Extended Cubes: Enhancing the Cube Attack by Extracting Low-Degree Non-Linear Equations

Shekh Faisal Abdul-Latip*
School of Computer Science and Software Engineering
University of Wollongong, Australia
sfal620@uowmail.edu.au

Mohammad Reza Reyhanitabar
School of Computer Science and Software Engineering
University of Wollongong, Australia
rezar@uow.edu.au

Willy Susilo
School of Computer Science and Software Engineering
University of Wollongong, Australia
wsusilo@uow.edu.au

Jennifer Seberry
School of Computer Science and Software Engineering
University of Wollongong, Australia
jennie@uow.edu.au

ABSTRACT

In this paper, we propose an efficient method for extracting simple low-degree equations (e.g. quadratic ones) in addition to the linear ones, obtainable from the original cube attack by Dinur and Shamir at EUROCRYPT 2009. This extended cube attack can be successfully applied even to cryptosystems in which the original cube attack may fail due to the attacker's inability in finding sufficiently many independent *linear* equations. As an application of our extended method, we exhibit a side channel cube attack against the PRESENT block cipher using the Hamming weight leakage model. Our side channel attack improves upon the previous work of Yang, Wang and Qiao at CANS 2009 from two aspects. First, we use the Hamming weight leakage model which is a more relaxed leakage assumption, supported by many previously known practical results on side channel attacks, compared to the more challenging leakage assumption that the adversary has access to the "exact" value of the internal state bits as used by Yang et al. Thanks to applying the extended cube method, our attack has also a reduced complexity compared to that of Yang et al. Namely, for PRESENT-80 (80-bit key variant) as considered by Yang et al., our attack has a time complexity 2^{16} and data complexity of about 2^{13} chosen plaintexts; whereas, that of Yang et al. has time complexity of 2^{32} and needs about 2^{15} chosen plaintexts. Furthermore, our method directly applies to PRESENT-128 (i.e. 128-bit key variant) with time complexity of 2^{64} and the same data complexity of 2^{13} chosen plaintexts.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Code Breaking

General Terms

Security

Keywords

Algebraic cryptanalysis, cube attacks, extended cube, PRESENT, side channel attacks

1. INTRODUCTION

The cube attack, put forth by Dinur and Shamir at EUROCRYPT 2009 [16], is a generic type of algebraic attacks that may be applied against any cryptosytem, provided that the attacker has access to a bit of information that can be represented by a "low-degree" multivariate polynomial over GF(2) of the secret and public variables of the target cryptosytem. Dinur and Shamir in [16] compared the cube attack to some of the previously known similar techniques and stated that the attack generalizes and improves some of those methods. As some of the previously known similar attacks, which exploit the vulnerability of ciphers with low-degree polynomials, we refer to [36, 35].

The cube attack aims to derive low-degree (especially linear) implicit equations that can be exploited for constructing distinguishers, e.g. [4], and/or key recovery attacks, e.g. [16, 4]. 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. When using the original cube attack [16, 37], one tries to derive independent *linear* equations over secret variables of the cryptosystem. This system of linear

*Shekh Faisal Abdul-Latip is currently with the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka.

ASIACCS '11, March 22–24, 2011, Hong Kong, China.

equations can be easily solved to recover the value of the secret variables by using the well-known Gaussian elimination method.

This Paper. Our work is motivated by the observation that in most cases, for properly designed cryptographic algorithms, it may not be possible to extract a sufficient number of independent 'linear' equations using the (preprocessing phase of the) original cube attack. In fact various potential extensions of the cube attack were suggested to be considered for future research in [16]. One of the methods to generalize the original cube attack, left in [16] for future work without further elaboration, is that one should try to find and employ some additional low degree nonlinear equations, e.g. equations of degree 2 or 3, provided that the system of equations is simple (sparse and low degree) and solvable using existing methods. In this paper we elaborate this idea and develop an extension of the cube attack to extract such (low degree) nonlinear equations. To demonstrate the application of our extended cube method, we provide a side channel cube attack against the PRESENT block cipher [10], which improves upon the previous work of Yang, Wang and Qiao at CANS 2009.

Side Channel Cube Attack. In attempting 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.

We note that the idea of combining algebraic cryptanalysis with side channel attacks was already introduced by Bogdanov, Kizhvatov and Pyshkin at INDOCRYPT 2008 [8], and also recently investigated in several other works such as [33, 17, 37]. Compared to the recent side channel cube attack of Yang et al. [37], our attack in this paper offers two improvements: it is based on a more relaxed leakage model; namely, the Hamming weight leakage model, and it has a better (i.e. reduced) complexity as well. The improved complexity is due to applying the extension of the cube attack, to derive simple low degree nonlinear (especially quadratic) equations, which itself is of an independent interest, as the primary contribution of our paper.

Comparing Side Channel Cube attack of [37] with Our attack. The leakage model used by Yang et al. [37] assumes that adversary has access to the exact value of some of the internal state bits after each round. We note that obtaining the exact value of the internal state bits in practice will require a probe station that allows the attacker to monitor the value of a specific bit position in the internal state during the encryption or decryption process. This implies an intrusive physical measurement and is known to involve a wide range of difficulties such as penetrating the device to access its internals and guessing which bit position is being recorded. To relax the leakage model, in contrast, we assume the Hamming weight leakage as a more common side channel leakage model, e.g. see [2, 7, 13, 14, 28].

From time and data complexity viewpoints, we show that, for PRESENT-80 (80-bit key variant of PRESENT), our attack has time complexity of 2^{16} and data complexity of about 2^{13} chosen plaintexts; whereas, the attack of Yang et al. has time complexity of 2^{32} and needs about 2^{15} chosen plaintexts. Also our method directly applies to PRESENT-128 (i.e. 128-bit key variant) with time complexity of 2^{64} and the same data complexity of 2^{13} chosen plaintexts, and is the only attack in this model considered against PRESENT-128.

We should stress that both of these side channel cube attacks against PRESENT, provided by Yang et al. in [37] and our attack in this paper, need clean leaked data values (i.e. the exact value of some internal state bits in the case of [37] and the Hamming weight of the internal state in our case). Hence, to the best of our knowledge these are only of a theoretical interest, at the moment, and do not directly impose any real threat to the security of PRESENT implementations in practice, where the side channel information measurements (e.g. power traces, EM radiations, or timing) are almost always noisy. We refer to [17, 33] for some related discussions on the possibility of handling the noisy data obtained from side channels when combined with the algebraic attacks. We note that, the current issue is that these methods are very sensitive to measurement noise levels and can only handle very low error rates than what may happen in practice.

Organization of the Paper. In Section 2 and 4, respectively, we review the cube attack and the construction of the PRESENT block cipher. Section 3 and 5 contain the main contribution of this paper, where we provide the notion of an extended cube for extracting nonlinear equation of low degree and the details of the improved side channel cube attack on PRESENT. Section 6 concludes the paper.

