A shortened version of this paper appears under the same title
in the proceedings of *Indocrypt 2005* (S. Maitra, C.E. Venimadhavan, R. Venkatesan (eds.)), LNCS, Springer-Verlag.

Near Optimal Algorithms for Solving Differential Equations of Addition with Batch Queries*

Souradyuti Paul Bart Preneel

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

Abstract

Mixing *modular addition* (+) with *exclusive-or* (⊕) is extensively used in design of symmetric ciphers as the operations are very fast and their combination is non-linear over GF(2). The paper investigates the strength of *modular addition* against *differential cryptanalysis* (DC) where the differences of inputs and outputs are expressed as *XOR*. In particular, we solve two very frequently used equations (1) $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ and (2) $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$, known as the *differential equations of addition* (DEA), with a set of *batch* queries. Although solution to this problem with *adaptive* queries, which is easier and less practical, was previously known, a *nontrivial* solution with *batch* queries has remained open. The two major contributions of the paper are (*i*) the determination of lower bounds on the required number of *batch* queries to solve the equations and (*ii*) the design of two algorithms which solve them with queries close to optimal. Our algorithms require 2^{n-2} and 6 queries to solve (1) and (2) where the lower bounds are $\frac{3}{4} \cdot 2^{n-2}$ (theoretically proved) and 4 (based on extensive experiments) respectively. This exponential lower bound is an important theoretical benchmark which certifies (1) as *strong* against DC. On the other hand, the constant number of queries to solve (2) discovers a major weakness of *modular addition* against DC.

Muller, at FSE'04, showed a key recovery attack on the Helix stream cipher (presented at FSE'03) with 2^{12} *adaptive chosen plaintexts* (ACP). However, the data complexity of the attack with *chosen plaintexts* (CP) was not known previously. Using our results we recover the secret key of the Helix cipher with only $2^{35.64}$ *chosen plaintexts* (CP) which has so far been the only CP attack on this cipher (the attack is under the same assumption as that of Muller's attack). Considering the abundant use of this component, the results seem useful to evaluate the security of many block ciphers against DC.

Keywords. Differential Cryptanalysis, Addition, Lower bound, Binary Tree.

*This work was supported in part by the Concerted Research Action (GOA) Mefisto 2000/06 and Ambiorix 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

1 Introduction

Addition and XOR. Mixing two different group operations is a common technique adopted by the designers of symmetric ciphers to make cryptanalysis difficult. The main reason behind the wide use of mixing *modular addition* (or simply *addition*) with XOR is their high speed on modern machines and their non-linearity over GF(2). Helix [8], IDEA [14], Mars [3], RC6 [19], Twofish [20] and the MD-family of hash functions are a few applications of *addition* and XOR. Plenty of research has also been spent on analyzing behaviors of addition, XOR and their mixing. Staffelbach and Meier worked on determination of the probability distribution of the carry for integer addition [21]. Wallén investigated the linear approximations of modular addition [23]. Lipmaa *et al.* dealt with the equation $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$ and its dual to investigate the differential properties [15, 16]. Paul and Preneel showed that the satisfiability of an arbitrary set *differential equations of addition* is in the complexity class P and designed algorithms to solve them efficiently [18].

Batch and Adaptive Queries. Cryptologic analogue of a query is a plaintext (or a ciphertext depending on the mode of attack). Similarly, attacks based on *batch* and *adaptive* queries are equivalent to *chosen plaintext* (CP) attack and *adaptive chosen plaintext* (ACP) attack respectively. A CP attack is more practical than the corresponding ACP attack because, in the latter case, we assume the attacker to be powerful enough to submit queries adaptively, i.e., the next query is submitted based on the answers to the previous queries – a situation which is difficult to implement in practice. There is a large number of research papers launching CP and ACP attacks on many practical ciphers. For example, the boomerang attack introduced by Wagner is an ACP attack [22] and, therefore, less practical. The slide attacks by Biryukov and Wagner can be implemented using both CP and ACP but with different amount of data and time [5]. The time-memory trade-off attack by Hellman [9] and the differential attack on DES by Biham and Shamir [4] are CP attacks; so they are attributed more practical importance. In this paper we will deal with a very frequently encountered set of equations in secret key cryptography, known as *differential equations of addition* (explained in the next paragraph), to solve them with a set of *batch* queries (equivalent to a CP attack).

Summary of the Results. *Addition mod* 2^n is extensively used as a block cipher component. *Differential Cryptanalysis* (DC) is one of the most powerful attacks against block ciphers [4]. In this paper we investigate the security of *addition* under DC. In particular, we deal with the following two equations where differences of inputs and outputs of *addition* are expressed as *exclusive-or*

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

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

These equations are known as *differential equations of addition* (DEA). In course of cryptanalysis of MD5, Berson observed in 1992 that, for large n, it is *hard* to analyze *modular addition* when differences are expressed as XOR [2]. The apparent difficulty to analyze combination of *addition* and XOR seems to justify its widespread application in symmetric cryptography. The solution to the above equations with *adaptive* queries (an (α, β) is a query) was first given in [17] and subsequently improved to optimality in [18]. However, a *nontrivial* solution with a set of *batch* queries remained elusive (the problem, which was raised in [18], is elaborated in Section 2). As argued before, solution with adaptive queries is easy and less practical because of the assumption of more powerful adversary. This paper solves the equations with *batch* queries. Our algorithm solves (1) with 2^{n-2} queries where a lower

bound on the number of queries is $\frac{3}{4} \cdot 2^{n-2}$ (for all $n > 3$), i.e., the lower bound is optimal up to a constant factor of $\frac{4}{3}$. This exponential lower bound constitutes an important theoretical reference point which endorses the equations's strong resistance against DC. On the other hand, (2) has been solved with only 6 (for all $n > 2$) queries which is two more than a conjectured lower bound (note that the total number of queries is 2^{2n}) – this fact shows a major cryptographic weakness of *addition* under DC and therefore, this component should be used with caution. In practical cryptanalysis, using these results we are successful to recover the secret key of a recently proposed stream cipher Helix [8], which was a candidate for consideration in the 802.11i standard, with $2^{35.64}$ chosen plaintexts which has so far been the only CP attack on this cipher (the earlier attack was an ACP attack [17]). In view of plenty of applications of *addition* and XOR, our analyses seem suitable to evaluate cryptographic strength of many ciphers. In addition, solutions to the above equations emerge as typical tasks in many branches of mathematics and computers science such as combinatorics, Boolean algebra, computational complexity. For example, the results may be useful to solve equations involving *modular multiplication* and T-functions [12, 11]. Last but not the least, the technique used to derive all the results of this article, is purely combinatorial and easier than the traditional algebraic methods to solve multivariate polynomial equations such as Gröbner bases and its variants [1, 7].

