interesting properties of Table 1. Firstly, in Eqn. (24), $\tilde{\gamma}_i \neq \tilde{\gamma}'_i$ implies and is implied by the fact that G_i corresponds to two rows of Table 1. Similarly, in Eqn. (25), $\tilde{\gamma}_i = \tilde{\gamma}'_i$ implies and is implied by the fact that G_i corresponds to two rows of Table 1. Secondly, if G_i corresponds to two rows then G_{i-1} corresponds to exactly one row. A reason is, if i is even then $\tilde{\gamma}_{i-1} \neq \tilde{\gamma}'_{i-1}$; if i is odd then $\tilde{\gamma}_{i-1} = \tilde{\gamma}'_{i-1}$.

Based on these observations, we construct G_i and L_i for all $i \in \mathbb{Z}_{n-1}$ (see line 7, 8, 9, 10). Note that if G_i refers to only one row of Table 1, then $|L_i| = 2$ and vice-versa. Similarly, if G_i refers to two rows of Table 1, then $|L_i| = 4$ and vice-versa. After submission of the first two queries (a, b) and (c, d) if G_i corresponds to only one row (or $|L_i| = 2$) for each $i \in \mathbb{Z}_{n-1}$ then our job is done (see line 11). If $|L_i| = 4$ for some $i \in \mathbb{Z}_{n-1}$ then we will submit a third query by modifying the query (c, d) according to the rules described in lines 12 to 15 and lines 16 to 19 (see Fig. 2). Now we take the oracle output $\tilde{\gamma}' = O(c, d)$ (line 20). The query (c, d) is selected in such a way that, if $|L_i| = 4$ for an even i then $\tilde{\gamma}_i = \tilde{\gamma}'_i$ (see line 14 and 15); if $|L_i| = 4$ for an odd i then $\tilde{\gamma}'_i = 1 \oplus \tilde{\gamma}_i$ (see line 18 and 19). We now, construct G_i and L'_i using queries (a, b), (c, d) and the outputs $\tilde{\gamma}$ and $\tilde{\gamma}'$ (see lines 21 to 24). Clearly, if $|L_i| = 4$ then $|L'_i| = 2$. We replace all $|L_i| = 4$ with $L_i = L'_i$ (line 25 and 26). Now, $|L_i| = 2$ for all $i \in \mathbb{Z}_{n-1}$ (note that Eqn. (21) enforces $|L_i| = 2$ for all $i \in \mathbb{Z}_{n-1}$). Finally, we conclude that Algo2 is correct as it computes L_i, for all $i \in \mathbb{Z}_{n-1}$, that are compatible with the entire set of oracle output $A = \tilde{D}$.

Theorem 8. *Worst case lower bounds on the number of queries, as derived in Theorem 6 and 7, to solve (5) and (3) respectively, are optimal.*

Proof. Upper bound on the number of queries required by Algo1 (see Fig. 1) is listed below.
(i) 0 when $n = 1$ (see line 2).

(ii) 1 when $n = 1 + t$ and $t > 0$. The required query is shown in line 3.
(iii) 1 when $x = 0$ and $n > 1$. The only required query is shown in line 3.
(iv) $(n - t - 1)$, when $n - 1 > t \geq 0$ and t is the position of the least significant '1' of x (the position is determined in line 10). The first two queries are shown in line 3 and 15. The loop in lines 19 to 27 requires a maximum of $(n-t-3)$ queries. Note that each iteration submits at most one query in either line 22 or 26.

Therefore, lower bound computed in Theorem 6 is optimal.

Upper bound on the number of queries required by Algo2 (see Fig. 2) is as follows.
(i) 0 when $n = 1$ (see line 2).
(ii) 2 when $n = 2$. One can show from Table 1 that, for $n = 2$, on the queries shown in lines 3 and 4 , $|L_1| = 2$. Therefore, a third query is not required (see line 11).
(iii) 3 when $n > 2$ (third query is submitted in line 20).

Therefore, lower bound computed in Theorem 7 is optimal. □

Asymptotic Time and Memory : For Algo1, the memory complexity is $\theta(n)$. The time complexity of Algo1 is $O(n^2)$ because of the loop in lines 19 to 27 (assuming that the oracle takes $O(n)$ time to return the output $\tilde{\gamma}$ on any input $(0, \beta)$). For Algo2, the memory and time complexities are $\theta(n)$ each.

