On the Security of NOEKEON against Side Channel Cube Attacks

Shekh Faisal Abdul-Latip[1,2], Mohammad Reza Reyhanitabar[1], Willy Susilo[1], and Jennifer Seberry[1]

[1] Center for Computer and Information Security Research, School of Computer Science and Software Engineering, University of Wollongong, Australia
{sfal620,rezar,wsusilo,jennie}@uow.edu.au
[2] Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malaysia
shekhfaisal@utem.edu.my

Abstract. In this paper, we investigate the security of the NOEKEON block cipher against side channel cube attacks. NOEKEON was proposed by Daemen et al. for the NESSIE project. The block size and the key size are both 128 bits. The cube attack, introduced by Dinur and Shamir at EUROCRYPT 2009, is a new type of algebraic cryptanalysis. The attack may be applied if the adversary has access to a single bit of information that can be represented by a *low degree* multivariate polynomial over GF(2) of secret and public variables. In the side channel attack model, the attacker is assumed to have access to some leaked information about the internal state of the cipher as well as the plaintext and ciphertext. Adopting the notion of a single bit leakage as formalized by Dinur and Shamir, we assume that the attacker has only one bit of information about the intermediate state after each round. Using this side channel attack model, we show that it is possible to extract 60 *independent* linear equations over 99 (out of 128) key variables. To recover the whole 128-bit key, the attack requires only about 2^{10} chosen plaintext and $O(2^{68})$ time complexity.

Keywords: Algebraic cryptanalysis, block ciphers, cube attacks, NOEKEON, side channel attacks.

1 Introduction

Almost any cryptosystem can be represented by a system of multivariate polynomial equations over a finite field, e.g. GF(2). The cube attack, formalized by Dinur and Shamir at EUROCRYPT 2009 [9], is a generic type of algebraic attacks. The attack aims to derive low-degree equations that can be exploited for constructing distinguishers, e.g. [2], and/or key recovery attacks, e.g. [9, 2]. An interesting feature of the cube attack is that it only requires a black-box access to a target cryptosystem and may be applied even if only a few output bits can be accessed by an adversary.

J. Kwak et al. (Eds.): ISPEC 2010, LNCS 6047, pp. 45–55, 2010.

For a properly designed ("good") cipher, whose algebraic representation over GF(2) is of degree d (i.e. the maximum degree of the output bits represented as boolean functions is d), the cube attack will require about 2^d computations. Therefore, *in general*, the attack's success depends on whether this computational complexity is feasible or not; however we note that this "generic complexity" does not imply that the attack may not be successful for specific cases. The cube attack has been successfully applied (for key recovery) against a reduced-round variant of the Trivium [6] stream cipher in [9] and (as a distinguisher) against a reduced-round variant of the MD6 hash function [14] in [2].

In trying to apply cube attacks to block ciphers, the main problem is that the degree of the polynomial representing a ciphertext bit grows exponentially with the number of rounds in the cipher. Hence, the cube attack usually becomes ineffective after a few rounds if one considers only the standard attack model that is used in the well-known statistical attacks such as the Differential and Linear attacks. Nevertheless, considering the practical implementations of the block cipher, especially in resource limited systems such as smart cards, there is a stronger attack model, namely the side channel attack model, where the adversary is given more power by having access to some "*limited* " information leaked about the internal state of the cipher. This information leakage can be via physical side channels, such as timing, electrical power consumption, electromagnetic radiation, probing, etc.

Dinur and Shamir in [10] proposed a side channel attack model, where the adversary is assumed to have access to "*only one bit of information*" about the internal state of the block cipher after each round. This one bit of information can be, for example, a single bit of the internal state itself or a bit of information about the Hamming weight of the internal state. They showed that the cube attack in this single-bit-leakage side channel model can recover the secret key of the Serpent [1] and AES [7] block ciphers much easier than the previously known side channel attacks. Recently, Yang et al. at CANS 2009 [15] investigated the side channel cube attack on the PRESENT block cipher [5]. It is worth noticing that the single-bit-leakage side channel cube attack against a block cipher is different from a cube attack against a reduced-round variant of the cipher; while in the former the adversary has access to only one bit of information about the internal state, in the latter the adversary has access to the whole internal state after a reduced number of rounds.