1.1 Notation

The ith bit of an n-bit integer l is denoted by l_i (l_0 denotes the least significant bit or the 0th bit of l). The operation *addition modulo* 2^n over $\mathbb{Z}_{2^n}$ can be viewed as a binary operation over $\mathbb{Z}_2^n$ (we denote this operation by '$+$') using the bijection that maps $(l_{n-1}, \cdots, l_0) \in \mathbb{Z}_2^n$ to $l_{n-1}2^{n-1} + \cdots + l_0 2^0 \in \mathbb{Z}_{2^n}$. The symbols '$\oplus$' and '$\wedge$' denote the operations *bit-wise exclusive-or* and *bit-wise and* of two n-bit integers respectively. We will denote $a \wedge b$ by ab. Throughout the paper, $[p, q]$ denotes a set containing all integers between the integers p and q including both of them. Unless otherwise stated, n denotes a positive integer. The size of a set S is denoted by $|S|$.

2 The Problem: Solving DEA with Batch Queries

In Sect. 1, we already gave the motivation for solving the following two *differential equations of addition* (DEA) over $\mathbb{Z}_2^n$,

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

$$(x + y) \oplus (x + (y \oplus \beta)) = \gamma\,, \tag{4}$$

where x, y are the *only fixed* unknown variables. In the present context, solving (3) is understood to be solving the set of all 2^{2n} equations generated by ranging (α, β) with the corresponding γ for a fixed unknown (x, y) (for (4) the number of all equations is 2^n). Such an (α, β) is referred to as a *query*. The number of solutions satisfying all 2^{2n} equations is no more than that of any subset of the equations. Therefore, solving these 2^{2n} equations reduces the search space of the secret (x, y) to the minimum. This fact is the main motivation for dealing with all 2^{2n} equations. The most captivating question that follows immediately is that whether it is possible to solve a subset of the 2^{2n} equations and obtain the same solution as that of the entire 2^{2n} equations. Therefore, our problem is an optimization problem which focuses on minimizing the number of equations to solve (3) and (4). In cryptographic applications, optimizing the number of equations to solve DEA can be translated

into reduction of the data complexities of many attacks [17], [18]. Next, we formalize the problem for the sake of easy understanding of many results.

2.1 The Power of the Adversary

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

1. An adversary has unrestricted computational power and an infinite amount of memory.
2. An adversary submits a *set* of queries $\{(\alpha, \beta)\}$ in a *batch*, to an honest oracle[1] which computes the γ's using the fixed unknown (x, y) in (3) and returns them to the adversary. We will often refer to that fixed (x, y) as the *seed* of the oracle.

Such an oracle with seed (x, y) is viewed as a mapping $O_{xy} : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$ and defined by

$$O_{xy} = \{(\alpha, \beta, \gamma) \,|\, (\alpha, \beta) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n, \gamma = (x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta))\}\,. \tag{5}$$

An adversarial model, similar to the one described above for (3), can be constructed for (4) by setting $(\alpha, \beta) \in \{0\}^n \times \mathbb{Z}_2^n$ and the mapping $O_{xy} : \{0\}^n \times \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$.

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

2.2 Formalizing the Task: The Useful Set $\tilde{D}$ and the Solution Set $\tilde{D}$-consistent

O_{xy}, defined in (5), generates a family of mappings $\mathcal{F} = \{O_{xy} \;|\, (x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n\}$. If $D \in \mathcal{F}$ then D is called a *character set*. Note that $|D| = 2^{2n}$ and therefore, a *character set* uniquely represents a set of 2^{2n} equations if we deal with (3). The aim of the adversary is to find all (x, y) satisfying these 2^{2n} equations from a subset of the *character set* D. The set of all such satisfiable (x, y)'s is called D-satisfiable. If we work with (4) then $|D| = 2^n$.

Equivalent Task. Applying the following transformation on a *character set* D we compute $\tilde{D}$,

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

We call $\tilde{D}$ a *useful set*. Note $|\tilde{D}| = 2^{2n}$ if (3) is considered. An element $(\alpha, \beta, \tilde{\gamma}) \in \tilde{D}$ corresponds to the following equation

$$(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) \oplus \alpha \oplus \beta = \tilde{\gamma}.$$

Let $\tilde{D}$-consistent denote the set of all (x, y)'s satisfying all 2^{2n} equations corresponding to $\tilde{D}$. It can be shown that

$$D\text{-satisfiable} = \tilde{D}\text{-consistent}.$$

Therefore, the task is equivalent to determination of $\tilde{D}$-consistent from a subset of the *useful set* $\tilde{D}$. Note that there is a bijection between D and $\tilde{D}$. If we deal with (4) then $|\tilde{D}| = 2^n$.

[1] An honest oracle correctly computes γ and returns it to the adversary.

Equivalent Oracle Output. The oracle output γ on query (α, β) will be adjusted to $\tilde{\gamma} = \alpha \oplus \beta \oplus \gamma$ for easy understanding of many deductions.

Rules of the Game. Below, we describe the rules followed by the adversary who determines the set $\tilde{D}$-consistent. The essence of the whole problem is brought out in the following points.

1. The adversary starts with no information about x and y except their size n.
2. The adversary submits a set of queries (α, β)'s in a batch, irrespective of the seed (x, y). The oracle returns to the adversary the set of $\tilde{\gamma}$'s corresponding to the queries and the chosen (x, y).
3. The adversary fails if, with the submitted queries, she is unable to compute $\tilde{D}$-consistent for some $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$.

We search for an algorithm that determines $\tilde{D}$-consistent, for all $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$, with the same set of submitted queries. Furthermore, there is an additional challenge to reduce the number of required queries as much close as possible to the minimum.

3 Preliminary Work: Computing the Number of Solutions

Before embarking on designing algorithms to solve the equations, we first establish the number of all solutions for different seeds (x, y)'s, i.e., the size of $\tilde{D}$-consistent (see Sect. 2.2 for the definition). We follow a series of steps (Sect. 3.1 and 3.2) leading up to the formulation of $\tilde{D}$-consistent in Sect. 3.3. We shall heavily use the results of this section to obtain two important contributions of the paper: *(i)* lower bounds on the number of queries (described in Sect. 4) and *(ii)* the proofs of correctness of our algorithms which computes $\tilde{D}$-consistent (explained in Sect. 5). The results of this section are provided in [18]. However, to make the paper self-contained, we include them.

3.1 Step 1: Relation among Input Bits

Let $A \subseteq \tilde{D}$ where $\tilde{D}$ is a *useful set*. Take an arbitrary element $(\alpha, \beta, \tilde{\gamma}) \in A$ $(n > 1)$. Observe that $\tilde{\gamma}_{i+1}$ can be computed using only the *preceding* bits x_i, y_i, c_i, α_i, $\beta_i, \tilde{\gamma}_i$, $\forall\, i \in [0, n-2]$, from the following three equations

$$\tilde{\gamma}_{i+1} = c_{i+1} \oplus \tilde{c}_{i+1},\ c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i,\ \tilde{c}_{i+1} = \tilde{x}_i \tilde{y}_i \oplus \tilde{x}_i \tilde{c}_i \oplus \tilde{y}_i \tilde{c}_i$$

where c_i is the carry at the ith position of $(x + y)$, $\tilde{x}_i = x_i \oplus \alpha_i$, $\tilde{y}_i = y_i \oplus \beta_i$ and $\tilde{c}_i = c_i \oplus \tilde{\gamma}_i$. Table 1 lists the values of $\tilde{\gamma}_{i+1}$ as computed from all values of x_i, y_i, c_i, α_i, β_i, $\tilde{\gamma}_i$.