4 Improving an Attack on the Helix Stream Cipher

Helix which was proposed by Ferguson *et al.* [4] is a stream cipher with a combined MAC functionality. The primitive uses combination of addition and XOR to generate pseudorandom bits. Recently a differential attack was found against Helix by Muller [5]. They have solved the equation, $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ for (x, y) to recover secret information (x, y) using β and the corresponding γ. Every time β corresponds to a chosen plaintext. They solved the equation many times to launch an attack. The algorithm they used requires $3(n - 1)$ queries every time. Therefore, the most natural challenge, from an algorithmic point of view, is to reduce the number of queries and if possible to attain an optimality. For $n = 32$ bits (which is the size of the Helix output word), they required 93 queries whereas Algo1 (see Fig. 1) takes *at most* 31 queries when the position of the least significant '1' of x (denoted by t) is zero. Note that, if $t > 0$ then the number of queries is less. However, the most important fact is that the number of queries cannot be further reduced in the worst case as

our algorithm is worst case optimal. This fact can be straightaway used to reduce the data complexity of that particular attack on Helix cipher by, *at least*, a factor of 3 without exploring other possibilities to reduce the data further. However, in the best case, Algo1 requires only 2 queries (one can easily show that there exists seed of the oracle $(x,\ y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ with any $t \in \mathbb{Z}_{n-2}$ for which (5) can be solved with only 2 queries) and the improvement in such case is a factor of 46.5.

5 Conclusion and Further Research

In this paper we have dealt with two equations that mix operations *addition modulo* 2^n and *exclusive-or*. Such equations are the basic building blocks for many symmetric ciphers. Using simple relations among carry-bits and the input bits we have solved those equations with an optimal number of queries. These results have superseded the previous best known results by a constant factor. Furthermore, the results cannot be further improved as we reached an optimality. To show a practical use we reduce the data complexity of an attack on Helix stream cipher. In addition to that, the solution techniques motivate further research to solve more complex equations (e.g. combination of *modular addition*, *exclusive-or* and *modular multiplication*) with a *minimum* number of queries and also to calculate differential properties of addition with complexities less than the existing ones. We expect that these results will be useful in the cryptanalysis of other ciphers too.

References

1. E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Crypto '90* (A. Menezes, S. A. Vanstone, eds.), vol. 537 of *LNCS*, pp. 2-21, Springer-Verlag, 1991.
2. H. Lipmaa, S. Moriai, "Efficient Algorithms for Computing Differential Properties of Addition," *FSE 2001* (M. Matsui, ed.), vol. 2355 of *LNCS*, pp. 336-350, Springer-Verlag, 2002.
3. O. Staffelbach, W. Meier, "Cryptographic Significance of the Carry for Ciphers Based on Integer Addition," *Crypto '90* (A. Menezes, S. A. Vanstone, eds.), vol. 537 of *LNCS*, pp. 601-614, Springer-Verlag, 1991.
4. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, T. Kohno, "Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive," *Fast Software Encryption 2003* (T. Johansson, ed.), vol. 2887 of *LNCS*, pp. 330-346, Springer-Verlag, 2003.
5. F. Muller, "Differential Attacks against the Helix Stream Cipher," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 94-108, Springer-Verlag, 2004.

6. L. Lipmaa, J. Wallén, P. Dumas, "On the Addititive Differential Probability of Exclusive-Or," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 317-331, Springer-Verlag, 2004.
7. C. Burwick, D. Coppersmith, E. D'Avignon, Y. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford and N. Zunic, "MARS – A Candidate Cipher for AES," Available Online at `http://www.research.ibm.com/security/mars.html`, June 1998.
8. R. L. Rivest, M. Robshaw, R. Sidney, Y. L. Yin, "The RC6 Block Cipher," Available Online at `http://theory.lcs.mit.edu/ rivest/rc6.ps`, June 1998.
9. X. Lai, J. L. Massey, S. Murphy, "Markov Ciphers and Differential Cryptanalysis," *Eurocrypt '91* (W. Davis, ed.), vol. 547 of *LNCS*, pp. 17-38, Springer-Verlag, 1991.
10. B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C.Hall, N. Ferguson, "The Twofish Encryption Algorithm: A 128-Bit Block Cipher," John Wiley & Sons, April 1999, ISBN: 0471353817.
11. D. E. Knuth, *"The Art of Computer Programming," vol. 2, Seminumerical Algorithms,* Addison-Wesley Publishing Company, 1981.
12. A. Klimov, A. Shamir, "New Cryptographic Primitives Based on Multiword T-Functions," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 1-15, Springer-Verlag, 2004.
13. A. Klimov, A. Shamir, "Cryptographic Applications of T-Functions," *Selected Areas in Cryptography 2003* (M. Matsui, R. J. Zuccherato, eds.), vol. 3006 of *LNCS*, pp. 248-261, Springer-Verlag, 2004.
14. J. Wallen, "Linear Approximations of Addition Modulo 2^n," *Fast Software Encryption 2003* (T. Johansson, ed.), vol. 2887 of *LNCS*, pp. 261-273, Springer-Verlag, 2003.