Another View on Cube Attack, Cube Tester, AIDA and Higher Order Differential Cryptanalysis

Bo Zhu[1], Guang Gong[1], Xuejia Lai[2] and Kefei Chen[2]

[1] Department of Electrical and Computer Engineering,
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
{bo.zhu,ggong}@uwaterloo.ca
[2] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
{laix,kfchen}@sjtu.edu.cn

Abstract. This paper introduces a more in-depth cryptanalysis framework for tweakable cryptosystems than Cube Attack, Cube Tester, algebraic IV differential attack (AIDA), and higher order differential cryptanalysis in Boolean algebra. Through the view of the new framework, the differences among these existing analysis methods are clarified. Furthermore, a principle is proposed to design secure systems against such cryptanalyses.

Key words: algebraic IV differential attack, Boolean function, Cube Attack, Cube Tester, higher order differential cryptanalysis, tweakable cryptosystem

1 Introduction

Due to the fast developments of RFID and Wireless Sensor Networks, a large amount of novel security algorithms, e.g., blockciphers and Message Authentication Codes, are designed to fit such resource-restrained environments, because normally designed schemes may need a lot of storage and memory, especially when they need to speed up computations. The threshold consideration in the environments of RFID and sensor network is minimizing the consumption of storage and the footprint of hardware implementation. Such demands lead cryptographers to devise more compact and dedicated algorithms without undermining security levels. Consider, for example, the design of PRESENT blockcipher [1]. The round function of PRESENT consists of a bit-pattern permutation layer and several small substitution-boxes, which are easily and cost-effective to be realized in both hardware and software. In addition, a newly proposed KATAN & KTANTAN blockcipher family [2] heavily relies on the computations of Boolean functions. What we can see here is that, after achieving a certain level of security, cryptographers gradually concentrate on or dive into Boolean algebra to construct more compact and efficient designs, not to mention the recall of the eSTREAM project [3].

With the development of Boolean algebra based algorithms, a lot of cryptanalysis approaches of Boolean functions are proposed to provide more profound and detailed perspective to analyze and design cryptosystems. At Eurocrypt'09, Dinur and Shamir published their paper about Cube Attack [4], which can treat any cryptographic scheme as black-box Boolean functions and discover its potential security flaws. Later at FSE'09, Aumasson and Meier, together with the two authors of Cube Attack, published a new type of attack called Cube Tester [5], in which property testing algorithms are used to detect non-randomness of Boolean functions. Nevertheless, the confusion does exist that how to define the contributions of Cube Attack and Cube Tester, compared with two other kinds of cryptanalysis methods – algebraic IV differential attack (AIDA) proposed by Vielhabor [6, 7] in 2007 and higher order differential cryptanalysis mentioned by Lai [8] and Knudsen [9] in 1990s. The fundamental ideas of these attacks are very similar, i.e. collecting many chosen plaintexts and summing up their corresponding ciphertexts to gain certain properties for the ease of attacks.

The term *tweakable* is mentioned in [4], and also used in [5], to describe the cryptosystems whose input variables can be divided into two kinds: some variables are public and can be changed or fixed during cryptanalysis (so-called tweakable, e.g., plaintexts); the other variables are secret and cannot be altered when the cryptosystems are working on line (e.g., secret keys). The goal of attacking tweakable cryptosystems is to extract the values of the secret variables, or more generally to distinguish the systems from truly random functions or permutations. Almost all cryptosystems, including public-key systems, can be seen as tweakable systems. The practical attack procedures on such systems include chosen plaintext attack (on blockciphers), chosen IV attack (on stream ciphers), and chosen message attack (on MACs).

Our Contributions. We first give a comprehensive definition of tweakable cryptosystems, by following which we point out an inappropriate part in the analysis of Cube Tester. In the second place, a systematic and profound cryptanalysis framework for tweakable systems is provided, which is more in-depth than foregoing analysis methods, such as Cube Attack, Cube Tester, AIDA, and higher order differential cryptanalysis in Boolean algebra. We also clarify the differences among these methods for the first time. Thirdly, a new principle for cryptosystems to resist such attacks is proposed, which can serve as the guideline and measurement to design secure systems.

Organization. The next section is the core part of this paper, which includes the new definition of tweakable cryptosystems, the more comprehensive cryptanalysis of tweakable systems, and the principle for tweakable cryptosystem design. In Section 3, the relationships of the new proposed cryptanalysis with existing methods are well investigated, and the differences among these methods are also clarified. Section 4 discusses the practical procedures to perform such cryptanalysis. Section 5 concludes the paper and points out several potential aspects for further works.