3.2 Step 2 : Computation of Parameters G_i, $S_{i,0}$ and $S_{i,1}$ from A

We now determine an important quantity, denoted by G_i, for nonempty $A \subseteq \tilde{D}$. In G_i, we store the ith and $(i+1)$th bits of $\tilde{\gamma}$ and the ith bit of α and β for all $(\alpha, \beta, \tilde{\gamma}) \in A$. We call G_i the ith core of A. More formally (suppose $n > 1$),

$$G_i = \{(\alpha_i, \beta_i, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}) \,|\, (\alpha, \beta, \tilde{\gamma}) \in A\}, \quad i \in [0, n-2]\,. \tag{6}$$

Table 1: The values of $\tilde{\gamma}_{i+1}$ corresponding to the values of x_i, y_i, c_i, α_i, β_i, $\tilde{\gamma}_i$. A row and a column are denoted by R(l) and Col(k)

(x_i, y_i, c_i)	$(\alpha_i,\ \beta_i, \tilde{\gamma}_i)$								
	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)	R(0)
(0,0,0)	0	0	0	1	0	1	1	1	R(1)
(1,1,1)									
(0,0,1)	0	0	1	0	1	0	1	1	R(2)
(1,1,0)									
(0,1,0)	0	1	0	0	1	1	0	1	R(3)
(1,0,1)									
(1,0,0)	0	1	1	1	0	0	0	1	R(4)
(0,1,1)									
Col(0)	Col(1)	Col(2)	Col(3)	Col(4)	Col(5)	Col(6)	Col(7)	Col(8)	

We now discuss what the expression "$G_i \Rightarrow (x_i,\ y_i,\ c_i)$" means for a known G_i. Let $|G_i| = g$. Take an element $(\alpha_i,\ \beta_i,\ \tilde{\gamma}_i,\ \tilde{\gamma}_{i+1}) \in G_i$. In Table 1, find the row(s) of the fourth coordinate $\tilde{\gamma}_{i+1}$ in the column specified by the first three coordinates $(\alpha_i,\ \beta_i,\ \tilde{\gamma}_i)$ in R(0) and put them in set F_{i1}. Find $F_{i1}, \cdots F_{ig}$ for all g elements of G_i. Let $F_i = \bigcap_j F_{ij}$ and R(x)$\in F_i$. If $(x_i,\ y_i,\ c_i)$ is in Col(0)$\times$R(x) then we say $G_i \Rightarrow (x_i,\ y_i,\ c_i)$. If $F_i = \phi$ then no such $(x_i,\ y_i,\ c_i)$ exists. We compute $S_{i,\,j} = \{(x_i,\ y_i)\,|\,G_i \Rightarrow (x_i,\ y_i,\ c_i = j)\}$. Now, we show a fundamental relation between $S_{i,\,0}$ and $S_{i,\,1}$ that will be used to obtain several results.

Proposition 1 *For all nonempty set $A \subseteq \tilde{D}$ and all $n > 1$, $|S_{i,\,0}| = |S_{i,\,1}|\ \ \forall\, i \in [0,\ n-2]$.*

We set,

$$|S_{i,\,0}| = |S_{i,\,1}| = S_i \quad \forall\, i \in [0,\ n-2]. \tag{7}$$

The example, provided below, will also ease the understanding of the subsequent results.

Example. $(G_i, S_{i,\,0}, S_{i,\,1})$ 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 (6)). Now, from Table 1, $F_{01} = \{R(1),\ R(3)\}$, $F_{02} = \{R(1),\ R(2)\}$, $F_{11} = \{R(1),\ R(4)\}$, $F_{12} = \{R(2),\ R(4)\}$, $F_{13} = \{R(1),\ R(4)\}$. Therefore, $F_0 = F_{01} \cap F_{02} = \{R(1)\}$ and $F_1 = F_{11} \cap F_{12} \cap F_{13} = \{R(4)\}$. Now $G_0 \Rightarrow (0,\ 0,\ 0)$, $G_0 \Rightarrow (1,\ 1,\ 1)$ as $(0,\ 0,\ 0)$, $(1,\ 1,\ 1)$ are in Col(0)$\times$R(1). Similarly, $G_1 \Rightarrow (1,\ 0,\ 0)$, $G_1 \Rightarrow (0,\ 1,\ 1)$. Thus, $S_{0,\,0} = \{(0,\ 0)\}$, $S_{0,\,1} = \{(1,\ 1)\}$, $S_{1,\,0} = \{(1,\ 0)\}$, $S_{1,\,1} = \{(0,\ 1)\}$. Therefore, $S_0 = S_1 = 1$. □

3.3 Step 3 (final): Formulation of the Size of $\tilde{D}$-consistent

Let $A \subseteq \tilde{D}$. The definition of A-consistent which denotes the set of all $(x,\ y)$'s satisfying the equations represented by A is a natural extension of the definition of $\tilde{D}$-consistent. The general formula for the number of solutions for any given set of DEA (the set may not contain all possible equations) is shown in the following proposition (the proof is given in Appendix A.4).

Proposition 2 *Let $A \neq \phi$ and S denote $|A$-consistent$|$. Then,*

$$S = \begin{cases} 0 & \text{if } \tilde{\gamma}_0 = 1 \text{ for some } (\alpha,\, \beta,\, \tilde{\gamma}) \in A, \\ 4 \cdot \prod_{i=0}^{n-2} S_i & \text{if } \tilde{\gamma}_0 = 0,\, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A \text{ and } n > 1, \\ 4 & \text{if } \tilde{\gamma}_0 = 0,\, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A \text{ and } n = 1. \end{cases}$$

The S_i's are defined in (7).

Now, as we are interested in the size of $|\tilde{D}$-consistent$|$ for (4) and (3), we obtain the results in Theorem 1 and Theorem 2 as special cases of Proposition 2.

Theorem 1 *Let the position of the least significant '1' of x in the equation*

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

be t and $x,\, y,\, \beta,\, \gamma \in \mathbb{Z}_2^n$. Let a useful set $\tilde{D}$ *be given. Then $|\tilde{D}$-consistent$|$ is*
(i) 2^{t+3} if $n - 2 \geq t \geq 0$,
(ii) 2^{n+1} otherwise (including the case when $x = 0$).

Proof. *(i)* $n - 2 \geq t \geq 0$. Using the procedure described in Sect. 3.2, we construct the ith core $G_i,\ \forall\, i \in [0,\, n-2]$, from the *useful set* $\tilde{D}$. We derive that

$$G_i = \{(0,\, 0,\, 0,\, a_i),\, (0,\, 1,\, 0,\, b_i)\}\,, \qquad \forall i \in [0,\, t] \tag{8}$$

$$G_i = \{(0,\, 0,\, 0,\, c_i),\, (0,\, 0,\, 1,\, d_i)\,(0,\, 1,\, 0,\, e_i),\, (0,\, 1,\, 1,\, f_i)\},\ \forall i \in [t+1,\, n-2]\,. \tag{9}$$

In the above equations, $a_i,\, b_i,\, c_i,\, d_i,\, e_i,\, f_i \in [0,\, 1]$. Note that only (8) is relevant if $t = n-2$. The fact that we are able to extract only the first three coordinates of the elements of G_i, $\forall i \in [0,\, n-2]$, can be proved using the following two auxiliary lemmas, the proofs of which are given in Appendix A.1.