2. A REVIEW ON THE CUBE ATTACK

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_is with $i \in I$. By factoring the master polynomial p by the monomial t_I, we have:

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

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_is with $i \in I$ and fixing the value of all the remaining x_is with $i \notin I$, the resultant polynomial equals $p_{S(I)} \pmod 2$. 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_is with $i \in I$, and leaving all the remaining variables (i.e. x_is with $i \notin I$) undetermined. For example, if $I = \{1, 2\}$ then

$C_I = \{(0, 0, x_3, \cdots, x_\ell), (0, 1, x_3, \cdots, x_\ell), (1, 0, x_3, \cdots, x_\ell), (1, 1, x_3, \cdots, x_\ell)\}$

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 from [16] states the main observation used by the cube attack.

Theorem 1. *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_is) in two steps, namely the preprocessing and online phases, which are described shortly.

Preprocessing Phase. During the preprocessing phase, the attacker first finds sufficiently many maxterms, i.e. t_Is, 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 [9] 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:

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

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_is 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_Is 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. EXTENDED CUBE: DERIVING LOW DEGREE EQUATIONS

3.1 The Main Observation

Based on the the output of the BLR linearity test [9], which is used in [16] during the process of finding a maxterm; i.e. t_I with associated linear superpoly $p_{S(I)}$, we can distinguish the following cases:

1. If the test output is "success", i.e., all (tested) vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy relation (2) then we have either:
   - *a superpoly that is constant 1*; i.e. $p_{S(I)} = 1$, which trivially satisfies relation (2) for any pair of vectors $\mathbf{x}$ and $\mathbf{y}$.
   - *a non-exist superpoly*, which may happen if t_I is neither a monomial nor a common subterm of some monomials in the master polynomial p, and hence cannot be factored out. This makes $p_I = 0$ for any pair of vectors $\mathbf{x}$ and $\mathbf{y}$, and hence both sides of equation (2) evaluate to 0.
   - *a linear superpoly*, which is the case if neither of the previous two cases (i.e. constant 1 or non-exist case) happens.
2. If the test output is "fail", i.e., at least one pair of vectors $\mathbf{x}$ and $\mathbf{y}$ is found not satisfying relation (2) then the superpoly $p_{S(I)}$ is *nonlinear*.

Note that the above observation is the basis for constructing a distinguisher as shown in [4], which was then formalized as Lemma 1 in [1] as shown below.

Lemma 1 ([1]). *Let p be a given (black-box) master polynomial over GF(2) in ℓ variables $x_1, \cdots, x_\ell$; $I \subseteq \{0, 1, \cdots, \ell\}$; $s = |I|$, and t_I denote the multiplication of x_is 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 (cf. Sec. 2). t_I exists 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 gives $p_I[\mathbf{x}] = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}}[\mathbf{x}] = 1$.*

Next, we propose an efficient method for deriving nonlinear equations of low degree. We introduce the notion of *extended cube* to efficiently extract nonlinear equations of degree D as follows.

Definition 1. *Adopting the notion of a boolean cube C_I (cf. Sec. 2), where $I \subseteq \{1, \cdots, \ell\}$ is the index subset and $s = |I|$, an extending cube C_K indexed by a subset $K \subseteq \{1, \cdots, \ell\}$ of size r (i.e. $r = |K|$) such that $I \cap K = \phi$, can be combined with the cube C_I to construct a larger extended cube as $C_{I \cup K}$ consisting of 2^{r+s} boolean vectors formed by assigning 2^{r+s} values to x_is with $i \in I \cup K$, and leaving all the remaining variables (i.e. x_is with $i \notin I \cup K$) undetermined.*

To illustrate Definition 1, let $I = \{1, 2\}$ and $K = \{3\}$, then

$C_{I \cup K} = \{(0, 0, 0, x_4, \cdots, x_\ell), (0, 0, 1, x_4, \cdots, x_\ell), \cdots, (1, 1, 1, x_4, \cdots, x_\ell)\}$

Any vector $\mathbf{w} \in C_{I \cup K}$ defines a derived polynomial $p_{|\mathbf{w}}$ with $\ell - (r + s)$ variables whose degree may be the same or lower than the degree of the master polynomial p. Summing the 2^{r+s} derived polynomials over GF(2) defined by the vectors in the extended cube $C_{I \cup K}$, we get a new polynomial $p_{(I \cup K)}$

defined by $p_{(I\cup K)} \triangleq \sum_{\mathbf{w}\in C_{I\cup K}} p_{|\mathbf{w}}$. Thus, we can revise the notion of tweakable master polynomial p in equation (1) as

$$p(x_1, ..., x_\ell) = t_I \cdot X_K \cdot p_{S(I\cup K)} + q(x_1, ..., x_\ell) \qquad (3)$$

where t_I is a subterm of size s over x_is with $i \in I$; X_K is a subterm of size r over x_is with $i \in K$, and $p_{S(I\cup K)}$ is the superpoly of $t_I \cdot X_K$ in p. Note that since we factored out both subterms t_I and X_K from p, the superpoly $p_{S(I\cup K)}$ does not contain any common variable with t_I and X_K, and each term t_J in the residue polynomial q misses at least one variable from $t_I \cdot X_K$. Now using the main theorem of the cube attack (namely, Theorem 1), if we sum p over '$t_I \cdot X_K$', by assigning all the possible combinations of 0/1 values to the x_is with $i \in I \cup K$ and fixing the value of all the remaining x_is with $i \notin I \cup K$, the resultant polynomial equals to $p_{S(I\cup K)} \pmod 2$; i.e. $p_{(I\cup K)} = p_{S(I\cup K)}$.

This observation enables the attacker to derive nonlinear superpoly equations of degree D over the secret variables k_is in two steps; namely the preprocessing and online phases, as described in the following.

3.2 Attack Phases

Preprocessing Phase. During the preprocessing phase to derive polynomial equations $p_{S(I)}$s of degree D, the degree d of the master polynomial p should be estimated in some way such as through the known structure of the cipher or using a variant of the random walk as proposed in [20]. Knowing the degree d of the master polynomial enables the attacker to know the size of t_I (i.e. $s = d - D$) in order to find the nonlinear superpoly of degree D. Next, the attacker finds many monomials t_Is, such that each t_I consists of a subset of public variables $v_1, \cdots, v_m$, and the corresponding superpoly $p_{S(I)}$ is a polynomial of degree D. To find those t_Is, the attacker chooses a monomial t_I of size s one at a time and performs the generalized version of the BLR test as proposed by Dinur and Shamir in [16] on $p_{S(I)}$ over the secret variables $k_1, \cdots, k_n$, while the value of the public variables v_is with $i \notin I$ are fixed (to 0 or 1). For example, if one uses this generalized version of the BLR test to capture the superpoly $p_{S(I)}$ of degree 2, κ sets of vectors $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \{0,1\}^n$, representing samples of n-bit secret keys, are chosen independently and uniformly at random, and then for each pair of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ the attacker sums the polynomial p over t_I to verify whether or not each one of them satisfies the relation:

$$p_{S(I)}[\mathbf{0}] + p_{S(I)}[\mathbf{x}] + p_{S(I)}[\mathbf{y}] + p_{S(I)}[\mathbf{z}] + p_{S(I)}[\mathbf{x}+\mathbf{y}] + p_{S(I)}[\mathbf{x}+\mathbf{z}] + p_{S(I)}[\mathbf{y}+\mathbf{z}] = p_{S(I)}[\mathbf{x}+\mathbf{y}+\mathbf{z}] \qquad (4)$$

Note that, the relation will capture all polynomials of degree $D \leq 2$. Hence, to obtain only the polynomials of degree 2, one should filter the linear, constant 1 and non-exist superpolys during the test. Having all κ sets of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ satisfying the relation (4), one can tell that the superpoly $p_{S(I)}$ (of the monomial t_I) is of degree at most 2, after repeating the test sufficiently many times, that is using a sufficient number of samples κ.