2 Definitions, Lemmas and Theorem

In this section, we will first give a comprehensive definition of tweakable cryptosystems for later use. And a more theoretically advanced attack on tweakable systems is well investigated, along with which several mathematical theorems are given and proved. Finally, a principle to design tweakable cryptosystems is proposed, which can serve as the measurement or guideline of designing systems with high-level security.

2.1 A Comprehensive Definition of Tweakable Cryptosystems

Almost every tweakable cryptosystem can be represented as

$$F_\kappa(\chi) : V_r \times V_s \to V_t \ ,$$

where $\kappa = (k_1, k_2, \cdots, k_r) \in V_r$ is the vector consisting of all the variables that cannot be changed during cryptanalysis (secret variables, e.g., secret key bits), and $\chi = (x_1, x_2, \cdots, x_s) \in V_s$ is the vector of all the tweakable variables (public variables, e.g., plaintext bits). Because every output variable is definitely determined by the inputs κ and χ, we can write each of the output variables as a function of κ and χ, as follows.

$$F_\kappa(\chi) = (f_{1,\kappa}(\chi), f_{2,\kappa}(\chi), \cdots, f_{t,\kappa}(\chi))$$

If we treat every bit of input and output as an individual variable, which implies V_r, V_s and V_t are over the field $\mathbb{F}_2 = \{0, 1\}$, we possibly get the most detailed representations of the cryptosystem. Each component function of $F_\kappa(\chi)$ can be written as a Boolean function in Algebraic Normal Form (ANF for short, which consists of only two kinds of operations, XOR and AND).

$$f_{i,\kappa}(\chi) = f_{i,\kappa}(x_1, x_2, \cdots, x_s) = \bigoplus_{\alpha \in V_s} c_{i,\alpha}(\kappa) x_1^{a_1} x_2^{a_2} \cdots x_s^{a_s} \ , \quad (1)$$

where $\alpha = (a_1, a_2, \cdots, a_s)$. Here every $c_{i,\alpha}(\kappa)$ is a Boolean function with respect to κ. When the system is online and the secret key κ is fixed, the value of each $c_{i,\alpha}(\kappa)$ is fixed too, that is either 0 or 1, which serves as a coefficient of the Boolean function $f_{i,\kappa}(x_1, x_2, \cdots, x_s)$.

Attacks on tweakable systems are generally considered to include two different phases, *offline phase* (so-called preprocessing) and *online phase*. During offline phase, adversaries can use both public variables (which are tweakable) and secret variables (fixed later in online phase), to discover the inner security shortcomings of the systems. For example, adversaries can take advantage of the detailed specification of a scheme, or even treat the whole algorithm as a black box and take some experiments with it. Such analysis or computations in offline phase may only need to be performed once as long as enough information about the structures of the schemes is gotten. In Section 4, we will talk about the procedures of offline phase detailedly.

What we want to emphasize here are the rules to analyze cryptosystems during online phase, which can also, in turn, determine the analysis methods used in offline phase. After clarifying the structures of the cryptosystems in offline phase, the information is used in online phase to extract the values of secret variables or distinguish the algorithms from true randomness. During online phase, adversaries are allowed to choose, change, or fix each of the public variables, but are unable to alter the values of secret variables. In addition, adversaries may be able to know or fix part of secret variables. The most significant difference of this definition with the foregoing ones is the additional notion that fixing certain secret variables (choosing one and fixing it, which are still unalterable) is allowed, which is a very natural requirement in practice and can be seen as a situation of related-key attacks. The protections against this additional condition may be able to guarantee that leaking partial secret information will not significantly affect the overall security of the cryptosystems.

2.2 Cryptanalysis of Tweakable Cryptosystems

In this subsection, a mathematical theorem is given, which forms the solid foundation for our cryptanalysis. We will illustrate and prove it step by step.

Here we focus on only one component function of the tweakable system $F_\kappa(\chi)$, just as the function (1). For the ease of description, if $\alpha = (a_1, a_2, \cdots, a_n)$ and $\beta = (b_1, b_2, \cdots, b_n) \in V_n$, and $a_i \le b_i$ holds for any i, we define $\alpha \trianglelefteq \beta$ (assuming $0 < 1$ in $\mathbb{F}_2$). And $wt(\cdot)$ denotes the hamming weight of a vector.