In this paper, we investigate the security of the NOEKEON block cipher [8] against the cube attack in the single-bit-leakage side channel attack model. NOEKEON is designed to be efficient and secure on a wide range of platforms including the environment with limited resources such as smart cards. The designers of NOEKEON have shown how the cipher can be adapted to provide an anti-DPA variant of NOEKEON which is secure against differential power analysis (DPA) attacks. However, the implementation cost for the anti-DPA version is almost twice that of a normal NOEKEON implementation. They also noticed that to prevent higher-order DPA attacks one will have to use even less efficient variants. If the cipher were to be implemented in a time-critical system,

it is likely that this kind of anti-DPA variants not to be implemented due to the efficiency/security trade-off. Many papers on side channel attacks such as in [12, 13] also concentrate on countermeasures which minimize the leakage from the implementation.

We note that these kind of countermeasures against specific types of side channel attacks (e.g. timing attack or DPA) do not remove the possibility of all side channel attacks such as electromagnetic radiations, probing, etc. In this paper we do not consider these specific issues and countermeasures about the actual physical aspects of the implementation attacks and how information leakage can be measured. Rather, we assume the single-bit-leakage side channel model as an abstract attack model, and concentrate on investigating the security of ciphers against cube attacks in this attack model.

Our Contribution. We investigate the security of the NOEKEON block cipher against cube attacks in the single-bit-leakage side channel attack model. First, we show how to determine the appropriate round to find most of the key bits. Using a single bit of the internal state after the second round, we extract 60 *independent* linear equations over 99 key variables. To recover the whole 128-bit key, the attack requires about 2^{10} chosen plaintext and $O(2^{68})$ time complexity.

Organization of the Paper. In Section 2 and 3, we review the cube attack and the construction of NOEKEON block cipher, respectively. Section 4 contains the main contribution of this paper, where we provide the details of the side channel cube attack on NOEKEON. Section 5 concludes the paper and provides some open problems for future research.

2 A Review on the Cube Attack

In algebraic attacks, one aims at recovering the secret key in cryptosystems by manipulating and solving the underlying algebraic equations. Solving a system of multivariate equations over a finite field $\mathbb{F}$, in general, is known to be an NP-hard problem [11]. However, it has been demonstrated that finding solutions faster than by the exhaustive search may be possible if the algebraic equations have a relatively low algebraic degree when considered as multivariate polynomial equations. An ideal situation is when one can derive enough number of independent linear equations which are easily solvable (e.g. by Gaussian elimination); the cube attack aims at doing this.

The main point of the cube attack is that, the multivariate "master" polynomial $p(v_1, \cdots, v_m, k_1, \cdots, k_n)$, representing an output bit of a cryptosystem over GF(2) of secret variables k_i (key bits) and public variables v_i (i.e. plaintext or initial values), may induce algebraic equations of lower degrees, in particular *linear* equations. The cube attack provides a method to derive such lower degree (especially linear) equations, given the master polynomial only as a black-box which can be evaluated on the secret and public variables.

Let's ignore the distinction between the secret and public variables' notations and denote all of them by $x_i, \cdots, x_\ell$, where $\ell = m + n$. Let $I \subseteq \{1, ..., \ell\}$ be a

subset of the variable indexes, and t_I denote a monomial term containing multiplication of all the x_i's with $i \in I$. By factoring the master polynomial p by the monomial t_I, we have:

$$p(x_1, \cdots, x_\ell) = t_I \cdot p_{S(I)} + q(x_1, \cdots, x_\ell) \tag{1}$$

where $p_{S(I)}$, which is called the *superpoly* of t_I in p, does not have any common variable with t_I, and each monomial term t_J in the residue polynomial q misses at least one variable from t_I. A term t_I is called a "*maxterm*" if its superpoly in p is linear polynomial which is not a constant, *i.e.* $deg(p_{S(I)}) = 1$.