Lemma 1 *For all $(0,\, \beta,\, \tilde{\gamma}) \in \tilde{D}$, $\tilde{\gamma}_i = 0,\ \forall\, i \in [0,\, t]$.*

Lemma 2 *For all $i \in [t+1,\, n-1]$ there exists $(0,\, \beta,\, \tilde{\gamma}) \in \tilde{D}$ with $\tilde{\gamma}_i = 1$.*

However, only the first three coordinates of the elements of the G_i's are sufficient to determine the S_i's (S_i is defined in Sect. 3.2). It is easy to verify from Table 1 that $S_i = 2,\, \forall i \in [0,\, t]$ and $S_i = 1,\, \forall i \in [t+1,\, n-2]$. From Proposition 2

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

(ii) The proof is similar to the above one using Proposition 2. □

Theorem 2 *Let a* useful set $\tilde{D}$ *be given for the equation*

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

with $x,\, y,\, \alpha,\, \beta,\, \gamma \in \mathbb{Z}_2^n$. Then $|\tilde{D}$-consistent$|$=4.

Proof. The proof is similar to the above proof. Details are provided in Appendix A.2. □

4 Lower Bounds on the Number of Queries

Armed with the results derived in the previous section, we are now ready to establish one of the major contributions of this article, i.e., lower bounds on the number of queries, submitted in a batch, to solve the said equations. As it is already clear from the previous discussion that the trivial method is to submit all possible queries and then solve it; the challenge lies with a *nontrivial* solution which uses less number of queries. It is always regarded as an important theoretical benchmark as to how far it is possible to reduce the number of queries. The significance of a lower bound is that no algorithm can solve the equations with queries less than it.

We already noticed that more queries tend to reduce the search space of the secret (x, y). In our formal framework, if $A \subseteq B \subseteq \tilde{D}$ then $\tilde{D}$-consistent $\subseteq B$-consistent $\subseteq A$-consistent. This implies that $|\tilde{D}\text{-consistent}| \leq |B\text{-consistent}| \leq |A\text{-consistent}|$. Note that our algorithm constructs $A \subseteq \tilde{D}$, $\forall (x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$, using the submitted queries and the corresponding outputs such that $|\tilde{D}\text{-consistent}| = |A\text{-consistent}|$. The algorithm fails if $|\tilde{D}\text{-consistent}| < |A\text{-consistent}|$, for some $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$. We will use the condition – $|A$-consistent$|$ cannot be *strictly* greater than $|\tilde{D}$-consistent$|$ – to compute a lower bound on the number of queries. In Theorem 3 we identify a property of A, in terms of the $(n-3)$th core G_{n-3}, where the above condition is violated. In Theorem 4, we use this fact to obtain a lower bound.

Theorem 3 *We consider the equation*

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

where the position of the least significant '1' of x is t with $n - 3 \geq t \geq 0$. Let all the submitted queries and the oracle outputs be stored in the set A (note that $\phi \subset A \subseteq \tilde{D}$ where $\tilde{D}$ is a useful *set). Suppose that there is no query $(0, \beta)$ for which the oracle output is $\tilde{\gamma}$ with $\tilde{\gamma}_{n-2} = 1$. Then $|A$-consistent$| > |\tilde{D}$-consistent$|$.*

Proof. If there is no query $(0, \beta)$ for which the oracle output is $\tilde{\gamma}$ with $\tilde{\gamma}_{n-2} = 1$ then the $(n-3)$th core G_{n-3} (corresponding to A) contains no element $(0, \beta_{n-3}, \tilde{\gamma}_{n-3}, \tilde{\gamma}_{n-2})$ with $\tilde{\gamma}_{n-2} = 1$. This implies G_{n-2} contains no element $(0, \beta_{n-2}, \tilde{\gamma}_{n-2}, \tilde{\gamma}_{n-1})$ with $\tilde{\gamma}_{n-2} = 1$. Therefore, G_{n-2} is of one of the following forms,

$$G_{n-2} = \{(0, 0, 0, 0)\} \quad \text{or } \{(0, 0, 0, 0), (0, 1, 0, b)\}\,.$$

Now, from Table 1, $S_{n-2} = 2^l$ for either of the cases, where $l > 0$. Similarly, using Theorem 1, $S_i \geq 2$, $\forall\, i \in [0, t]$. Also $S_i \geq 1$, $\forall\, i \in [t + 1, n - 3]$ (when $n - 4 \geq t$). Therefore, from Proposition 2, $|A\text{-consistent}| = 2^{t+3+k}$ for some $k > 0$. From Theorem 1, $|\tilde{D}\text{-consistent}| = 2^{t+3}$. Therefore, $|A\text{-consistent}| > |\tilde{D}\text{-consistent}|$. □

Instead of establishing a lower bound for the entire seed space $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$, we derive a lower bound for a subset of $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$, denoted by $V_{n,\,0}$ which is the collection of all (x, y)'s with $x_0 = 1$ (that is, the position of the least significant '1' of x is zero). Note that $|V_{n,\,0}| = 2^{2n-1}$. It is easy to conclude that the lower bound derived for $V_{n,\,0}$ is also a lower bound for the entire seed space $\mathbb{Z}_2^n \times \mathbb{Z}_2^n$.

Theorem 4 *A lower bound on the number of queries $(0, \beta)$'s, submitted in a batch, to solve*

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

where $(x, y) \in V_{n,\,0}$ is (i) $3 \cdot 2^{n-4}$ if $n \geq 4$, (ii) 2 if $n = 3$, (iii) 1 if $n = 2$ and (iv) 0 if $n = 1$.

Proof. *(i)* When $n \geq 4$. Let all the submitted queries and the oracle outputs be stored in the set A ($\phi \subset A \subseteq \tilde{D}$ where $\tilde{D}$ is a *useful* set) and $|A\text{-consistent}| = |\tilde{D}\text{-consistent}|$ for all $(x, y) \in V_{n,0}$. By Theorem 3, a *necessary* condition is that there must exist at least one query $(0, \beta)$ for which the oracle output is $\tilde{\gamma}$ with $\tilde{\gamma}_{n-2} = 1$ otherwise $|A\text{-consistent}| > |\tilde{D}\text{-consistent}|$. We shall henceforth denote a query $(0, \beta)$ by β.

We first encode the bit-string of a query β as the edges and the corresponding output $\tilde{\gamma}$ as the nodes (denoted by circles) on a path of the full binary tree as shown in Fig. 1. The possible values of β_i are denoted as the edges of the tree between the depth i and the depth $(i+1)$ (the root of the tree is at depth 0). Similarly the possible values of $\tilde{\gamma}_i$ can be assigned to the nodes at the depth i. Note that all possible 2^n queries are encoded in the tree.