To derive efficiently a nonlinear equation $p_{S(I)}$ of degree D over secret variables k_is, one should identify the subset $S \subseteq \{1, \cdots, n\}$ that consists of the secret variable indexes within $p_{S(I)}$, in which each k_i with $i \in S$ is either a term or a subterm of $p_{S(I)}$. To do this, the subterm X_K (cf. equation (3)) is assigned with each secret variable $k_i \in \{k_1, \cdots, k_n\}$ one at a time while the subterm t_I is fixed to the monomial in which its superpoly $p_{S(I)}$ is of degree D, and all public variables v_is with $i \notin I$ are fixed to 0 or 1. For each assignment of X_K, we choose κ sets of vector $\mathbf{x} \in \{0,1\}^{n-1}$ representing samples of $n-1$ secret variables k_is with $i \notin K$ independently and uniformly at random, and verify that X_K (or similarly the secret variable k_i that is assigned to X_K) exists as a variable in the superpoly $p_{S(I)}$ if $p_{(I\cup K)}[\mathbf{x}] = \sum_{\mathbf{w}\in C_{I\cup K}} p_{|\mathbf{w}}[\mathbf{x}] = 1$ for at least an instance vector $\mathbf{x}$ due to Lemma 1. This procedure is shown by Algorithm 1 in Fig. 1.

Algorithm 1.

INPUT : n; // the total number of secret key variables
t_I; // the monomial in which $deg(p_{S(I)}) = D$
OUTPUT : S; // the set of secret variable indexes within $p_{S(I)}$
repeat
  assign X_K (cf. equation (3)) with a secret variable $k_i \in \{k_1, \cdots, k_n\}$ which has not been considered;
  choose κ vectors $\mathbf{x} \in \{0,1\}^{n-1}$ independently and uniformly at random;
  repeat
    choose one of κ vectors $\mathbf{x}$ which has not been used to represent $n-1$ secret variables $k_i \notin X_K$;
    compute $p_{(I\cup K)}[\mathbf{x}] = \sum_{\mathbf{w}\in C_{I\cup K}} p_{|\mathbf{w}}[\mathbf{x}]$;
  until $p_{(I\cup K)}[\mathbf{x}] = 1$ **or** all κ vectors $\mathbf{x}$ have been used;
  if $p_{(I\cup K)}[\mathbf{x}] = 1$ is the case **then**
    consider X_K as a variable within the superpoly $p_{S(I)}$ of degree D;
  endif
until all n secret variables k_is have been considered;

Figure 1: Finding secret variables within a superpoly equation

Having the set of secret variables k_is with $i \in S$ of the nonlinear superpoly $p_{S(I)}$ of degree D enables the attacker to derive the nonlinear equation over the secret variables by finding all terms of degrees $0, 1, \cdots, D$ within the superpoly equation.

Lemma 2. *The monomial t_I (cf. equation (1)) is a term in polynomial p if and only if fixing all variables $x_i \notin t_I$ to zero results in $p_I = \sum_{\mathbf{w}\in C_I} p_{|\mathbf{w}} = 1$. On the other hand, the monomial X_K (cf. equation (3)) is a term in the superpoly $p_{S(I)}$ if and only if fixing all variables $x_i \notin t_I \cdot X_K$ to zero results in $p_{(I\cup K)} = \sum_{\mathbf{w}\in C_{I\cup K}} p_{|\mathbf{w}} = 1$.*

Proof. If t_I is a term in the polynomial p, assigning all $x_i \notin t_I$ to zero will make t_I become the only remaining term in p; that is, the value of the superpoly $p_{S(I)}$ is evaluated to 1. From the fact that $p_I = p_{S(I)}$ due to Theorem 1, we have $p_I = 1$ which shows the existence of t_I as a term in p. On the other hand, if X_K is a term in the superpoly $p_{S(I)}$, assigning all variables $x_i \notin t_I \cdot X_K$ to zero will make $t_I \cdot X_K$ become the only remaining term in the polynomial p. Hence, the value of the superpoly $p_{S(I\cup K)}$ of $t_I \cdot X_K$ is evaluated to 1. Since $p_{(I\cup K)} = p_{S(I\cup K)}$ (cf. Sec. 3.1) then $p_{(I\cup K)} = 1$, which shows the existence of $t_I \cdot X_K$ as a term in p. Hence, the existence of $t_I \cdot X_K$ as a term in p implies that X_K exists as a term in the superpoly $p_{S(I)}$ because X_K (factored out from the superpoly $p_{S(I)}$) is a subterm of the term $t_I \cdot X_K$. □

Algorithm 2.

INPUT : S; // a set of indexes for secret variables k_is in a superpoly $p_{S(I)}$
D; // the degree of the superpoly $p_{S(I)}$
T; // a set of cube indexes for monomials of degree 1 until D over k_is with $i \in S$
$N = |S|$; // the number of secret variables k_is in $p_{S(I)}$
t_I; // in which $deg(p_{S(I)}) = D$
$r = 0$; // the initial size for terms in $p_{S(I)}$
OUTPUT: $p_{S(I)}$s; // the derived low degree nonlinear equations
repeat
  increase the size r by 1;
  repeat
    assign X_K with a monomial from T of size r which has not been considered;
    assign all variables $v_i, k_i \notin t_I \cdot X_K$ to 0s;
    if $p_{(I \cup K)} = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}} = 1$ **then**
      write X_K as a term in superpoly $p_{S(I)}$;
    endif
  until all $\binom{N}{r}$ terms have been considered;
until $r = D$;
assign public variables v_is with $i \notin I$ and secret variables $k_1, \cdots, k_n$ to 0s;
if $p_I = \sum_{\mathbf{w} \in C_I} p_{|\mathbf{w}} = 1$ **then**
  write '1' as a constant in superpoly $p_{S(I)}$;
endif

Figure 2: Deriving a nonlinear superpoly equation of degree D

Suppose $N = |S|$ is the number of secret variables k_is with $i \in S$ of the superpoly $p_{S(I)}$ of degree D. To derive $p_{S(I)}$, firstly we assign the subterm X_K one at a time with a monomial indexed by a subset $K \in T$ where T is a set of cube indexes of monomials constructed from all combinations of k_is from degree 1 until degree D with $i \in S$. For example, if $S = \{1, 2, 3\}$ and the degree of the superpoly $p_{S(I)}$ is 2, then $T = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}\}$.