The main observation of the cube attack is that, if we sum p over t_I, i.e. by assigning all the possible combinations of 0/1 values to the x_i's with $i \in I$ and fixing the value of all the remaining x_i's with $i \notin I$, the resultant polynomial equals to $p_{S(I)} \pmod 2$. More formally, a subset I of size s (where $s \leq \ell$) defines a boolean cube C_I containing 2^s boolean vectors which are formed by assigning all 2^s values to the x_i's with $i \in I$, and leaving all the remaining variables (i.e. x_i's with $i \notin I$) undetermined. For example, if $I = \{1, 2\}$ then $C_I = \{(0, 0, x_3, \cdots, x_\ell), (0, 1, x_3, \cdots, x_\ell), \quad (1, 0, x_3, \cdots, x_\ell), (1, 1, x_3, \cdots, x_\ell)\}$. Any vector $\mathbf{w} \in C_I$ defines a derived polynomial $p_{|\mathbf{w}}$ with $\ell - s$ variables whose degree may be the same or lower than the degree of the master polynomial p. Summing the 2^s derived polynomials over GF(2) defined by the vectors in the cube C_I, we get a new polynomial p_I defined by $p_I \triangleq \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}$. The following theorem states the main observation used by the cube attack.

Theorem 1 (The Main Observation [9]). *Given a polynomial p* over GF(2) *with ℓ variables, and any index subset $I \subseteq \{1, \cdots, \ell\}$, we have $p_I = p_{S(I)}$.*

Given access to a cryptographic function with public and secret variables, this observation enables an attacker to recover the value of the secret variables (k_i's) in two steps, namely preprocessing and online phase, which are described in the sequel.

Preprocessing Phase. During the preprocessing phase, the attacker first finds sufficiently many maxterms, i.e. t_I's, such that each t_I consists of a subset of public variables $v_1, \cdots, v_m$. To find the maxterms, the attacker performs a probabilistic linearity test on $p_{S(I)}$ over the secret variables $k_i \in \{k_1, \cdots, k_n\}$ while the value of the public variables not in t_I are fixed (to 0 or 1). For example, the BLR test of [4] can be used for this purpose. This test requires the attacker to choose a sufficient number of vectors $\mathbf{x}, \mathbf{y} \in \{0, 1\}^n$ independently and uniformly at random representing samples of n-bit key, and then for each pair of vectors $\mathbf{x}$ and $\mathbf{y}$, the attacker sums the polynomial p over t_I to verify whether or not each one of them satisfies the relation:

$$p_{S(I)}[\mathbf{0}] + p_{S(I)}[\mathbf{x}] + p_{S(I)}[\mathbf{y}] = p_{S(I)}[\mathbf{x} + \mathbf{y}] \tag{2}$$

If all the vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy the relation, with high probability $p_{S(I)}$ is linear over the secret variables; that is, t_I is a maxterm. Then the next step is to derive linearly independent equations in the secret variables k_i's from $p_{S(I)}$ that

are closely related to the master polynomial p, such that, solving them enables the attacker to determine the values of the secret variables.

Online Phase. Once sufficiently many linearly independent equations in the secret variables are found, the preprocessing phase is completed. In the online phase, the attacker's aim is to find the value of the right-hand side of each linear equation by summing the black box polynomial p over the same set of maxterms t_I's which are obtained during the preprocessing phase. Now, the attacker can easily solve the resultant system of the linear equations, e.g. by using the Gaussian elimination method, to determine the values of the secret (key) variables.

3 A Brief Description of the NOEKEON Block Cipher

NOEKEON [8] is a block cipher with a block and key length of 128 bits. It produces a ciphertext after iterating a round function 16 times, followed by a final output function. The specification of NOEKEON [8], provides a key schedule which converts the 128-bit "Cipher Key" (i.e. the original key) into a 128-bit "Working Key", which is used in the round function. However, the use of the key schedule is optional. If related-key attack scenarios [3] are not of a concern, then the key schedule is not applied (i.e. the Cipher Key is used directly as the Working Key), and the cipher is called to be used in the "direct-key mode". Otherwise, it operates in the "indirect-key mode", where the Cipher Key is processed by the key schedule algorithm to produce the Working Key.