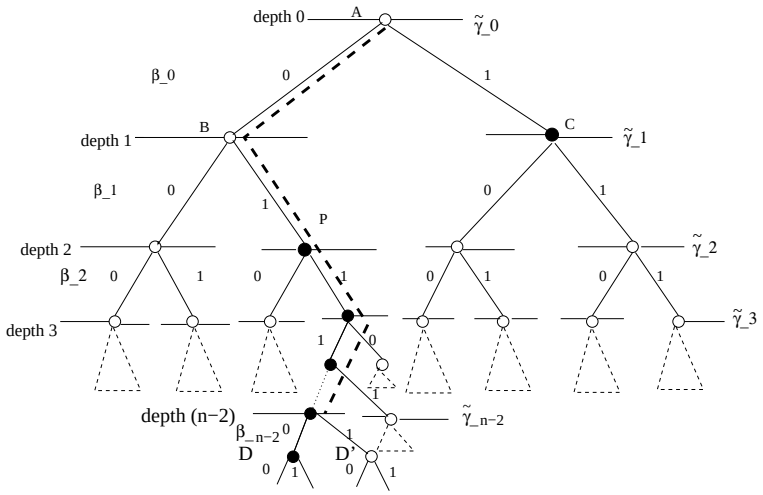

Figure 1: An arbitrary path P in the subtree (black node indicates value 1 and white node 0)

The Approach. We shall isolate a subtree and show that, if an arbitrary path in that subtree is *not* present as the prefix of one of the submitted queries then there exists a seed $(x, y) \in V_{n,0}$ such that, on all other queries, the outputs are $\tilde{\gamma}$'s with $\tilde{\gamma}_{n-2} = 0$. Therefore a lower bound is the number of all paths present in that particular subtree.

Node Assignment Rule. The rule shows how to select the values of $\tilde{\gamma}$'s for all the queries. In the nodes of the *entire* tree we now put the values of $\tilde{\gamma}$. We select an *arbitrary* path P (which is a prefix of a query) in the subtree whose leaf nodes are at the depth $(n-2)$ and whose first two edges are $(0,\ 1)$ and $(1,\ 0)$ and $(1,\ 1)$ (see Fig. 1). Note that the values at the nodes B and C will be 0 and 1 respectively because $x_0 = 1$. Now we put 1 in all nodes on the path P from the depth 2 till the depth $(n-2)$. The two child nodes D and D' of the last node on P are assigned 0 and 1 arbitrarily. All other nodes in the tree are assigned 0. The intuition that such an assignment rule gives a valid solution is derived from an observation in Table 1 (see Sect. 3.1) that the matrix cut off by rows R(1), R(2) and R(3) and columns Col(2), Col(3) and Col(4) has only diagonal elements 1 (formally proved in Lemma 3).

Proof Continued. Suppose P is not a prefix of any query in A. As shown in Fig. 1, all nodes at depth $(n-2)$, except the one on the path P, are assigned zero. Therefore, there is no query β in A such that the corresponding output $\tilde{\gamma}$ has $\tilde{\gamma}_{n-2} = 1$. This leads to a contradiction. Therefore, there

must be a query in A whose prefix is P. Now P is an arbitrary path in the subtree constructed above. Now the total number of paths (or prefixes of queries) in the subtree is $3 \cdot 2^{n-4}$. The following lemma completes the proof.

Lemma 3 *For any arbitrary P with the first two edges $(0,\ 1)$ or $(1,\ 0)$ or $(1,\ 1)$ in the tree constructed above, all queries and their outputs encoded in the tree according to the* Node Assignment Rule*, produce a valid solution $(x,\ y) \in V_{n,\,0}$.*

Proof. For any arbitrary P, the core G_i's $(0 \leq i \leq n-2)$, computed from the values of $\tilde{\gamma}$'s and β's (according to the *Node Assignment Rule*), are of one of the following forms

$$\begin{aligned} G_0 &= \{(0,\,0,\,0,\,0),\,(0,\,1,\,0,\,0)\}\,, \\ G_i &= \{(0,\,0,\,0,\,0),\,(0,\,1,\,0,\,0),\,(0,\,0,\,1,\,a_i),\,(0,\,1,\,1,\,b_i)\}\,, \qquad 1 \leq i \leq n-2 \end{aligned}$$

where $a_i,\ b_i \in [0,\,1]$ and $a_i = 1 \oplus b_i$. Now each $S_i > 0$ (obtained from Table 1 using the G_i's). Therefore, the number of valid solutions $S = 4 \cdot \prod_{i=0}^{n-2} S_i > 0$ (see Proposition 2). In fact the number solutions is 8 (verification of a part of Theorem 1). □□
The proofs of *(ii)*, *(iii)* and *(iv)* are immediate from Table 1. □

Lower Bound for the equation $(x + y) \oplus ((x \oplus \alpha) + (y \oplus \beta)) = \gamma$. For the equation lower bounds on the number of *batch* queries, for $n = 2$ and 3, are 2 and 3 respectively. This can be proved by searching through all possible $x,\ y,\ \alpha,\ \beta$ and $\tilde{\gamma}$ exhaustively. However, the situation becomes intractable when $n \geq 4$ when the number of all possible $x,\ y,\ \alpha,\ \beta$ and $\tilde{\gamma}$ for only 3 queries becomes extremely large (2^{60} for $n = 4$). We did extensive experiments with many test vectors and found that three queries were insufficient to solve the equations when $n \geq 4$. We state the following conjecture.

Conjecture 1 *A lower bound on the number of queries $(\alpha,\ \beta)$, submitted in a batch, to solve*

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

is 4 if $n \geq 4$.

5 Algorithms

In this section, we present two algorithms Algorithm 1 and Algorithm 2 to solve (4) and (3) with a set of *batch* queries. The inputs to the algorithms are the *bit-length* n, the oracle O and the Table 1. The outputs are the G_i's (defined in Sect. 3.2) computed from a set of queries and their replies. Our target is to select a subset of all possible queries such that the set of solutions derived from the G_i's is the same as $\tilde{D}$-consistent, for all $(x,\ y)$'s. Algorithm to compute the actual solution set from the set of G_i's is described in Appendix A.3. Below we discuss the motivation and correctness of the algorithms; we leave many smaller details, while the pseudocode covers all cases.

Discussion: Algorithm 1*(sketch).* The number of queries required by the algorithm is 2^{n-2} which is one fourth of all possible 2^n queries (note that a lower bound in $\frac{3}{4} \cdot 2^{n-2}$). The two *for loop*'s (in steps 8-17 and 10-13) are the most important parts of the algorithm. For easy understanding of the algorithm, let us see how the algorithm works when the position of the least significant '1', of x is zero (i.e., $x_0 = 1$). Note that we have to submit queries such that the G_i's $(0 \leq i \leq n-2)$ obtained from them correspond to $S_0 = 2$ and $S_i = 1 \forall i \in [1, n-2]$ (see Theorem 1). In the tth iteration of the

Algorithm 1 Algorithm to solve the equation $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$