In each assignment, all $v_i, k_i \notin t_I \cdot X_K$ are set to zero. Then to verify the existence of the monomial $X_K \in T$ as a term in $p_{S(I)}$, we compute $p_{(I \cup K)} = \sum_{\mathbf{w} \in C_{I \cup K}} p_{|\mathbf{w}}$. If the value of $p_{(I \cup K)}$ is equal to 1, then with probability 1, X_K is a term in the superpoly $p_{S(I)}$ due to Lemma 2. Finally, we determine whether there also exists a constant term (i.e. a term of degree 0) in the superpoly $p_{S(I)}$ by setting all public variables v_is for $i \notin I$ and all secret variables $k_1, \cdots, k_n$ to zero, and computing $p_I = \sum_{w \in C_I} p_{|\mathbf{w}}$. Similarly, if the value of p_I is equal to 1, then with probability 1, a constant term exists within the superpoly $p_{S(I)}$ due to Lemma 2. Thus, the procedure for finding all terms in the nonlinear superpoly $p_{S(I)}$ of degree D requires $\binom{N}{1} + \binom{N}{2} + ... + \binom{N}{D} + 1$ number of computations. The procedure is shown by Algorithm 2 in Fig. 2. The total complexity to deduce a nonlinear equation of degree D using Algorithm 1 and Algorithm 2 is

$$\kappa n 2^{d-D+1} + \sum_{i=0}^{D} 2^{d-D+i} \binom{N}{i}$$

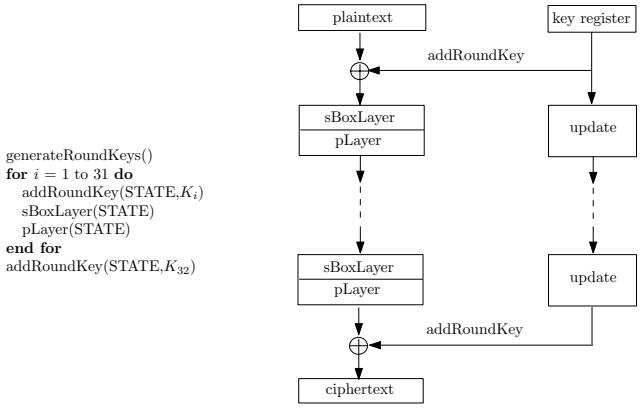

Figure 3: A top-level algorithmic description of 31-round PRESENT encryption.

Table 1: Specification of PRESENT S-box.

i	0	1	2	3	4	5	6	7
$\mathsf{S}[i]$	C	5	6	B	9	0	A	D
i	8	9	A	B	C	D	E	F
$\mathsf{S}[i]$	3	E	F	8	4	7	1	2

Online Phase. Once sufficiently many equations in the secret variables k_is have been found, the preprocessing phase is complete. In the online phase, the attacker's aim is to find the value of the right-hand side of each of the equations (both linear and nonlinear ones) by summing the black box polynomial p over the same set of t_Is obtained during the preprocessing phase. Now, the attacker can solve the resultant system of equations, e.g. by using the standard linearization method, a SAT solver [6], etc., to determine the values of the secret variables.

4. A BRIEF DESCRIPTION OF THE PRESENT BLOCK CIPHER

PRESENT [10] is a block cipher with a 64-bit block. The recommended key length is 80 bits, but a 128-bit key variant is also proposed. PRESENT produces a ciphertext after iterating a Substitution-Permutation Network (SP-Network) 31 times. In each round, a 64-bit round key $k_i = \kappa_{63}\kappa_{62} \cdots \kappa_0$, for $1 \leq i \leq 32$ is introduced to the current STATE $b_{63} \cdots b_0$ using addRoundKey as follows:

$$b_j \rightarrow b_j \oplus \kappa_j^i$$

for $0 \leq j \leq 63$. The round key K_{32} is used for post-whitening after the final round. The addition of round key k_i using the addRoundKey in each round always follows by sBoxLayer (i.e. a nonlinear substitution layer) and pLayer (i.e. a linear bitwise permutation). The graphical representation of PRESENT is shown in Fig. 3.

PRESENT uses a single 4-bit S-box S as shown in Table 1, which is applied 16 times in parallel in each round. The 4-bit nibble i at the input of an S-box is substituted by the 4-bit $\mathsf{S}[i]$ in output, i.e. $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$.

The pLayer which provide linear bitwise permutation is shown in Table 2. During the permutation, each bit i of

Table 2: Specification of the PRESENT permutation layer.

i	0	1	2	3	4	5	6	7
$P(i)$	0	16	32	48	1	17	33	48
i	8	9	10	11	12	13	14	15
$P(i)$	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23
$P(i)$	4	20	36	52	5	21	37	53
i	24	25	26	27	28	29	30	31
$P(i)$	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39
$P(i)$	8	24	40	56	9	25	41	57
i	40	41	42	43	44	45	46	47
$P(i)$	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55
$P(i)$	12	28	44	60	13	29	45	61
i	56	57	58	59	60	61	62	63
$P(i)$	14	30	46	62	15	31	47	63

STATE is moved to bit position $P(i)$.

The difference between the 80-bit key and the 128-bit key variants of PRESENT is on the key schedule.

For the 80-bit key variant, the user-supplied key that is stored in key register K is represented as $k_{79}k_{78}\cdots k_0$. The 64-bit round key $K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0$ consists of the leftmost bits of the current contents (i.e. at round i) of register K. Thus the round key at round i can be depicted as:

$$K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0 = k_{79}k_{78}\cdots k_{16}$$

For each round i, after applying addRoundKey, the key register K is updated as in the following steps:

1. $[k_{79}k_{78}\cdots k_1k_0] = [k_{18}k_{17}\cdots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ `round_counter`

For the 128-bit key variant, the user supplied key that stored in a key register K is represented as $k_{127}k_{126}\cdots k_0$. Similarly, the 64-bit round key $K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0$ consists of the leftmost bits of the current contents (i.e. at round i) of register K. Thus the round key at round i for 128-bit key variant can be depicted as:

$$K_i = \kappa_{63}\kappa_{62}\cdots\kappa_0 = k_{127}k_{126}\cdots k_{64}$$

After applying addRoundKey in each round, the key register K is updated as in the following steps:

1. $[k_{127}k_{126}\cdots k_1k_0] = [k_{66}k_{65}\cdots k_{68}k_{67}]$
2. $[k_{127}k_{126}k_{125}k_{124}] = S[k_{127}k_{126}k_{125}k_{124}]$
3. $[k_{123}k_{122}k_{121}k_{120}] = S[k_{123}k_{122}k_{121}k_{120}]$
4. $[k_{66}k_{65}k_{64}k_{63}k_{62}] = [k_{66}k_{65}k_{64}k_{63}k_{62}] \oplus$ `round_counter`

5. SIDE CHANNEL CUBE ATTACK ON PRESENT

In this section, we explain our side channel cube attack against PRESENT using the Hamming weight leakage model and our extended cube method.

HAMMING WEIGHT. Let $B = b_{\beta-1}\cdots b_0$ be the binary string of length β bits, representing the internal state of the cipher. The Hamming weight of B is the number of bits with value 1 in the binary representation of B, which can be computed as $HW(B) = \sum_{j=0}^{\beta-1} b_j$ and has a value between 0 and β. Clearly, the Hamming weight can also be viewed as a boolean vector mapping $HW : \{0,1\}^b \rightarrow \{0,1\}^{\lfloor \log_2 b \rfloor + 1}$, where the first bit of $HW(B)$, i.e. the least significant bit (LSB), is the XOR of all bits from B; the MSB of $HW(B)$ is the AND of all bits from B; and each bit in between is a boolean function whose degree increases as the bit position gets closer to the MSB.