A graphical representation of the round function of NOEKEON is shown in Fig. 1. It consists of two linear functions, Θ and Π (as illustrated in the figure), and a nonlinear function Γ which is described shortly. We describe the encryption mode of the cipher. The description of the cipher in the decryption (i.e. inverse) mode is also quite straightforward, where the inverse of the component functions are used (we refer to [8] for a complete description of the cipher). The constants C_1 and C_2 are two round constants. C_2 is set to zero during the encryption process and C_1 is set to zero during the decryption process. The constant C_1 that is used during the encryption process can be computed in recursive way as follows:

C_1^0 = 0x80;
if (C_1^r & 0x80 != 0) then C_1^{r+1} = $C_1^r \ll 1$ ^ 0x1B
else C_1^{r+1} = $C_1^r \ll 1$;

Let $A_r = A_r^0 A_r^1 A_r^2 A_r^3$ denote the 128-bit internal state after round r; where A_r^i's are 32-bit words, and A_0 contains the input plaintext P to the cipher. Then NOEKEON encryption algorithm can be described as follows:

For $r = 0$; $r < 16$; r++
$\quad A_{r+1} = \Pi^{-1}(\Gamma(\Pi(\Theta(A_r, K))))$;
$A_{17} = \Theta(A_{16}, K)$

where A_{17} denotes the 128-bit output ciphertext, and $K = K_0K_1K_2K_3$ is the 128-bit working key (K_i's are 32-bit words).

The nonlinear transformation Γ operates on 32 4-tuples of bits (4-bit boxes) using a S-box which is shown in Table 1 (4-bit values are represented by a hexadecimal number). The 4-bit boxes which are input to the S-Boxes, at the round r, are formed by selecting four bits from the words $A_r^0, A_r^1, A_r^2, A_r^3$, that are in the same position. For example, box 1 consists of the first bit from each word, box 2 contains the second bit from each word, and so on.

Table 1. Specification of the S-box in the function Γ

Input:	$x_3x_2x_1x_0$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Output:	$y_3y_2y_1y_0$	7	A	2	C	4	8	F	0	5	9	1	E	3	D	B	6

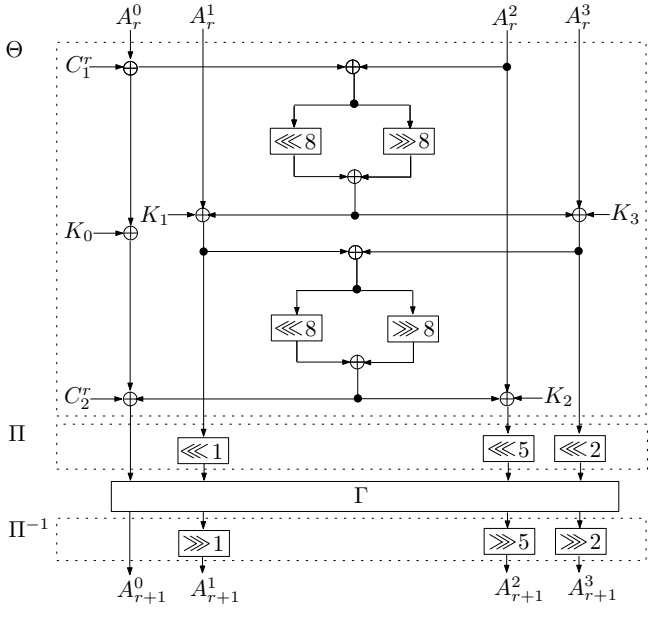

Fig. 1. The round function of NOEKEON

4 The Side Channel Cube Attack on NOEKEON

4.1 Finding the Efficient Round for Side Channel Attacks

In order to apply the single-bit-leakage side channel cube attack on the NOEKEON block cipher, we need to find out the round in which the cipher begins achieving complete diffusion. This enables us to find most of the key variables from low degree master polynomials in early rounds of the encryption process.

To do this, we determine the number of key variables that can be found within the master polynomials representing each bit of the internal state after each round; i.e. the master polynomials representing $A_r^j[i]$ for $1 \leq r \leq 16, 0 \leq j \leq 3$, and $0 \leq i \leq 31$.