Input: Oracle O, n, Table T
Output: The core G_i's

If $n \leq 0$ then exit with a comment "Invalid Input";
If $n = 1$ then return an empty set ϕ indicating that all 4 solutions are possible and exit;
$\beta = (1, 1, \cdots, 1, 1)_n$; /*The first query*/
$\tilde{\gamma} = O(\beta)$; /*Oracle output*/
$Q = \{\beta\}$ and $A = \{(0, \beta, \tilde{\gamma})\}$; /*Collecting the query and the oracle output*/
If $n = 2$ then Go to Step 20;
If $n > 3$ /*If $n = 3$, the execution automatically jumps to Step 18)*/
    For all $t \in [1, n-3]$ in increasing order
        {Initialize $Q' = \phi$
        For all $\beta \in Q$
            $\{\beta' = (1, 1, \cdots, \beta'_t = 0, \beta_{t-1}, \cdots, \beta_0)$; /*Constructing a new query*/
            $O(\beta') = \tilde{\gamma}'$; /*Oracle output*/
            $Q' = Q' \cup \{\beta'\}$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\};\}$/*Collecting the oracle output*/
        $\beta' = (1, 1, \cdots, \beta'_t = 1, 0, \cdots, 0)$; /*Constructing a new query*/
        $O(\beta') = \tilde{\gamma}'$; /*Oracle output*/
        $Q' = Q' \cup \{\beta'\}$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\}$; /*Collecting oracle output*/
        $Q = Q \cup Q';\}$
$\beta' = (1, 1, \cdots, \beta'_{n-2} = 1, 0, \cdots, 0)$; /*Constructing the last query*/
$O(\beta') = \tilde{\gamma}'$ and $A = A \cup \{(0, \beta', \tilde{\gamma}')\}$; /*Oracle output*/
Return the core G_i's for all $i \in [0, n-2]$ computed from A.

bigger loop we submit a set of queries which ensures that $G_t = \{(0, 1, 1, a), (0, 0, 1, b), (0, 1, 0, c)\}$ which implies that $S_t = 1$. The tth iteration also produces at least one output $\tilde{\gamma}$ with $\tilde{\gamma}_{t+1} = 1$ which will be used in the next loop. If there is no output with $\tilde{\gamma}_{t+1} = 1$ then $S_{t+1} > 1$ and hence the algorithm fails (see Theorem 3). The proof of correctness of the algorithm when $x_0 \neq 1$ is similar to the above argument.

Discussion: Algorithm 2 *(sketch).* The number of queries required by the algorithm is 6 which is two more than the best known lower bound (also note that the number of all possible queries is 2^{2n}). The proof of correctness of this algorithm is by showing that the submitted queries produce $S_i = 1$ for all $i \in [0, n-2]$ (Theorem 2). Six queries are submitted in steps 3, 5, 8, 10, 12, 14. Now we consider only the first two queries in steps 3 and 5. For these queries, $G_i = \{(1, 0, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (0, 1, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}$ if i even. Note that, in this case, if $\tilde{\gamma}_i = \tilde{\gamma}'_i$ then $S_i = 1$ otherwise $S_i = 2$ (from Table 1). Also observe that $G_i = \{(1, 0, \tilde{\gamma}_i, \tilde{\gamma}_{i+1}), (1, 0, \tilde{\gamma}'_i, \tilde{\gamma}'_{i+1})\}$ if i odd. In this case, if $\tilde{\gamma}_i \neq \tilde{\gamma}'_i$ then $S_i = 1$ otherwise $S_i = 2$. Next, we observe a combinatorial pattern in the precomputed Table 1 which shows that, if $S_i = 2$ then $S_{i-1} = 1$ (omitting proof). The 3rd and the 4th queries are generated from the second query assuming $S_i = 2$ for some odd i's. We change all the even numbered bits of the second query $(\alpha[2], \beta[2])$. It can be shown that making $(\alpha[2]_i, \beta[2]_i) = (0, 0)$ and $(1, 1)$ for all even i's ensures that $S_i = 1$ for all odd i's. Exactly the same way the 5th and the 6th queries are generated from the second query $(\alpha[2], \beta[2])$ assuming that $S_i = 2$ for some even i's. Now we change the odd numbered bits of $(\alpha[2], \beta[2])$ to $(0, 0)$ and $(1, 1)$ to ensure that all $S_i = 1$ for all even i's. Therefore, the number of solutions derived from these 6 queries is $S = 4 \cdot \prod_{i=0}^{n-2} S_i = 4$ as suggested in Theorem 2.

Algorithm 2 Algorithm to solve the equation $(x + y) \oplus ((x + \alpha) + (y \oplus \beta)) = \gamma$

Input: Oracle O, n, Table T
Output: the core G_i's

1: If $n \leq 0$ then exit with a comment "Invalid Input";
2: If $n = 1$ then return an empty set ϕ indicating that all 4 solutions are possible and exit;
3: $(\alpha[1], \beta[1]) = ((11 \cdots 11)_n, (00 \cdots 00)_n)$;/*First Query*/
4: $\tilde{\gamma}[1] = O(\alpha[1], \beta[1])$; /*Oracle Output*/
5: $(\alpha[2], \beta[2]) = ((\cdots 101010)_n, (\cdots 010101)_n)$;/*Second Query*/
6: $\tilde{\gamma}[2] = O(\alpha[2], \beta[2])$; /*Oracle Output*/
7: If $n = 2$ then Go to Step 16;
8: $(\alpha[3], \beta[3]) = ((\cdots 101010)_n, (\cdots 000000)_n)$;/*Third Query*/
9: $\tilde{\gamma}[3] = O(\alpha[3], \beta[3])$; /*Oracle Output*/
10: $(\alpha[4], \beta[4]) = ((\cdots 111111)_n, (\cdots 010101)_n)$;/*Fourth Query*/
11: $\tilde{\gamma}[4] = O(\alpha[4], \beta[4])$; /*Oracle Output*/
12: $(\alpha[5], \beta[5]) = ((\cdots 000000)_n, (\cdots 010101)_n)$;/*Fifth Query*/
13: $\tilde{\gamma}[5] = O(\alpha[5], \beta[5])$; /*Oracle Output*/
14: $(\alpha[6], \beta[6]) = ((\cdots 101010)_n, (\cdots 111111)_n)$;/*Sixth Query*/
15: $\tilde{\gamma}[6] = O(\alpha[6], \beta[6])$; /*Oracle Output*/
16: $A = \{(\alpha[i], \beta[i], \tilde{\gamma}[i]) \mid \text{ for all } i\text{'s}\}$
17: Return the core G_i's for all $i \in [0, n-2]$ computed from A.

6 Cryptographic Applications

Strength of modular addition against DC. In view of the large scale application of modular addition in symmetric cryptography, our results seem effective in the evaluation of security of cipher components which mixes two different group operations addition and XOR. The fact that only 6 *queries* submitted in a batch (which is a more practical attack scenario than adaptive queries) are sufficient to reduce the search space of the secret (x, y) from 2^{2n} to only 4 (for all $n \geq 4$) should be recognized as a warning to the designers (see Algorithm 2). On the other hand the high exponential lower bound on the number of queries for another differential equation of addition ($3 \cdot 2^{n-4}$ for all $n \geq 4$) underlines an important theoretical reference point which advocates it as *relatively stronger* under DC (see Theorem 4). One direct application of our results in practical cryptanalysis is described below. At this moment, we are not aware of other applications of the results, yet it can very likely be used to evaluate cryptographic strengths of many modern ciphers which use modular multiplication combined with modular addition and XOR.