Theorem 1. *Given a keyed Boolean function,*

$$f_\kappa(\chi) = f_\kappa(x_1, x_2, \cdots, x_n) = \bigoplus_{\alpha \in V_n} c_\alpha(\kappa) x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \ , \qquad (2)$$

where κ is fixed (but unknown), for any chosen pair of vectors $\omega, \upsilon \in V_n$, where $\omega \trianglelefteq \upsilon$, we can get the value of

$$\bigoplus_{\alpha \,:\, \omega \trianglelefteq \alpha \trianglelefteq \upsilon} c_\alpha(\kappa) \ ,$$

by choosing $2^{wt(\omega)}$ inputs and summing up their corresponding outputs of $f_\kappa(\chi)$.

We will explain this theorem from simpler cases, and then give a complete proof at the end of this subsection.

Lemma 1. *In the equation (2), if there is a monomial $x_{i_1} x_{i_2} \cdots x_{i_m}$ that is not the sub-monomial of any other one, then we can calculate the coefficient of $x_{i_1} x_{i_2} \cdots x_{i_m}$ by choosing 2^m inputs and summing up their corresponding outputs of $f_\kappa(\chi)$.*

A more mathematical expression of this lemma is that, if there exists a vector $\beta \in V_n$ such that $c_\alpha(\kappa) = 0$ holds for any α, where $\beta \trianglelefteq \alpha \in V_n$ but $\alpha \ne \beta$, then we can calculate $c_\beta(\kappa)$ by using $2^{wt(\beta)}$ chosen inputs. It is easy to see that Lemma 1 is a special case of Theorem 1, when $\omega = \beta$ and certain coefficients are always zeros. And later in Section 3.1, we will show that Lemma 1 is exactly the same as the theoretical foundation of Cube Attack.

To prove this lemma, we first introduce a convenient tool to analyze Boolean functions, called *bitwise higher order differential cryptanalysis* [10], which is an application of higher order differential cryptanalysis [8] in Boolean algebra.

Definition 1 (Bitwise Derivative [10]). *For a Boolean function $f(x_1, x_2, \cdots, x_n)$, the* bitwise derivative *of f with respect to the variable x_m is defined as*

$$\delta_{x_m} f = f_{x_m=0} \oplus f_{x_m=1} \ .$$

The 0-th bitwise derivative is defined to be f itself. The i-th, where $i \ge 2$, bitwise derivative with respect to the variable sequence $(x_{m_1}, x_{m_2}, \cdots, x_{m_i})$ is defined as

$$\delta^{(i)}_{x_{m_1}, x_{m_2}, \cdots, x_{m_i}} f = \delta_{x_{m_i}} (\delta^{(i-1)}_{x_{m_1}, x_{m_2}, \cdots, x_{m_{i-1}}} f) \ .$$

Two important properties of bitwise derivative:

1. The result is independent of the ordering of the variable sequence. For instance, it is easy to see that
$$\delta^{(2)}_{x_1,x_2} f = \delta^{(2)}_{x_2,x_1} f \ .$$
2. The variable x_m will never appear in the Boolean function $\delta_{x_m} f$. More specifically, when calculating $\delta_{x_m} f$, the variable x_m in the monomials of f will be removed, and the monomials that do not consist of x_m will be eliminated directly. Take the Boolean function $f(x_1, x_2, x_3) = x_1x_2 \oplus x_2x_3$ for example.
$$\begin{aligned}\delta_{x_1} f(x_1, x_2, x_3) &= (0 \cdot x_2 \oplus x_2x_3) \oplus (1 \cdot x_2 \oplus x_2x_3)\\ &= (0 \cdot x_2 \oplus 1 \cdot x_2) \oplus (x_2x_3 \oplus x_2x_3)\\ &= x_2 \oplus 0\end{aligned}$$

Now we can prove Lemma 1 by using the power of bitwise higher order differential cryptanalysis.

Proof of Lemma 1. When calculating the m-th bitwise derivative of $f_\kappa(\chi)$ with respect to the public variable sequence $(x_{i_1}, x_{i_2}, \cdots, x_{i_m})$, i.e.
$$\delta^{(m)}_{x_{i_1},x_{i_2},\cdots,x_{i_m}} f_\kappa(x_1, x_2, \cdots, x_n) \ , \tag{3}$$
each of the public variables x_{i_1}, x_{i_2}, $\cdots$, and x_{i_m} in the monomial $x_{i_1}x_{i_2}\cdots x_{i_m}$ will be removed and only the coefficient will be left. In addition, because the monomial $x_{i_1}x_{i_2}\cdots x_{i_m}$ is not the sub-monomial of any other one in the Boolean function (2), every other monomial misses at least one variable from x_{i_1}, x_{i_2}, $\cdots$, and x_{i_m}, and thus these monomials will be eliminated directly when computing (3). Therefore, the function (3) is equal to the coefficient of the monomial $x_{i_1}x_{i_2}\cdots x_{i_m}$.