5.1 Application to the PRESENT Block Cipher

The difference between the two variants of PRESENT is in the key schedule algorithm as described in Sec. 4. Note that, at each round i the addition of the round key k_i to the internal state using addRoundKey is performed before updating the key register K. Therefore, the first 64 bits of the secret key for both variants (i.e. $k_{79}\cdots k_{16}$ for the 80-bit key variant PRESENT-80 and $k_{127}\cdots k_{64}$ for the 128-bit key variant PRESENT-128) are applied directly to the internal state at the first round. Hence, both variants are equally susceptible to any attack that can recover the first 64 bits of the secret key, e.g. by considering the Hamming weight leakage after the first round as is the case for our attack.

Assuming access to the value of the Hamming weight of the internal state after the first round, which can be represented as a byte, we consider all bits starting from the LSB position towards the MSB position (of the 8-bit binary representation of the Hamming weight). To search for maxterms, we use various cube sizes which increase as we consider more bits towards the MSB. We follow the technique which is a variant of the random walk [20] as used also in [16] to find the implicit degree d of the master polynomial p. Knowing the degree d of the master polynomial enables us to find the maxterms easily, as the maxterms are expected to be found when we choose the cube of size $s = d - 1$. To know whether a particular selected monomial t_I is a maxterm, we apply the BLR test by choosing 100 pairs of vectors $\mathbf{x}$, $\mathbf{y} \in \{0,1\}^n$ (where n is the length of the secret key which is either 80 for PRESENT-80 or 128 for PRESENT-128), and verify t_I as a maxterm only if all the 100 pair of vectors ($\mathbf{x}$, $\mathbf{y}$) satisfy relation (2). The linear equation is then derived by finding all terms within the equation. To find whether a secret variable k_i is a term of the linear equation, we check whether the value of $p_I = \sum_{\mathbf{w}\in C_I} p_{|\mathbf{w}}$ will change if we change the value of k_i to either 0 or 1, while setting all other variables (i.e. secret and public variables) which are not in the monomial t_I to fixed values. If changing the value of the secret variable k_i changes the value of p_I, one can tell that k_i is a term in the linear equation. To check whether a constant term exists in the equation, we set all the variables not in the monomial t_I to zero and compute p_I and check if it is equal to 1 that indicates the existence of the constant term in the equation.

We ran our simulation for several weeks to find maxterms; i.e., t_Is with "linear" superpoly. However, it was not possible to find enough linearly independent equations. Then, using the extended cube method, we continued finding some additional quadratic equations. To do this we chose the cube of size $s = d - 2$. This time, to know whether a particular monomial t_I which we select has a superpoly $p_{S(I)}$ of degree 2, we apply the generalized BLR test (cf. Sec. 3.1) by selecting 100 samples of vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in \{0,1\}^n$ and verify that $p_{S(I)}$ is of degree at most 2 only if all the 100 samples of vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ satisfy relation (4). The quadratic equation is then derived using two steps. First, we find the secret variables k_is that exist within the quadratic equation using Algorithm 1 where we choose 300 (i.e. $\kappa = 300$) samples of vector $\mathbf{x}$ independently and uniformly at random (cf. Lemma 1 and Algorithm 1 for more detail). Secondly, we derive all terms of degree 0 (constant terms), degree 1 and degree 2 within the quadratic equation using Algorithm 2.

After searching for nonlinear superpoly equations of degree 2, we have been able to find 62 simple quadratic equations, which combined with 32 independent linear equations (among the previously obtained set of linear equations), can provide a unique solution for all the 64 secret key variables. The system of equations can then be easily solved by linearizing the system and solving it using the well-known Gaussian elimination method. Table 3 and Table 4, which are provided in the Appendix, show the sets of equations which can provide the unique solution for the 64 secret key variables, respectively, for PRESENT-80 and PRESENT-128.

Having this set of equations, the preprocessing phase is completed and the attacker during the online phase should find the value of the right-hand side of each equation by summing the master polynomials (corresponding the second and third bits of the Haming weight) over each cube which is listed in Table 3 and Table 4. Since we have 2 cubes of size 2, 62 cubes of size 4 and 30 cubes of size 8, then we require $2 \times 2^2 + 62 \times 2^4 + 30 \times 2^8 \approx 2^{13}$ chosen plaintexts to find the value of right-hand side of all 94 equations. Considering that we have been able to find a unique solution for all the 64 secret key variables, the total time complexity to find the correct 80-bit key and 128-bit key reduces to 2^{16} and 2^{64} respectively.

6. CONCLUSIONS

We showed an extension to the cube attack to efficiently derive low degree nonlinear superpoly equations to help improve the success level in a key recovery attack, especially in the cases that only limited number of maxterms can be found by the original cube attack. We also investigated the security of the PRESENT block cipher against side channel cube attack, assuming the Hamming weight leakage model and applying our extended cube attack. Our attack improves upon the previous attack of Yang et al. on PRESENT-80, from both leakage model and complexity viewpoints. Improving the attack against PRESENT-128, e.g. by combining more leakage information from several other rounds of the cipher, and investigating the applicability of the proposed extended cube attack to other ciphers, especially in the traditional cryptanalytical framework (without side channel leakage assumptions), are proposed as interesting works for future research.

7. REFERENCES

[1] Abdul-Latip, S.F., Reyhanitabar, M.R., Susilo, W., Seberry, J.: On the Security of NOEKEON against Side Channel Cube Attacks. In: Kwak, J., Deng, R., Won, Y. (Eds.) ISPEC 2010. LNCS, vol. 6047, pp. 45–55. Springer, Heidelberg (2010)

[2] Akkar, M.L., Bévan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible... In: Okamoto, T. (Ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489-Ű502. Springer, Heidelberg (2000)

[3] Anderson, R., Biham, B., Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. In First Advanced Encryption Standard (AES) Conference, (1998)

[4] Aumasson, J.P., Dinur, I., Meier, M., Shamir, A.: Cube Testers and Key Recovery Attacks On Reduced-Round MD6 and Trivium. In: Dunkelman, O. (Ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009)

[5] Aumasson, J.P., Dinur, I., Henzen, L., Meier, W., Shamir, A.: Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. IACR ePrint Archive, Report 2009/218 (2009), http://eprint.iacr.org/2009/218

[6] Bard, G.V., Courtois, N.T., Jefferson, C.: Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers. IACR ePrint Archive, Report 2007/024 (2007), http://eprint.iacr.org/2007/024

[7] Bévan, R., Knudsen, R.: Ways to Enhance Differential Power Analysis. In: Lee, P.J, Lim, C.H. (Eds.) ICISC 2002. LNCS, vol. 2587, pp. 327-Ű342. Springer, Heidleberg (2003)