Cryptanalysis of Helix. Helix, proposed by Ferguson *et al.* [8], is a stream cipher with a combined MAC functionality. This cipher was a candidate for consideration in the 802.11i standard. The main component of the primitive is combination of *addition* and XOR. The fact that the internal state of Helix depends on the plaintext allows for cryptanalysis with *chosen plaintexts* (CP) and *adaptive chosen plaintext* (ACP). Muller mounted an ACP attack which recovers the secret key of the Helix cipher with 2^{12} plaintexts. We refer the readers to [17] for a detailed analysis of the attack. Our key recovery attack goes on the same line as Muller's attack. We cannot describe the attack in full detail because of space constraints, however, we pick out a portion which is critical to our CP attack. The crux of the whole attack is solving the equation $(x + y) \oplus (x + (y \oplus \beta)) = \gamma$ with β's and the corresponding γ's to recover the secret information (x, y), in the framework described in Sect. 2.1, for

50 times ($n = 32$ for the Helix cipher). Every time β corresponds to a CP. Algorithm 1 shows that the above equation can be solved with 2^{n-2} CP's (2^{30} for $n = 32$). Therefore, the total data complexity of our CP attack is $50 \cdot 2^{30}$, i.e., $2^{35.64}$ plaintexts.

Note. Apparently, from the quantities of the required plaintexts, it may look that our attack is less effective than Muller's attack. However, comparison between a CP and an ACP attack in terms of data complexities is unjustified. On the other hand, the working principles a CP attack makes it more practical than an ACP attack. It was not known till now how to mount a CP attack on the Helix cipher and its data complexity.

7 Conclusion and Further Research

The results of the paper contribute to both theory and practice. We showed a lower bound on the number of *batch* queries to solve a DEA which is optimal up to a constant factor. For solving another DEA, our algorithm uses number of queries which is constant asymptotically. Our results are used directly to recover the key of the Helix cipher with chosen plaintexts rather than adaptive chosen plaintexts which has so far been the best CP attack on this cipher. The paper also leaves many interesting questions open. One possible research direction may be to close the gap between the lower and upper bounds on the number of queries to solve DEA. Another way to extend the work is to analyze components which combine more complex transformations such as modular multiplication, T-functions with addition.

References

[1] I. A. Ajwa, Z. Liu, P. S. Wang, "Gröbner Bases Algorithm," *ICM Technical Report*, February 1995, Available Online at `http://icm.mcs.kent.edu/reports/1995/gb.pdf`.

[2] T. A. Berson, "Differential Cryptanalysis Mod 2^{32} with Applications to MD5," *Eurocrypt 1992* (R. A. Rueppel, ed.), vol. 658 of *LNCS*, pp. 71-80, Springer-Verlag, 1993.

[3] 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.

[4] 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.

[5] A. Biryukov, D. Wagner, "Slide Attacks," *Fast Software Encryption 1999*, (Lars R. Knudsen, ed.), vol. 1636 of *LNCS*, pp. 245-259, Springer-Verlag, 1999.

[6] N. Courtois, J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations," *Asiacrypt 2002* (Yuliang Zheng, ed.), vol. of *LNCS*, pp. 267-287, Springer-Verlag, 2002.

[7] J. Faugère, "A new effecient algorithm for computing Gröbner bases (F_4)," *Journal of Pure and Applied Algebra*, vol. 139, pp. 61-88, 1999, Available Online at `http://www.elsevier.com/locate/jpaa`.

[8] 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.

[9] M. E. Hellman, "A Cryptanalytic Time-Memory Trade-off," IEEE Transaction on Information Theory, vol. IT-26, No. 4, July, 1980.

[10] D. E. Knuth, *"The Art of Computer Programming," vol. 2, Seminumerical Algorithms,* Addison-Wesley Publishing Company, 1981.

[11] 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.

[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. Kipnis, A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystems by Relinearization," *Crypto 1999* (M. Wiener, ed.), vol. 1666 of *LNCS*, pp. 19-30, Springer-Verlag, 1999.

[14] 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.

[15] 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.

[16] L. Lipmaa, J. Wallén, P. Dumas, "On the Additive Differential Probability of Exclusive-Or," *Fast Software Encryption 2004* (B. Roy, W. Meier, eds.), vol. 3017 of *LNCS*, pp. 317-331, Springer-Verlag, 2004.

[17] 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.

[18] S. Paul and B. Preneel, "Solving Systems of Differential Equations of Addition (Extended Abstract)," *10th Australasian Conference on Information Security and Privacy, ACISP 2005* (Colin Boyd and Juan Gonzalez, eds.), vol. 3574 of *LNCS*, pp. 75-88, Springer-Verlag, 2005, Extended Version available online on IACR ePrint Archive as Report 2004/294 at `http://eprint.iacr.org/2004/294`, April 2005.

[19] 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.

[20] 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.

[21] 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.

[22] D. Wagner, "The Boomerang Attack," *Fast Software Encryption 1999*, (Lars R. Knudsen, ed.), vol. 1636 of *LNCS*, pp. 156-170, Springer-Verlag, 1999.

[23] J. Wallén, "Linear Approximations of Addition Modulo 2^n," *Fast Software Encryption 2003* (T. Johansson, ed.), vol. 2887 of *LNCS*, pp. 261-273, Springer-Verlag, 2003.

A Appendix

A.1 Proofs of Lemma 1 and 2

Claim 1 *For all* $(0, \beta, \tilde{\gamma}) \in \tilde{D}$, $\tilde{\gamma}_i = 0 \; \forall\, i \in [0, t]$.

Proof. If the position of the least significant '1' of x is t then $c_i = \tilde{c}_i = 0 \; \forall\, i \in [0, t]$ and $\forall\, \beta \in \mathbb{Z}_2^n$. Recall $\tilde{\gamma}_i = c_i \oplus \tilde{c}_i$. This proves the lemma. □

Claim 2 *For each* $i \in [t+1, n-1]$, *there exists* $(0, \beta, \tilde{\gamma}) \in \tilde{D}$ *with* $\tilde{\gamma}_i = 1$.

Proof. We prove the lemma by induction on i. The statement is true if $i = t+1$. Suppose, the statement is true if $i = k$ for some $k \in [t+1, n-2]$, that is, there exists $(0, a, b) \in \tilde{D}$ with $b_k = 1$ (induction hypothesis). 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 all $p \in \mathbb{Z}_2^n$, 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. □

A.2 Number of Solutions for (3)

Claim 3 *Let a* useful set $\tilde{D}$ *be given for the equation*

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

with $x, y, \alpha, \beta, \gamma \in \mathbb{Z}_2^n$. *Then* $|\tilde{D}$*-consistent*$|=4$.