The following lemma is a corollary of the main observations supporting the cube attack as described in Sec. 2.

Lemma 1. *Let p be a given (black-box) master polynomial over GF(2) in ℓ variables $x_1, \cdots, x_\ell$; $I \subseteq \{1, \cdots, \ell\}$; $s = |I|$, and t_I denote the multiplication of x_i's with $i \in I$. Let $p_I \triangleq \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}$ be the derived polynomial obtained by summing p over the cube C_I (see Sec. 2). t_I appears in p, either as a monomial or as a common subterm of some monomials, if and only if there exist at least a vector $\mathbf{x} \in \{0,1\}^{\ell-s}$ which satisfies $p_I[\mathbf{x}] = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}[\mathbf{x}] = 1$.*

Proof. If t_I is neither a monomial nor a common subterm of some monomials in the master polynomial p, then each monomial term in p must miss at least one variable from t_I (i.e an x_i with $i \in I$). Therefore, in the summation of p over the cube C_I each monomial will be summed an even number of times, and hence will be canceled out as the coefficients are from GF(2); that is, $p_I \equiv 0$ (and hence for all $\mathbf{x} \in \{0,1\}^{\ell-s}$ we have $p_I[\mathbf{x}] = 0$). To complete the proof, simply note that if t_I appears as a monomial or a common subterm of some monomials in p, then p can be factored as $p(x_1, \cdots, x_\ell) = t_I \cdot p_{S(I)} + q(x_1, \cdots, x_\ell)$, where $p_{S(I)}$ is either the constant 1 (when t_I is an independent monomial in p) or a polynomial over at most $\ell - s$ variables (when t_I is a common subterm of some monomials). From Theorem 1 we know that $p_I = p_{S(I)}$, and hence $p_I[\mathbf{x}] = p_{S(I)}[\mathbf{x}] = 1$ for *at least* a vector $\mathbf{x} \in \{0,1\}^{\ell-s}$. □

This lemma essentially provides the underlying idea for the following probabilistic test, proposed originally in [2], to detect whether a given t_I appears either as a monomial or as a common subterm of some monomials within a master polynomial p:

Step 1: select a value of vector $\mathbf{x} = (x_{s+1}, \cdots, x_\ell) \in \{0,1\}^{\ell-s}$.
Step 2: sum the polynomial $p(x_1, \cdots, x_\ell)$ over the cube C_I (i.e. over all values of $(x_1, \cdots, x_s)$, where $s = |I|$), to get $p_I[\mathbf{x}]$ $(= p_{S(I)}[\mathbf{x}])$.
Step 3: repeat the previous two steps N times and record the values of $p_I[\mathbf{x}]$.

A random master polynomial p is expected to contain at least a monomial $x_1 \cdots x_s \cdots x_j$ $(j \geq s)$ in which $t_I = (x_1, \cdots, x_s)$ is either the monomial itself (for $j = s$) or a subterm of the monomial (for $j > s$), with high probability. Hence, after a sufficiently large number of repetitions N, one would find at least one nonzero superpoly $p_{S(I)}$ with high probability. However if t_I is neither a term nor a common subterm in p, the superpoly $p_{S(I)}$ will always evaluate to zero.

We apply this test to *estimate* the the number of key variables within the master polynomials representing each bit of the internal state; i.e. the master polynomials representing $A_r^j[i]$ for $1 \leq r \leq 16, 0 \leq j \leq 3$, and $0 \leq i \leq 31$. To

determine whether a key variable k_i, $0 \leq i \leq 127$, exists in a master polynomial p, the following process is repeated for each i $(0 \leq i \leq 127)$. Let $t_I = k_i$, randomly choose "sufficiently many" vectors $\mathbf{x} \in \{0,1\}^{128+127}$ (i.e. the space for 128 bits of the plaintext and the secret variables excluding k_i), and sum the polynomial p over t_I for each sample vector $\mathbf{x}$. If at least one sample vector $\mathbf{x}$ results in $p_I = 1$, then from Lemma 1, we can conclude that the key variable k_i exists within the master polynomial p. In our experiment, only after testing about 300 random sample vectors, we have been able to successfully determine that a high portion of the key variables exist in the master polynomials after only two rounds, as shown in Table 2.