Without loss of generality, assume $x_{i_1}x_{i_2}\cdots x_{i_m} = x_1x_2\cdots x_m$. To calculate $\delta^{(m)}_{x_1,x_2,\cdots,x_m} f_\kappa(\chi)$, we need to collect 2^m chosen inputs of the form
$$(*_1, *_2, \cdots, *_m, x_{m+1}, x_{m+2}, \cdots, x_n) \ ,$$
where $*$ means both 0 and 1 must be chosen, and sum up their corresponding outputs of $f_\kappa(\chi)$. The rest variables can be either 0 or 1, but have to be fixed. □

Higher order differential cryptanalysis is a kind of chosen-input attacks. The next lemma will show how to gain more in-depth properties of Boolean functions by using a further chosen-input method.

Lemma 2. *In the keyed Boolean function (2), we can get the value of $c_\beta(\kappa)$, where $\beta \in V_n$, by choosing $2^{wt(\beta)}$ inputs and summing up their corresponding outputs of $f_\kappa(\chi)$.*

Proof. By setting some variables of a Boolean function to be zeros, we can get a *sub-function* of the original function. Here the term sub-function describes a function which contains part of the monomials in the original function and the coefficients in the new function are exactly the same as the original ones.

Without loss of generality, assume
$$\beta = (1_1, 1_2, \cdots, 1_m, 0_{m+1}, 0_{m+2}, \cdots, 0_n) \ ,$$
where the subscripts denote the positions in the vector. By setting the variables $x_{m+1}, x_{m+2}, \cdots, x_n$ to be zeros, we can get the sub-function $f'_\kappa(\chi)$ as
$$f'_\kappa(x_1, x_2, \cdots, x_m) = f_\kappa(x_1, x_2, \cdots, x_m, 0_{m+1}, 0_{m+2}, \cdots, 0_n) \ .$$
Because the monomial $x_1x_2\cdots x_m$ must have the highest algebraic degree in the sub-function f'_κ, by following Lemma 1, we can get the coefficient of $x_1x_2\cdots x_m$ by calculating the m-th bitwise derivative of f'_κ with respect to the variable sequence $(x_1, x_2, \cdots, x_m)$, i.e.
$$\begin{aligned}c_\beta(\chi) &= \delta_{x_1,x_2,\cdots,x_m} f'_\kappa(x_1, x_2, \cdots, x_m)\\ &= \delta_{x_1,x_2,\cdots,x_m} f_\kappa(x_1, x_2, \cdots, x_m, 0_{m+1}, 0_{m+2}, \cdots, 0_n) \ .\end{aligned}$$
To compute this bitwise derivative, we need to collect 2^m chosen inputs of the form
$$(*_1, *_2, \cdots, *_m, 0_{m+1}, 0_{m+2}, \cdots, 0_n) \ ,$$
and sum up their corresponding outputs of $f_\kappa(\chi)$. □

Lemma 2 is the special case of Theorem 1, when $\omega = \upsilon = \beta$. Later in Section 3.3, we will show that Lemma 2 is theoretically the same as the fundamental principle of AIDA.

After the analysis of Lemma 1 and Lemma 2, it is much easier to explain and prove Theorem 1. In the proof of Lemma 2, we set some variables to be zeros to get a sub-function. Now we would like to show what we can get if some variables are set to be ones, and some of other variables are set to be zeros.

Proof of Theorem 1. Fixing certain variables in the function (2) to be ones, and some other variables to be zeros, we can get a small Boolean function, which consists of part of the variables but may not be a sub-function of the original function. The coefficients of the newly obtained, small function are the summations of certain coefficients in the original function.