Proof. Our approach is same as that of Theorem 1.
Case 1: $n \geq 2$**.** Corresponding to the *useful set* $\tilde{D}$ we determine G_i, $\forall\, i \in [0, n-2]$.

$$\begin{aligned} G_0 &= \{(0, 0, 0, a_0), (0, 1, 0, b_0), (1, 0, 0, c_0), (1, 1, 0, d_0)\}, && (10) \\ G_i &= \{(0, 0, 0, e_i), (0, 0, 1, f_i), (0, 1, 0, g_i), (0, 1, 1, h_i), \\ &\quad (1, 0, 0, m_i), (1, 0, 1, n_i), (1, 1, 0, p_i), (1, 1, 1, q_i)\}, \forall\, i \in [1, n-2]\,. && (11) \end{aligned}$$

In the equations $a_0, b_0, c_0, d_0, e_i, f_i, g_i, h_i, m_i, n_i, p_i, q_i \in [0, 1]$. From Table 1 we see that $S_i = 1$, $\forall\, i \in [0, n-2]$ (see Sect. 3.2 to compute S_i from G_i). Therefore, from Proposition 2

$$|\tilde{D}\text{-consistent}| = S = 4 \cdot \prod_{i=0}^{n-2} S_i = 4 \cdot \underbrace{1 \cdot 1 \cdots 1}_{(n-1)\text{ times}} = 4\,.$$

Case 2: $n = 1$**.** If $n = 1$ then $|\tilde{D}$-consistent$| = 4$. The proof is trivial using Proposition 2. □

A.3 Solving Systems of Equations from the G_i's

In Algorithm 3, we omitted details of many operations for easy understanding. The detailed analysis is provided separately. The correctness proof of the algorithm is the following theorem.

Theorem 5 *Let $\phi \subseteq A \subseteq \tilde{D}$ and $n > 1$. The following two statements are equivalent.*
1. $(x, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ is such that $G_i \Rightarrow (x_i, y_i, c_i)$, $\forall\, i \in [0, n-2]$.
2. $(x, y) \in A$-consistent.

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

Algorithm 3 Computing all solutions to a system of DEA from the G_i's

Input: G_i, $\forall i \in [0, n-2]$
Output: $\tilde{D}$-consistent
1: $L_i = \{(x_i, y_i, c_i) \,|\, G_i \Rightarrow (x_i, y_i, c_i)\}$ for all $i \in [0, n-2]$
2: Compute $M = \{((x_{n-1}, x_{n-2}, \cdots, x_0), (y_{n-1}, y_{n-2}, \cdots, y_0)) \,|\, (x_{n-1}, y_{n-1}) \in \mathbb{Z}_2^2,$
$(x_i, y_i, c_i) \in L_i,\ i \in [0, n-2],\ c_0 = 0, c_{i+1} = x_i y_i \oplus x_i c_i \oplus y_i c_i\}$.
3: Return (M).

Detailed Construction. Let G_i be known $\forall\, i \in [0, n-2]$ $(n > 1)$ for $A \subseteq \tilde{D}$ where $\tilde{D}$ is a *useful* set. Let $L_i = \{(x_i, y_i, c_i) \,|\, G_i \Rightarrow (x_i, y_i, c_i)\}\, \forall i \in [0, n-2]$. Let a set M be constructed from the L_i's in the following way,

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

We present an algorithm to construct M from the L_i's with memory $\mathcal{O}(n \cdot S)$ and time $\mathcal{O}(S)$ where $S = |\tilde{D}\text{-consistent}| = |M|$.

First we set

$$M_1 = \{((c_1), (x_0), (y_0)) | (x_0, y_0, 0) \in L_0,\ c_1 = x_0 y_0\}\,.$$

Now we construct a set M_k $\forall\, k \in [2, n-1]$ using the following recursion.

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

Now, we construct

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

Using Proposition 2 and Theorem 5 it is easy to show that $M = M_n$. Note that the size of each L_i is $\mathcal{O}(1)$ since the size of the Table 1 is $\mathcal{O}(1)$. Also note that $|M_n| = S$ and therefore the asymptotic memory requirement to construct M_n recursively following the above algorithm is $\mathcal{O}(n \cdot S)$ since $k = \mathcal{O}(n)$ and M_{k+1} can be constructed from M_k only. It is trivial to show that the time to construct M_n (i.e., M) from the L_i's is $\mathcal{O}(S)$. Thus, the set M can be constructed from the L_i's with memory $\mathcal{O}(n \cdot S)$ and time $\mathcal{O}(S)$.

A.4 Determination of |A-consistent|

Claim 4 *Let $A \neq \phi$ and S denote |A-consistent|. Then,*

$$S = \begin{cases} 0 & \text{if } \tilde{\gamma}_0 = 1 \text{ for some } (\alpha,\, \beta,\, \tilde{\gamma}) \in A, \\ 4 \cdot \prod_{i=0}^{n-2} S_i & \text{if } \tilde{\gamma}_0 = 0,\, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A \text{ and } n > 1, \\ 4 & \text{if } \tilde{\gamma}_0 = 0,\, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A \text{ and } n = 1. \end{cases}$$

The S_i's are defined in (7).

Proof. We prove this result by considering all cases individually.

Case 1 If $\tilde{\gamma}_0 = 1$ for some $(\alpha,\, \beta,\, \tilde{\gamma}) \in A$ then $c_0 = 1$ which is impossible.

Case 2 If $\tilde{\gamma}_0 = 0, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A$ and $n > 1$. From Theorem 5, S is the number of solutions $(x,\, y) \in \mathbb{Z}_2^n \times \mathbb{Z}_2^n$ such that $G_i \Rightarrow (x_i,\, y_i,\, c_i), \forall i \in [0,\, n-2]$. Let M_k denote the number of solutions for $((x_k,\, \cdots, x_0),\, (y_k,\, \cdots, y_0))$ such that $G_i \Rightarrow (x_i,\, y_i,\, c_i), \forall i \in [0,\, k]$ where $k \in [0,\, n-2]$. Note that G_k depends only on the least $(k+1)$ bits of x, y, α, β. We consider two subcases.
Case 2(a): $n > 2$. We determine |A-consistent| recursively. Let $M_l = M_{l,\,0} + M_{l,\,1}$ such that $M_{l,\,0}$ solutions produce $c_{l+1} = 0$ and $M_{l,\,1}$ solutions produce $c_{l+1} = 1$. Therefore, $\forall\, l \in [0,\, n-3]$

$$\begin{aligned} M_{l+1} &= M_{l,\,0} \cdot |S_{l+1,\,0}| + M_{l,\,1} \cdot |S_{l+1,\,1}| \\ &= S_{l+1} \cdot M_l\,. \end{aligned} \tag{12}$$

as $|S_{i,\,0}| = |S_{i,\,1}| = S_i, \forall\, i \in [0,\, n-2]$ (see Proposition 1). It is easy to show (a proof is by contradiction) that M_{l+1}, so calculated, gives the number of solutions for $((x_{l+1},\, \cdots, x_0), (y_{l+1},\, \cdots, y_0))$ such that $G_i \Rightarrow (x_i,\, y_i,\, c_i), \forall i \in [0,\, l+1]$. From (12),

$$M_{n-2} = \prod_{i=0}^{n-2} S_i \tag{13}$$

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

$$S = 4 \cdot M_{n-2} = 4 \cdot \prod_{i=0}^{n-2} S_i \quad \text{if } n > 2\,. \tag{14}$$

Case 2(b): $n = 2$. It is easy to show that $S = 4 \cdot S_0$ if $n = 2$.

Case 3 If $\tilde{\gamma}_0 = 0, \forall (\alpha,\, \beta,\, \tilde{\gamma}) \in A$ and $n = 1$. 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 $(x_{n-1},\, y_{n-1})$. □