[8] Bogdanov, A., Kizhvatov, I., Pyshkin, A.: Algebraic Methods in Side-Channel Collision Attcks and Practical Collision Detection. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 251–265. Springer, Heidelberg (2008)

[9] Blum, M., Luby, M., Rubinfield, R.: Self-Testing/Correcting with Application to Numerical Problems. In: STOC, pp. 73–83. ACM, New York (1990)

[10] 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)

[11] Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (Eds.) CHES 2004, LNCS, vol. 3156, pp. 16-Ű29. Springer, Heidelberg (2004)

[12] Canniere, C.D., Preneel, B.: TRIVIUM. In: Robshaw, M.J.B., Billet, O. (Ed.) New Stream Cipher Designs - The eSTREAM Finalists. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008)

[13] Clavier, C., Coron, J.S., Dabbous, N.: Differential Power Analysis in the Presence of Hardware Countermeasures. In: Koç, Ç.K., Paar, C. (Eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer, Heidelberg (2000)

[14] Coron, J.S., Kocher, P., Naccache, D.: Statistics and

Secret Leakage. In: Frankel, Y. (Ed.) FC 2000. LNCS, vol. 1962, pp. 157–173. Springer, Heidelberg (2001)
[15] Daemen, J., Rijmen, V.: AES Proposal: Rijndael. Technical Evaluation, CD-1: Documentation (1998)
[16] 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)
[17] Dinur, I., Shamir, A.: Side Channel Cube Attacks on Block Ciphers. Cryptology ePrint Archive, Report 2009/127 (2009), http://eprint.iacr.org/2009/127
[18] Englund, H., Johansson, T., Turan, M.S.: A Framework for Chosen IV Statistical Analysis of Stream Ciphers. In: Srinathan, K., Rangan, C.P., Yung, M. (Eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 268Ű-281. Springer, Heidelberg (2007)
[19] Filiol, E.: A New Statistical Testing for Symmetric Ciphers and Hash Functions. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (Eds.) ICICS 2002. LNCS, vol. 2513, pp. 342Ű-353. Springer, Heidelberg (2002)
[20] Fischer, S., Khazaei, S., Meier, W.: Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In: Vaudenay, S. (Ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 236–245. Springer, Heidelberg (2008)
[21] Goubin, L., Patarin, J.: DES and Differential Power Analysis - The Duplication Method. In: Koç, Ç.K., Paar, C. (Eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
[22] Hell, M., Johansson, T., Meier, M.: The Grain Family of Stream Ciphers. In: Robshaw, M., Billet, O. (Eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
[23] Khazaei, S., Meier, W.: New Directions in Cryptanalysis of Self-Synchronizing Stream Ciphers. In: Chowdhury, D.R., Rijmen, V., Das, A. (Eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 15Ű-26. Springer, Heidelberg (2008)
[24] Kocher, J., Jaffe, J., Jun, B.: Introduction to Differential Power Analysis and Related Attacks. (1998), http://www.cryptography.com/dpa/technical.
[25] Kocher, J., Jaffe, J., Jun, B.: Differential Power Analysis. In: Weiner, M.J. (Ed.) CRYPTO 99. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
[26] 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)
[27] Mangard, S.: Hardware Countermeasures against DPA Ű A Statistical Analysis of Their Effectiveness. In: Okamoto, T. (Ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 222–235. Springer, Heidelberg (2004)
[28] Mayer-Sommer, R.: Smartly Analysis the Simplicity and the Power of Simple Power Analysis on Smartcards. In: Koç, Ç.K., Paar, C. (Eds.) CHES 2000. LNCS, vol. 1965, pp. 78–92. Springer, Heidelberg (2000)
[29] OŠNeil, S.: Algebraic Structure Defectoscopy. IACR ePrint Archive, Report 2007/378 (2007), http://eprint.iacr.org/2007/378
[30] Oswald, E.: On Side-Channel Attacks and the Application of Algorithmic Countermeasures. PhD Thesis, Faculty of Science of the University of Technology Graz (IAIK-TUG), Austria, May (2003)
[31] Oswald, E., Mangard, S., Herbst, C., Tillich, S.: Practical Second-Order DPA Attacks for Masked Smart Card Implementations of Block Ciphers. In: Pointcheval, D. (Ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 192–207. Springer, Heidelberg (2006)
[32] 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/
[33] Renauld, M., Standaert, F.X.: Algebraic Side-Channel Attacks. IACR ePrint Archive, Report 2009/279 (2009), http://eprint.iacr.org/2009/279
[34] Saarinen, M.-J.O.: Chosen-IV Statistical Attacks on eStream Ciphers. In: Malek, M., Fernt'andez-Medina, E., Hernando, J. (Eds.) SECRYPT 2006, pp. 260Ű-266. INSTICC Press (2006)
[35] Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. IACR ePrint Archive, Report 2007/413 (2007), http://eprint.iacr.org/2007/413
[36] Lai, X.: Higher Order Derivatives and Differential Cryptanalysis. In: Communication and Cryptology, pp. 227–233. Kluwer Academic Publisher (1994)
[37] Yang, L., Wang, M., Qiao, S.: Side Channel Cube Attack on PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (Eds.) CANS 2009. LNCS, vol. 5888, pp. 379–391. Springer, Heidelberg (2009)

APPENDIX

Table 3: Hamming weight bit position, cube indexes and the superpoly equations for PRESENT-80 from the leakage after the first round.

HW Bit	Cube Indexes	Superpoly Equation
2	{59,58}	$k_{72} + k_{75}$
2	{63,60,55,54}	$k_{79} + k_{78} + k_{69} + k_{79}k_{69} + k_{78}k_{69} + 1$
2	{63,61,10,8}	$k_{77}k_{28}$
2	{63,61,11,9}	$k_{77}k_{25}$
2	{63,61,15,12}	$k_{77} + k_{77}k_{31} + k_{77}k_{30}$
2	{63,61,15,13}	$k_{77}k_{29}$
2	{63,61,18,16}	$k_{77}k_{36}$
2	{63,61,19,17}	$k_{77}k_{33}$
2	{63,61,2,0}	$k_{77}k_{20}$
2	{63,61,23,20}	$k_{77} + k_{77}k_{39} + k_{77}k_{38}$
2	{63,61,23,21}	$k_{77}k_{37}$
2	{63,61,26,24}	$k_{77}k_{44}$
2	{63,61,27,25}	$k_{77}k_{41}$
2	{63,61,3,1}	$k_{77}k_{17}$

Table 3: (continued)