Without loss of generality, assume

$$\begin{cases} \omega = (1_1, 1_2, \cdots, 1_i, 0_{i+1}, 0_{i+2}, \cdots, 0_j, 0_{j+1}, 0_{j+2}, \cdots, 0_n) \\ \upsilon = (1_1, 1_2, \cdots, 1_i, 1_{i+1}, 1_{i+2}, \cdots, 1_j, 0_{j+1}, 0_{j+2}, \cdots, 0_n) \end{cases} ,$$

where $i \leq j$. By setting the variables $x_{j+1}, x_{j+2}, \cdots, x_n$ to be zeros, we get a sub-function of $f_\kappa(\chi)$, just as follows.

$$f'_\kappa(x_1, x_2, \cdots, x_j) = f_\kappa(x_1, x_2, \cdots, x_j, 0_{j+1}, 0_{j+2}, \cdots, 0_n)$$

Furthermore, by setting part of the variables of f'_κ, $x_{i+1}, x_{i+2}, \cdots, x_j$, to be ones, we get a small function which contains less number of variables than f'_κ and f_κ.

$$\begin{aligned} g_\kappa(x_1, x_2, \cdots, x_i) &= f'_\kappa(x_1, x_2, \cdots, x_i, 1_{i+1}, 1_{i+2}, \cdots, 1_j) \\ &= f_\kappa(x_1, x_2, \cdots, x_i, 1_{i+1}, 1_{i+2}, \cdots, 1_j, 0_{j+1}, 0_{j+2}, \cdots, 0_n) \end{aligned}$$

The coefficient of the monomial $x_1 x_2 \cdots x_i$ in the newly obtained function g_κ is equivalent to the summation of several coefficients in f'_κ, which are also the coefficients of f_k. It is equal to

$$\bigoplus_{\alpha \,:\, \omega \trianglelefteq \alpha \trianglelefteq \upsilon} c_\alpha(\kappa) ,$$

where $\alpha \in V_n$. Calculating the i-th derivative of g_κ with respect to the variable sequence $(x_1, x_2, \cdots, x_i)$, we can get the coefficient of the monomial $x_1 x_2 \cdots x_i$, which implies,

$$\delta^{(i)}_{x_1, x_2, \cdots, x_i} g_\kappa(x_1, x_2, \cdots, x_i) = \bigoplus_{\alpha \,:\, \omega \trianglelefteq \alpha \trianglelefteq \upsilon} c_\alpha(\kappa) .$$

That is to say, we need to collect 2^i chosen inputs of the form

$$(*_1, *_2, \cdots, *_i, 1_{i+1}, 1_{i+2}, \cdots, 1_j, 0_{j+1}, 0_{j+2}, \cdots, 0_n) ,$$

and sum up their corresponding outputs of $f_\kappa(\chi)$. □

This theorem shows that if adversaries can get enough and specific inputs and their corresponding outputs of a tweakable cryptosystem, then certain information about secret keys can be recovered. If the obtained information can be represented as linear functions of the secret key bits, then the secret keys could easily be figured out. If the obtained coefficient is always inclined to be a constant value whatever the keys are chosen, then the system could simply be distinguished from randomness.

2.3 A Security Principle for Tweakable Cryptosystem Design

If certain components of a tweakable cryptosystem are far away from truly random distributions, such properties perhaps can be utilized by adversaries to perform actual attacks, such as differential attack and linear attack. To avoid such potential problems, defeat the cryptanalysis mentioned in the last subsection, and improve the security level of cryptosystems, the following definition is given to propose a security principle for tweakable system design.

Assume the following is the Boolean representation of a tweakable cryptosystem (component),

$$f_\kappa(\chi) = f_\kappa(x_1, x_2, \cdots, x_n) = \bigoplus_{\alpha \in V_n} c_\alpha(\kappa) x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n} \ , \tag{4}$$

where V_n is the vector space over $\mathbb{F}_2 = \{0, 1\}$ and each coefficient is a Boolean function with respect to the secret variable vector κ.

Definition 2 (Tweakable Balance). *In the keyed Boolean function (4), if for any nonempty sub-set of V_n, e.g., $S_n \subseteq V_n$, where $S_n \neq \emptyset$, and the all-one vector is not included in this sub-set, i.e.* $(1, 1, \cdots, 1) \notin S_n$, *then the Boolean function*

$$\bigoplus_{\alpha \in S_n} c_\alpha(\kappa) \tag{5}$$

should be

1. *balanced, or*
2. *if adversaries are allowed to fix (or know) the values of ϵ bits of secret variables, the resultant function after fixing these variables should be balanced.*

The Boolean function (4) is defined to be tweakably balanced *if it satisfies the first requirement. The function is defined to be* ϵ-th tweakably balanced *if it satisfies the second requirement.*

The feasibility of the attack. The value of the function (5) can be computed by using Lemma 2 directly. As long as we do not calculate the value of the coefficient of the monomial with the highest degree, i.e. $x_1 x_2 \cdots x_n$'s, we will not use up all the possible inputs. For example, the chosen input of the form $(1, 1, \cdots, 1)$ will be needed only when we want to compute the coefficient of $x_1 x_2 \cdots x_n$. Therefore, the attack to calculate any combination of the rest coefficients is valid and valuable in theory.