Table 2. The number of key variables within the black box polynomials

	$A_r^0[i]$ $0 \leq i \leq 31$	$A_r^1[i]$ $0 \leq i \leq 31$	$A_r^2[i]$ $0 \leq i \leq 31$	$A_r^3[i]$ $0 \leq i \leq 31$
1^{st} round $(r = 1)$	28	28	28	21
2^{nd} round $(r = 2)$	127	127	127	123
$3^{rd} - 16^{th}$ round $(r \geq 3)$	128	128	128	128

We also need to estimate the degree d of each (black-box) master polynomial in order to verify whether the cube attack can be efficiently implemented. For this purpose, we construct the boolean functions representing output bits of the NOEKEON's S-boxes. Let x_i and y_i be, respectively, the input and output bits of the S-box, for $i = 0, 1, 2, 3$. Each output bit y_i can be represented as a boolean function, as follows.

$$y_0 = x_0 + x_1 + x_2 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_3 + x_2x_3 + x_0x_2x_3 + 1$$
$$y_1 = x_2 + x_3 + x_0x_1 + x_1x_2 + x_0x_2x_3 + x_1x_2x_3 + 1$$
$$y_2 = x_0 + x_1 + x_1x_2 + x_2x_3 + 1$$
$$y_3 = x_0 + x_1x_2$$

It is easy to see that the highest degree of the four boolean functions is 3, which correspond to y_0 and y_1. Since the only nonlinear transformations in NOEKEON are the S-boxes (used by the function Γ), we can estimate the degree d of the master polynomial for any round. If we consider a single bit leakage after the third round, the degree of the master polynomials would be approximately between 16 and (the maximum) 256, which is not suitable for our side channel attack model, as it may be unlikely in practice to obtain a leakage that falls within the internal state bits with low degree master polynomials. However if we consider the degree after the second round, it turns out that the degrees of the master polynomials are considerably low, i.e. between 4 and 27. Considering that diffusion process is also almost complete after the second round (refer to Table 2), it turns out that the second round is an appropriate round for the single-bit-leakage side channel attack purpose.

4.2 Finding Maxterm Equations

As an example of the single-bit-leakage attack, in this section we provide the results of the analysis to find maxterm equations from a master polynomial associated with the first bit of the internal state after the second round, i.e. $A_2^0[0]$. (We note that this process can also be straightforwardly applied to any other single bit position in the internal state after the second round, i.e. considering the master polynomial of any $A_2^j[i]$ for $0 \leq j \leq 3$, and $0 \leq i \leq 31$).