HW Bit	Cube Indexes	Superpoly Equation
2	{63,61,31,28}	$k_{77} + k_{77}k_{47} + k_{77}k_{46}$
2	{63,61,31,29}	$k_{77}k_{45}$
2	{63,61,34,32}	$k_{77}k_{52}$
2	{63,61,35,33}	$k_{77}k_{49}$
2	{63,61,39,36}	$k_{77} + k_{77}k_{55} + k_{77}k_{54}$
2	{63,61,39,37}	$k_{77}k_{53}$
2	{63,61,42,40}	$k_{77}k_{60}$
2	{63,61,43,41}	$k_{77}k_{57}$
2	{63,61,47,44}	$k_{77} + k_{77}k_{63} + k_{77}k_{62}$
2	{63,61,47,45}	$k_{77}k_{61}$
2	{63,61,50,48}	$k_{77}k_{68}$
2	{63,61,51,49}	$k_{77}k_{65}$
2	{63,61,55,52}	$k_{77} + k_{77}k_{71} + k_{77}k_{70}$
2	{63,61,55,53}	$k_{77}k_{69}$
2	{63,61,57,56}	$k_{77} + k_{77}k_{76}$
2	{63,61,59,57}	$k_{77}k_{73}$
2	{63,61,7,4}	$k_{77} + k_{77}k_{23} + k_{77}k_{22}$
2	{63,61,7,5}	$k_{77}k_{21}$
2	{63,62,1,0}	$k_{77} + k_{20} + k_{77}k_{20} + 1$
2	{63,62,11,10}	$k_{77} + k_{25} + k_{77}k_{25} + 1$
2	{63,62,15,12}	$k_{77} + k_{31} + k_{30} + k_{77}k_{31} + k_{77}k_{30} + 1$
2	{63,62,15,14}	$k_{77} + k_{29} + k_{77}k_{29} + 1$
2	{63,62,18,16}	$k_{36} + k_{77}k_{36}$
2	{63,62,19,18}	$k_{77} + k_{33} + k_{77}k_{33} + 1$
2	{63,62,23,20}	$k_{77} + k_{39} + k_{38} + k_{77}k_{39} + k_{77}k_{38} + 1$
2	{63,62,23,22}	$k_{77} + k_{37} + k_{77}k_{37} + 1$
2	{63,62,25,24}	$k_{77} + k_{44} + k_{77}k_{44} + 1$
2	{63,62,27,26}	$k_{77} + k_{41} + k_{77}k_{41} + 1$
2	{63,62,3,2}	$k_{77} + k_{17} + k_{77}k_{17} + 1$
2	{63,62,31,28}	$k_{77} + k_{47} + k_{46} + k_{77}k_{47} + k_{77}k_{46} + 1$
2	{63,62,31,30}	$k_{77} + k_{45} + k_{77}k_{45} + 1$
2	{63,62,33,32}	$k_{77} + k_{52} + k_{77}k_{52} + 1$
2	{63,62,35,34}	$k_{77} + k_{49} + k_{77}k_{49} + 1$
2	{63,62,39,36}	$k_{77} + k_{55} + k_{54} + k_{77}k_{55} + k_{77}k_{54} + 1$
2	{63,62,39,38}	$k_{77} + k_{53} + k_{77}k_{53} + 1$
2	{63,62,42,40}	$k_{60} + k_{77}k_{60}$
2	{63,62,43,42}	$k_{77} + k_{57} + k_{77}k_{57} + 1$
2	{63,62,47,44}	$k_{77} + k_{63} + k_{62} + k_{77}k_{63} + k_{77}k_{62} + 1$
2	{63,62,47,45}	$k_{61} + k_{77}k_{61}$
2	{63,62,50,48}	$k_{68} + k_{77}k_{68}$
2	{63,62,51,49}	$k_{65} + k_{77}k_{65}$
2	{63,62,55,52}	$k_{77} + k_{71} + k_{70} + k_{77}k_{71} + k_{77}k_{70} + 1$
2	{63,62,55,53}	$k_{69} + k_{77}k_{69}$
2	{63,62,55,54}	$k_{77} + k_{69} + k_{77}k_{69} + 1$
2	{63,62,57,56}	$k_{77} + k_{76} + k_{77}k_{76} + 1$
2	{63,62,59,58}	$k_{77} + k_{73} + k_{77}k_{73} + 1$

Table 3: (continued)

HW Bit	Cube Indexes	Superpoly Equation
2	{63,62,7,4}	$k_{77} + k_{23} + k_{22} + k_{77}k_{23} + k_{77}k_{22} + 1$
2	{63,62,7,5}	$k_{21} + k_{77}k_{21}$
2	{63,62,9,8}	$k_{77} + k_{28} + k_{77}k_{28} + 1$
2	{63,62}	$k_{76} + k_{79}$
3	{63,62,61,60,57,52,51,49}	$k_{73} + k_{74} + 1$
3	{63,62,61,60,58,52,51,49}	k_{72}
3	{63,62,61,60,59,57,11,9}	k_{27}
3	{63,62,61,60,59,57,12,10}	k_{24}
3	{63,62,61,60,59,57,12,9}	$k_{25} + k_{26} + 1$
3	{63,62,61,60,59,57,15,13}	k_{31}
3	{63,62,61,60,59,57,19,17}	k_{35}
3	{63,62,61,60,59,57,20,17}	$k_{33} + k_{34} + 1$
3	{63,62,61,60,59,57,20,18}	k_{32}
3	{63,62,61,60,59,57,23,21}	k_{39}
3	{63,62,61,60,59,57,27,25}	k_{43}
3	{63,62,61,60,59,57,28,25}	$k_{41} + k_{42} + 1$
3	{63,62,61,60,59,57,28,26}	k_{40}
3	{63,62,61,60,59,57,3,1}	k_{19}
3	{63,62,61,60,59,57,31,29}	k_{47}
3	{63,62,61,60,59,57,35,33}	k_{51}
3	{63,62,61,60,59,57,36,33}	$k_{49} + k_{50} + 1$
3	{63,62,61,60,59,57,36,34}	k_{48}
3	{63,62,61,60,59,57,39,37}	k_{55}
3	{63,62,61,60,59,57,4,1}	$k_{17} + k_{18} + 1$
3	{63,62,61,60,59,57,4,2}	k_{16}
3	{63,62,61,60,59,57,43,41}	k_{59}
3	{63,62,61,60,59,57,44,41}	$k_{57} + k_{58} + 1$
3	{63,62,61,60,59,57,44,42}	k_{56}
3	{63,62,61,60,59,57,47,45}	k_{63}
3	{63,62,61,60,59,57,51,49}	k_{67}
3	{63,62,61,60,59,57,52,49}	$k_{65} + k_{66} + 1$
3	{63,62,61,60,59,57,52,50}	k_{64}
3	{63,62,61,60,59,57,55,53}	k_{71}
3	{63,62,61,60,59,57,7,5}	k_{23}

Table 4: Hamming weight bit position, cube indexes and the superpoly equations for PRESENT-128 from the leakage after the first round.