The sufficiency to defeat the attack. If the obtained Boolean function (5) is balanced, and κ is chosen randomly, the outputs will be uniformly distributed on $\mathbb{F}_2$. If the secret key κ is unknown, and the Boolean function is balanced, then the possibilities to get the values of 0 and 1 will be equal, in which case the cryptosystem is impossible to be distinguished from randomness.

A balanced function which is also ϵ-th order correlation immune (so-called ϵ-resilient function) seems to be ϵ-th tweakably balanced. The value of ϵ can serve as an indication of the security level of tweakable cryptosystems. The value of ϵ partially determines the hardness to attack the cryptosystem theoretically. This principle possibly leads cryptographers to look for the guidelines to help design round functions and key schedules, in order to ensure the tweakable balance. Perhaps the parameter ϵ could also be used to provide provable security for the necessary number of secret key bits, because for a balanced Boolean function of algebraic degree d of n variables its order of immunity ϵ will not be larger than $n - d - 1$ (see [11]).

The second requirement also ensures, to some extent, the Boolean representations of the coefficients in the function cannot be too simple, and thereby can increase the difficulty for adversaries to solve equations.

3 Relationships with Other Cryptanalysis Methods

In this section, the theoretical foundations of Cube Attack, Cube Tester, and algebraic IV differential attack (AIDA) are carefully investigated, which shows that all these theories are certain special cases of Theorem 1. Since higher order differential cryptanalysis (HODC) is not limited to analyze Boolean functions, Theorem 1 is just an application of HODC in Boolean algebra, cooperating with further chosen-input methods. The influence of the proposed cryptanalysis framework, in turn, to the theories of HODC is also discussed. Finally, a brief comparison among these cryptanalysis methods is given.

3.1 Relationship with Cube Attack

In Cube Attack [4], both public input bits and secret input bits are treated as variables, denoted by x_1, x_2, $\cdots$, x_n. A Boolean function can always be written as

$$p(x_1, x_2, \cdots, x_n) = t_I \cdot p_{S(I)} + q(x_1, x_2, \cdots, x_n) \ .$$

$I \subseteq \{1, 2, \cdots, n\}$ is an index set, and t_I is the monomial which is the multiplication of all the variables whose indices are in the set I. Here $p_{S(I)}$ is called the *superpoly* of I in p, which does not contain any common variable with t_I. And each monomial in $q(x_1, x_2, \cdots, x_n)$ misses at least one variable from I.

The *cube* C_I is defined as a set containing $2^{|I|}$ vectors by assigning all the possible combinations of 0/1 values to the variables whose indices are in I, and leaving all the other variables undetermined. Then we can obtain the superpoly $p_{S(I)}$ by summing up all the derived functions whose inputs are in the cube C_I. If $p_{S(I)}$ is a function with respect to secret variables, especially a linear function, then we may be able to solve the equations to figure out the secret information (e.g., certain secret key bits).

The basic idea of Cube Attack is exactly the same as Lemma 1. As long as the superpoly $p_{S(I)}$ is the Boolean function of secret variables, the monomial t_I is not the sub-monomial of any other monomial with respect to public variables in $p(x_1, x_2, \cdots, x_n)$. Theorem 1 provides a cryptanalysis framework suitable for more situations, and the idea of Cube Attack can be seen as a special case of Theorem 1.

3.2 Relationship with Cube Tester

The fundamental theory of Cube Tester [5] is the same as Cube Attack. However, the authors mentioned that assigning fixed values to certain public and secret variables for the sake of efficiency, such as choosing suitable values of some message (and key) words of MD6 in order to simplify the first several rounds. The authors also mentioned that assigning differential values to inputs will influence the coefficients in resultant Boolean functions. The analysis in Section 2 has detailedly investigated the underlying influence of such further chosen-input methods. Theorem 1 serves as a solid theoretical background for Cube Tester.

Moreover, instead of to extract secret information, Cube Tester intends to utilize efficient property testing algorithms to detect nonrandom behaviors of tweakable cryptosystems. This thought is very brilliant, but which property testers are suitable for the analysis of tweakable cryptosystems should be carefully considered. The best attack of Cube Tester on MD6 is obtained by testing the balance of the Boolean representations of coefficients.