Table 3. Cube indexes for maxterms and the linear superpoly for each maxterm

Cube Indexes	Maxterm Equation	Cube Indexes	Maxterm Equation
{11,75,82}	k_{39}	{0,53,60,97}	$k_{113} + k_{89} + k_{57} + k_{49} + k_{33}$
{27,82,91}	k_{55}	{0,4,39,62}	$k_{98} + k_{34} + k_{26} + k_{10} + k_2 + 1$
{1,34,42,127}	k_{100}	{1,4,34,58}	$k_{119} + k_{111} + k_{95} + k_{71} + k_{55} + k_{47}$
{0,4,49,126}	k_{102}	{1,3,25,66}	$k_{118} + k_{110} + k_{94} + k_{70} + k_{54} + k_{46}$
{2,7,36,107}	k_{105}	{0,1,24,99}	$k_{116} + k_{108} + k_{92} + k_{68} + k_{52} + k_{44}$
{1,34,42,79}	k_{108}	{0,20,31,99}	$k_{112} + k_{96} + k_{80} + k_{64} + k_{48} + k_{32}$
{105,118,126}	$k_{107} + 1$	{0,28,39,62}	$k_{114} + k_{106} + k_{50} + k_{42} + k_{26} + k_2$
{2,23,36,123}	k_{121}	{0,5,22,86}	$k_{113} + k_{97} + k_{81} + k_{65} + k_{49} + k_{33}$
{0,12,49,126}	k_{110}	{82,86,123}	$k_{114} + k_{106} + k_{90} + k_{66} + k_{50} + k_{42}$
{118,121,126}	k_{123}	{0,2,75,87}	$k_{115} + k_{107} + k_{91} + k_{67} + k_{51} + k_{43}$
{1,34,58,119}	k_{124}	{3,4,62,76}	$k_{116} + k_{61} + k_{60} + k_{52} + k_{36} + k_{28}$
{5,33,62,85}	$k_{126} + 1$	{70,82,123}	$k_{120} + k_{96} + k_{80} + k_{72} + k_{56} + k_{32}$
{4,13,37,126}	k_{33}	{20,84,119}	$k_{122} + k_{114} + k_{74} + k_{66} + k_{58} + k_{50}$
{104,120,126}	k_{36}	{0,2,15,67}	$k_{122} + k_{106} + k_{58} + k_{42} + k_{26} + k_{10}$
{96,104,126}	k_{44}	{1,3,9,66}	$k_{123} + k_{107} + k_{91} + k_{75} + k_{59} + k_{43}$
{0,7,8,64}	k_{60}	{1,4,34,42}	$k_{126} + k_{118} + k_{78} + k_{70} + k_{62} + k_{54}$
{0,6,50,67}	k_{63}	{2,3,101,125}	$k_{116} + k_{60} + k_{52} + k_{45} + k_{37} + k_{12} + k_4$
{2,31,36,107}	k_{97}	{0,7,9,107}	$k_{120} + k_{104} + k_{96} + k_{56} + k_{40} + k_{32} + k_0 + 1$
{0,3,101,125}	$k_{98} + 1$	{1,4,13,126}	$k_{118} + k_{110} + k_{102} + k_{78} + k_{54} + k_{46} + k_{38} + 1$
{46,121,126}	$k_{99} + 1$	{1,5,92,99}	$k_{125} + k_{109} + k_{101} + k_{69} + k_{61} + k_{45} + k_{37}$
{0,8,47,74}	$k_{103} + k_{38}$	{1,2,44,107}	$k_{127} + k_{119} + k_{111} + k_{63} + k_{56} + k_{55} + k_{47} + k_{23} + 1$
{2,89,96,101}	$k_{118} + k_{53}$	{1,2,60,89}	$k_{119} + k_{111} + k_{55} + k_{47} + k_{40} + k_{32} + k_{31} + k_7 + 1$
{0,7,8,90}	$k_{119} + k_{54}$	{1,36,99,126,127}	$k_{111} + k_{103} + k_{56} + k_{47} + k_{39} + k_{32} + k_{31} + k_{23}$
{0,8,47,66}	$k_{127} + k_{62}$	{3,38,114,124,127}	$k_{57} + k_{41} + k_{33} + k_1$
{30,82,123}	$k_{98} + k_{90} + k_{74} + k_{66} + k_{34}$	{27,54,122,124,127}	$k_{121} + k_{33} + k_{25} + k_9 + k_1 + 1$
{0,10,19,95}	$k_{99} + k_{91} + k_{75} + k_{67} + k_{35}$	{105,108,120,124,126,127}	$k_{92} + k_{61} + k_{28} + 1$
{0,1,64,99}	$k_{100} + k_{92} + k_{76} + k_{68} + k_{36}$	{57,120,121,124,126,127}	$k_{113} + k_{57} + k_{49} + k_{33} + k_{25} + 1$
{4,6,11,34}	$k_{103} + k_{95} + k_{79} + k_{71} + k_{39} + 1$	{102,113,122,124,126,127}	$k_{121} + k_{65} + k_{57} + k_{41} + k_{33} + 1$
{6,62,68,74}	$k_{104} + k_{80} + k_{72} + k_{64} + k_{40}$	{94,121,122,124,126,127}	$k_{108} + k_{60} + k_{44} + k_{37} + k_{36} + k_4$
{0,5,6,70}	$k_{105} + k_{81} + k_{73} + k_{65} + k_{41}$	{78,113,114,124,126,127}	$k_{127} + k_{119} + k_{79} + k_{71} + k_{63} + k_{55}$