HW Bit	Cube Indexes	Superpoly Equation
2	{63,62,25,24}	$k_{125} + k_{92} + k_{125}k_{92} + 1$
2	{63,62,27,26}	$k_{125} + k_{89} + k_{125}k_{89} + 1$
2	{63,62,3,2}	$k_{125} + k_{65} + k_{125}k_{65} + 1$
2	{63,62,31,28}	$k_{125} + k_{95} + k_{94} + k_{125}k_{95} + k_{125}k_{94} + 1$
2	{63,62,31,30}	$k_{125} + k_{93} + k_{125}k_{93} + 1$
2	{63,62,33,32}	$k_{125} + k_{100} + k_{125}k_{100} + 1$
2	{63,62,35,34}	$k_{125} + k_{97} + k_{125}k_{97} + 1$
2	{59,58}	$k_{120} + k_{123}$
2	{63,60,55,54}	$k_{127} + k_{126} + k_{117} + k_{127}k_{117} + k_{126}k_{117} + 1$
2	{63,61,11,9}	$k_{125}k_{73}$
2	{63,61,15,12}	$k_{125} + k_{125}k_{79} + k_{125}k_{78}$

Table 4: (continued)

HW Bit	Cube Indexes	Superpoly Equation
2	{63,61,15,13}	$k_{125}k_{77}$
2	{63,61,18,16}	$k_{125}k_{84}$
2	{63,61,19,17}	$k_{125}k_{81}$
2	{63,61,2,0}	$k_{125}k_{68}$
2	{63,61,23,20}	$k_{125} + k_{125}k_{87} + k_{125}k_{86}$
2	{63,61,23,21}	$k_{125}k_{85}$
2	{63,61,26,24}	$k_{125}k_{92}$
2	{63,61,27,25}	$k_{125}k_{89}$
2	{63,61,3,1}	$k_{125}k_{65}$
2	{63,61,31,28}	$k_{125} + k_{125}k_{95} + k_{125}k_{94}$
2	{63,61,31,29}	$k_{125}k_{93}$
2	{63,61,34,32}	$k_{125}k_{100}$
2	{63,61,35,33}	$k_{125}k_{97}$
2	{63,61,39,36}	$k_{125} + k_{125}k_{103} + k_{125}k_{102}$
2	{63,61,39,37}	$k_{125}k_{101}$
2	{63,61,42,40}	$k_{125}k_{108}$
2	{63,61,43,41}	$k_{125}k_{105}$
2	{63,61,10,8}	$k_{125}k_{76}$
2	{63,61,47,44}	$k_{125} + k_{125}k_{111} + k_{125}k_{110}$
2	{63,61,47,45}	$k_{125}k_{109}$
2	{63,61,50,48}	$k_{125}k_{116}$
2	{63,61,51,49}	$k_{125}k_{113}$
2	{63,61,55,52}	$k_{125} + k_{125}k_{119} + k_{125}k_{118}$
2	{63,61,55,53}	$k_{125}k_{117}$
2	{63,61,57,56}	$k_{125} + k_{125}k_{124}$
2	{63,61,59,57}	$k_{125}k_{121}$
2	{63,61,7,4}	$k_{125} + k_{125}k_{71} + k_{125}k_{70}$
2	{63,61,7,5}	$k_{125}k_{69}$
2	{63,62,1,0}	$k_{125} + k_{68} + k_{125}k_{68} + 1$
2	{63,62,11,10}	$k_{125} + k_{73} + k_{125}k_{73} + 1$
2	{63,62,15,12}	$k_{125} + k_{79} + k_{78} + k_{125}k_{79} + k_{125}k_{78} + 1$
2	{63,62,15,14}	$k_{125} + k_{77} + k_{125}k_{77} + 1$
2	{63,62,18,16}	$k_{84} + k_{125}k_{84}$
2	{63,62,19,18}	$k_{125} + k_{81} + k_{125}k_{81} + 1$
2	{63,62,23,20}	$k_{125} + k_{87} + k_{86} + k_{125}k_{87} + k_{125}k_{86} + 1$
2	{63,62,23,22}	$k_{125} + k_{85} + k_{125}k_{85} + 1$
2	{63,62,39,36}	$k_{125} + k_{103} + k_{102} + k_{125}k_{103} + k_{125}k_{102} + 1$
2	{63,62,39,38}	$k_{125} + k_{101} + k_{125}k_{101} + 1$
2	{63,62,42,40}	$k_{108} + k_{125}k_{108}$
2	{63,62,43,42}	$k_{125} + k_{105} + k_{125}k_{105} + 1$
2	{63,62,47,44}	$k_{125} + k_{111} + k_{110} + k_{125}k_{111} + k_{125}k_{110} + 1$
2	{63,62,47,45}	$k_{109} + k_{125}k_{109}$
2	{63,62,50,48}	$k_{116} + k_{125}k_{116}$
2	{63,62,51,49}	$k_{113} + k_{125}k_{113}$
2	{63,62,55,52}	$k_{125} + k_{119} + k_{118} + k_{125}k_{119} + k_{125}k_{118} + 1$
2	{63,62,55,53}	$k_{117} + k_{125}k_{117}$
2	{63,62,55,54}	$k_{125} + k_{117} + k_{125}k_{117} + 1$
2	{63,62,57,56}	$k_{125} + k_{124} + k_{125}k_{124} + 1$
2	{63,62,59,58}	$k_{125} + k_{121} + k_{125}k_{121} + 1$
2	{63,62,7,4}	$k_{125} + k_{71} + k_{70} + k_{125}k_{71} + k_{125}k_{70} + 1$

Table 4: (continued)

HW Bit	Cube Indexes	Superpoly Equation
2	{63,62,7,5}	$k_{69} + k_{125}k_{69}$
2	{63,62,9,8}	$k_{125} + k_{76} + k_{125}k_{76} + 1$
2	{63,62}	$k_{124} + k_{127}$
3	{63,62,61,60,57,52,51,49}	$k_{121} + k_{122} + 1$
3	{63,62,61,60,58,52,51,49}	k_{120}
3	{63,62,61,60,59,57,11,9}	k_{75}
3	{63,62,61,60,59,57,12,10}	k_{72}
3	{63,62,61,60,59,57,12,9}	$k_{73} + k_{74} + 1$
3	{63,62,61,60,59,57,15,13}	k_{79}
3	{63,62,61,60,59,57,19,17}	k_{83}
3	{63,62,61,60,59,57,20,17}	$k_{81} + k_{82} + 1$
3	{63,62,61,60,59,57,20,18}	k_{80}
3	{63,62,61,60,59,57,23,21}	k_{87}
3	{63,62,61,60,59,57,27,25}	k_{91}
3	{63,62,61,60,59,57,28,25}	$k_{89} + k_{90} + 1$
3	{63,62,61,60,59,57,28,26}	k_{88}
3	{63,62,61,60,59,57,3,1}	k_{67}
3	{63,62,61,60,59,57,31,29}	k_{95}
3	{63,62,61,60,59,57,35,33}	k_{99}
3	{63,62,61,60,59,57,36,33}	$k_{97} + k_{98} + 1$
3	{63,62,61,60,59,57,36,34}	k_{96}
3	{63,62,61,60,59,57,39,37}	k_{103}
3	{63,62,61,60,59,57,4,1}	$k_{65} + k_{66} + 1$
3	{63,62,61,60,59,57,4,2}	k_{64}
3	{63,62,61,60,59,57,43,41}	k_{107}
3	{63,62,61,60,59,57,44,41}	$k_{105} + k_{106} + 1$
3	{63,62,61,60,59,57,44,42}	k_{104}
3	{63,62,61,60,59,57,47,45}	k_{111}
3	{63,62,61,60,59,57,51,49}	k_{115}
3	{63,62,61,60,59,57,52,49}	$k_{113} + k_{114} + 1$
3	{63,62,61,60,59,57,52,50}	k_{112}
3	{63,62,61,60,59,57,55,53}	k_{119}
3	{63,62,61,60,59,57,7,5}	k_{71}