3.3 Relationship with Algebraic IV Differential Attack

Algebraic IV differential attack (AIDA) is presented first in [6] and later in [7] in order to provide a powerful tool to analyze stream ciphers, especially Trivium. In AIDA, the Boolean functions of stream ciphers are written as

$$f(v_1, v_2, \cdots, v_n, \kappa) = \bigoplus_{I \subseteq \{1,2,\cdots,n\}} a_I(\kappa) v_I^{\wedge} \ ,$$

where $v_1, v_2, \cdots, v_n$ are the bits of IV, and κ denotes the vector of secret key bits. The notation $v_I^{\wedge}$ is similar to t_I in Cube Attack, and $a_I(\kappa)$ is the coefficient of the monomial $v_I^{\wedge}$.

The fundamental principle of AIDA is the following theorem,

$$a_I(\kappa) = \bigoplus_{M \subseteq I} d_M \ ,$$

where d_M denotes the corresponding entries in the truth table when the IV bits whose indices are in the set M are set to be ones, and the other bits are left to be zeros. Thus, $2^{|I|}$ queries of the function f are enough for calculating the value of $a_I(\kappa)$. This theorem is theoretically the same as Lemma 2, which is also a special case of Theorem 1.

3.4 Relationship with Higher Order Differential Cryptanalysis

Higher order differential cryptanalysis (HODC) is proposed in early 1990s [8, 9], intending to discover more advanced characteristics than differential attack, by using multiple pairs of chosen inputs. HODC has a systematic and theoretical foundation, and can be used to analyze the functions over an Abelian group. The analysis in Section 2 is only a special application of HODC in Boolean algebra, associating with further chosen-input methods. [12] shows the similar idea as Lemma 1, but unfortunately the authors did not develop it further.

However, the analysis in Section 2 reminds us there might be some areas undiscovered in HODC. The theories of HODC concentrates on the decrease of the overall degree of the function, e.g.,

$$\deg(\Delta_a f) \leq \deg(f) - 1 ,$$

where $\Delta_a f$ denotes the derivative of f as the point a. Maybe after carefully choosing the points to calculate higher order derivatives, we can get more detailed information about the functions over an Abelian group, just as the keyed coefficients in Boolean functions, or we can minimize the amount of chosen inputs to gain certain properties.

3.5 Summary

From the foregoing analysis, we can see that, except higher order differential cryptanalysis, the theoretical foundations of Cube Attack, Cube Tester and AIDA can be derived from the theorem in Section 2. In addition, due to the result of Lemma 1 is covered by Lemma 2, AIDA is more in-depth than Cube Attack in theory. And because the methods used in Cube Tester are the most closed to Theorem 1, Cube Tester is more advanced than AIDA and Cube Attack. The essential difference is how to deal with the variables not in the so-called cube. Whether the values of these variables are chosen in purpose and what values the variables are set to be, can lead to great differences in theory.

What we want to emphasize here is that even if more in-depth and advanced theories are provided, the practical cryptanalysis procedures are still not trivial works. Just as searching the differential paths of blockciphers and hash functions, finding appropriate properties of Boolean functions (in this paper, choosing proper coefficients) also needs a large amount of computations and trials, and even some personal skills. For example, although AIDA [6] is proposed before the first announcement of Cube Attack at CRYPTO'08, and the theory of AIDA is somewhat more in-depth than Cube Attack, the actual result of analyzing Trivium via AIDA (breaking 576 rounds) is worse than the result in the paper about Cube Attack (breaking 767 rounds). The next section will discuss some practical procedures to perform such cryptanalysis.

4 Cryptanalysis Procedures in Practice

This section discusses some possible methods that can be used in offline phase to discover potential defects of tweakable cryptosystems, such as nonrandom behaviors.

4.1 Empirical Approaches

A straightforward way is to randomly choose inputs and then check whether the outputs are obviously away from the uniform distribution. The following is a simple algorithm to check the randomness of the Boolean function (4).

```
**Require:** $\omega \trianglelefteq \upsilon \in V_n$
1: $\xi \leftarrow 0$
2: (optionally) fix the values of $\epsilon$ variables in $\kappa = (k_1, k_2, \cdots)$
3: **for** $i = 1$ **to** $N$ **do**
4:     randomly pick the values of the (other) variables in $\kappa$
5:     compute $\eta = \bigoplus_{\alpha:\omega \trianglelefteq \alpha \trianglelefteq \upsilon} c_\alpha(\kappa)$ by using $2^{wt(\omega)}$ chosen $\chi = (x_1, \cdots, x_n)$
6:     $\xi \leftarrow \xi + \eta$
7: **end for**
8: **return** $\xi$
```

If N is big enough, in a truly random system, ξ should be closed to $N/2$. Cryptographers can also take advantage of some efficient property testing algorithms to speed up the discovery of inner flaws.