By running the preprocessing phase of the attack (on a single PC) for several weeks, we have been able to find collectively thousands of maxterm equations using different cubes sizes, where most of them were found to be redundant and linearly dependent equations. To filter the equations and obtain only linearly independent equations among them, we used the well-known Gaussian elimination. The elimination gives us only 60 linearly independent equations over 99 key variables. Table 3 shows the indexes of variables in the maxterms and the corresponding linearly independent equations that we have obtained. (The indexes for both the plaintext and the key variables start from index 0, namely the MSB, until 127).

As shown in the table, the maxterms start to appear within t_I's of size 3; we have 2 maxterms of size 3, 50 maxterms of size 4, 3 maxterms of size 5, and 5 maxterms of size 6. Hence, the total number of the chosen plaintexts for the online phase of the cube attack is $2 \times 2^3 + 50 \times 2^4 + 3 \times 2^5 + 5 \times 2^6 \approx 2^{10.27}$. Considering that we have 60 linearly independent equations over the key

variables, the total time complexity to find the correct 128-bit key reduces to $O(2^{68})$ (compared to the $O(2^{128})$ for an exhaustive key search attack).

5 Conclusions

We investigated the security of the direct-key mode of the NOEKEON block cipher against cube attacks in the (single-bit-leakage) side channel attack model. Our analysis shows that one can recover the 128-bit key of the cipher, by considering a one-bit information leakage from the internal state after the second round, with time complexity of $O(2^{68})$ evaluations of the cipher, and data complexity of about 2^{10} chosen plaintexts. At this step, we have been able to find 60 linearly independent equations over 99 key variables, but from an initial observation, it seems that some nonlinear equations of low degree especially of degree 2 can also be easily found, which may further reduce the complexity of the attack. We leave extending the attack and exploiting such quadratic equations for a future research.

References

[1] Anderson, R., Biham, B., Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. In: First Advanced Encryption Standard (AES) Conference (1998)
[2] Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In: Dunkelman, O. (ed.) Fast Software Encryption. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)
[3] Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 229–246. Springer, Heidelberg (1994)
[4] Blum, M., Luby, M., Rubinfield, R.: Self-Testing/Correcting with Application to Numerical Problems. In: STOC, pp. 73–83. ACM, New York (1990)
[5] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
[6] De Cannière, C., Preneel, B.: TRIVIUM. In: Robshaw, M.J.B., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)
[7] Daemen, J., Rijmen, V.: AES Proposal: Rijndael. Technical Evaluation, CD-1: Documentation (1998)
[8] Daemen, J., Peeters, M., Van Assche, G., Rijmen, V.: Nessie Proposal: NOEKEON. In: First Open NESSIE Workshop (2000), `http://gro.noekeon.org`
[9] Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009)
[10] Dinur, I., Shamir, A.: Side Channel Cube Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2009/127 (2009), `http://eprint.iacr.org/2009/127`

[11] Fraenkel, A.S., Yesha, Y.: Complexity of Problems in Games, Graphs, and Algebraic Equations. Discr. Appl. Math. 1, 15–30 (1979)
[12] Mamiya, H., Miyaji, A., Morimoto, H.: Efficient Countermeasures against RPA, DPA, and SPA. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 243–319. Springer, Heidelberg (2004)
[13] Mangard, S.: Hardware countermeasures against DPA – A statistical analysis of their effectiveness. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004)
[14] Rivest, R., Agre, B., Bailey, D.V., Crutchfield, C., Dodis, Y., Fleming, K.E., Khan, A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., Yin, Y.L.: The MD6 Hash Function - A Proposal to NIST for SHA-3, http://groups.csail.mit.edu/cis/md6/
[15] Yang, L., Wang, M., Qiao, S.: Side Channel Cube Attack on PRESENT. In: Miyaji, A., Echizen, I., Okamoto, T. (eds.) CANS 2009. LNCS, vol. 5888, pp. 379–391. Springer, Heidelberg (2009)