An advantage of this kind of experimental approaches is that we do not need to know the detailed constructions of the schemes. We can treat any tweakable system as a block box, and perform cryptanalysis.

4.2 Theoretical Methods

If the detailed specification of a cryptosystem is known, certainly it can be utilized to help us analyze the system. For example, if we can symbolically calculate the keyed Boolean representations of certain output bits, then we can clearly and concretely know some properties of the cryptosystem. This kind of methods is perhaps suitable for simple schemes whose Boolean representations do not have very high algebraic degrees and do not involve too many input variables.

In a well-designed cryptosystem, to get its Boolean representations may be quite time-consuming, so we would need some methods to simplify computations. Two possible ways are proposed as follows.

1. Do not consider public variables. For instance, set all the public variables in the function (2), $x_1, \cdots, x_n$, to be zeros, which will save a lot of space and time for computations. Then calculate the symbolic Boolean function with respect to the secret variables, $k_1, k_2, \cdots$. The obtained Boolean function is exactly the representation of the constant term in the function (2). If we can correctly calculate this Boolean function, then we can get certain information about the secret key bits in online phase by using only one chosen input.
2. Do not consider secret variables. For example, just indicate the coefficients in the function (2) with either 0 or 1. Under such circumstance, we possibly intend to check whether a monomial exists in the Boolean function, i.e. whether a coefficient is always equal to 0. This situation is much common to see when certain Boolean representations do not reach the possibly highest algebraic degrees due to the lame designs of round functions.

In [10], an efficient algorithm of the second kind is proposed, which is suitable for both practical calculations and theoretical proofs.

5 Conclusion and Further Works

This paper mainly concentrated on how to analyze the Boolean representations of tweakable cryptosystems, and provided a concrete mathematical theorem which is more in-depth than all the theories of higher order differential cryptanalysis in Boolean algebra, Cube Attack, Cube Tester and AIDA. The differences among these cryptanalysis methods are also clarified through the view of this new theorem. Moreover, a principle for designing secure tweakable cryptosystems is proposed to defeat such cryptanalysis methods.

With the development of dedicated and compact cryptosystem designs, the views of cryptanalysis methods become more and more deep and detailed. Such change seems to be natural and inevitable, because even if the overall condition of a system is secure enough, certain small components may still be flawed. These newly proposed cryptanalysis methods can possibly support or cooperate with traditional ways, especially differential cryptanalysis, linear cryptanalysis and integral cryptanalysis, which would be one of our further works.

As we mentioned in Section 3.5, although a more in-depth cryptanalysis framework is proposed, the actual analysis of cryptosystems still requires hard working. This paper only provided the theoretical foundation, and thus the influence of the new theorem to the practical world is not determined. Our further works would also include utilizing the new theorem to analyze some existing, practical cryptosystems.

References

1. Bogdanov, A., Knudsen, L., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-lightweight Block Cipher. CHES'07. LNCS 4727 (2007) pp. 450–466
2. Cannire, C.D., Dunkelman, O., Kneevi, M.: KATAN & KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. CHES'09. LNCS 5747 (2009) pp. 272–288
3. The eSTREAM Project. `www.ecrypt.eu.org/stream/`
4. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. EUROCRYPT'09. LNCS 5479 (2009) pp. 278–299
5. Aumasson, J., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. FSE'09. LNCS 5665 (2009) pp. 1–22

6. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. Cryptology ePrint Archive, Report 2007/413 (2007) `http://eprint.iacr.org/`.
7. Vielhaber, M.: AIDA Breaks BIVIUM (A&B) in 1 Minute Dual Core CPU Time. Cryptology ePrint Archive, Report 2009/402 (2009) `http://eprint.iacr.org/`.
8. Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry (1994) pp. 227
9. Knudsen, L.: Truncated and Higher Order Differentials. FSE'95. LNCS 1008 (1995) pp. 196–211
10. Zhu, B., Chen, K., Lai, X.: Bitwise Higher Order Differential Cryptanalysis. In: The First International Conference on Trusted Systems (INTRUST'09). To appear
11. Siegenthaler, T.: Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications (Corresp.). IEEE Transactions on Information Theory **30**(5) (1984) pp. 776–780
12. Moriai, S., Shimoyama, T., Kaneko, T.: Higher Order Differential Attack Using Chosen Higher Order Differences. SAC'98. LNCS 1556 (1999) pp. 106–117