Cube attack in finite fields of higher order

Andrea Agnesse\textsuperscript{1} \quad Marco Pedicini\textsuperscript{2}

\begin{itemize}
  \item \textsuperscript{1} Dipartimento di Matematica, Università Roma Tre
  Largo San Leonardo Murialdo 1, Rome, Italy
  \item \textsuperscript{2} Istituto per le Applicazioni del Calcolo “Mauro Picone”
  Consiglio Nazionale delle Ricerche
  Viale Manzoni 30, 00185 Rome, Italy
  \end{itemize}

Email: marco@iac.cnr.it

Abstract

We present in full details a version of the Dinur-Shamir Cube Attack (Dinur & Shamir 2009) for a generic finite field of order \( q \). In particular, when applied to multivariate monomials of degree \( d \) in \( k < d \) variables, the attack acts exactly in the same way if the selected monomial was using the degree \( k \) monomial in the same \( k \) variables.

Keywords: Algebraic cryptanalysis, Cube Attack.

1 Introduction

The Cube Attack is a new cryptographic attack based on multivariate polynomials over \( \mathbb{F}_2 \) suitable for both block and stream ciphers. In (Dinur & Shamir 2009), authors introduced this methodology as a variant of algebraic attacks at aiming a way to distill from a cryptographic encoding function a set of linear relations involving secret parameters (e.g., key bits) by means of tweakable ones (e.g., plaintext or initial vectors). The basic requirement for the attack is the possibility of describing the cryptographic scheme as a function in \( m + n \) variables that can be partitioned in public \( x_1, \ldots, x_n \) (i.e., that can be chosen during the attack, therefore tweakable in accord with (Dinur & Shamir 2009)) and private variables \( k_1, \ldots, k_m \) (i.e., those variables that have to be determined during the attack). Note that public variables can represent bits of the initial vector but in other scenarios, they could be bits of the key, or bits of the plaintext, see for instance (Aumasson et al. 2009), (Joux 2009).

As usual, the goal of the attack is to find the value of the private variables: by obtaining from the enciphering function enough linear relations that have to be satisfied by these variables and having a way to connect them with ciphertext. Whenever the number of independent linear relations is equal to the number of variables, the system can be solved. In order to do this the attacker has to evaluate the enciphering function by choosing assignments for both public and private variables. The values to be used are determined as an application of the following two theorems:

Theorem 1

For every polynomial \( p \) and for any subset of indices of variables \( I \), we define

\[ p_I := \sum_{x \in C_I} p|_x, \]

where \( C_I \) is the set of \( n \)-tuples such that the elements of index \( i \in I \) take all the possible combinations of values 0/1, while the ones with index \( i \notin I \) remain undetermined as a variable \( x_i \). So each element of \( C_I \) is a formal combination of boolean values and variables, and \( p_I \) is a polynomial which does not depend on variables with index in \( I \).

Then \( p_I = p_{S(I)} \) where \( p_{S(I)} \) is the quotient of the euclidean division of \( p \) by \( t_I := \prod_{i \in I} x_i \).

The quotient \( p_{S(I)} \) is called superpolynomial of the term \( t_I \). If for some index set \( I \), the corresponding polynomial \( p_{S(I)} \) is linear, then \( t_I \) is called maxterm of \( p \), and the following holds:

Theorem 2

Let \( t_I \) be a maxterm of a polynomial \( p \), so that its superpoly is \( p_{S(I)} = a_0 + a_1 x_1 + \ldots + a_n x_n \), and let \( X \) and \( X_j \) be the sets \( X = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I \} \) and \( X_j = \{ x \in \mathbb{F}_2^n : x_i = 0 \text{ for all } i \notin I \cup \{ j \} \text{ and } x_j = 1 \} \). Then

\begin{enumerate}
  \item \( a_0 = \sum_{x \in X} p|_x \)
  \item \( a_j = a_0 + \sum_{x \in X_j} p|_x \) for all \( j \notin I \)
\end{enumerate}

The two theorems above can be easily proven considering that in \( \mathbb{F}_2 \) the sum equals the difference and the fact that the characteristic of the field is 2. In the next sections we show how the theorems can be easily generalized to every finite field \( \mathbb{F}_q \).

In Section 2, we introduce the attack as presented in (Dinur & Shamir 2009). In Section 3, we describe the various phases of the attack when the polynomial representation of the enciphering function is available, while in Section 4 we describe a strategy to perform the attack in a realistic scenario, i.e., when we do not have the explicit expression of the polynomial but we have only access to it. Therefore, it can be accessed as a “black box” function. In Section 5, we present the original contribution of this paper by discussing the cube attack in \( \mathbb{F}_2 \) that was only claimed as possible in (Dinur & Shamir 2009).

2 Scenario of the attack

The basic requirement for the attack is that the cryptographic scheme can be expressed as a multivariate function in \( m + n \) variables over \( \mathbb{F}_2 \). Then we
may think this enciphering function as a polynomial
\[ p(v_1, \ldots, v_m, x_1, \ldots, x_n). \]
The crucial point in the
cube attack is that there are variables that can
be chosen, these variable are here denoted by \( v_1, \ldots, v_m \) and
they are called public variables. These variables
are for instance, known plaintext variables or they are
variables associated to bits of the initial vector. On
the other hand, there are variables that the attacker
cannot control, they are denoted by \( x_1, \ldots, x_n \) and
in typical cases they are the secret variables which
contain the key bits.

The aim of the attack is to “solve” the polynomial,
i.e., to find the values of the secret variables.

The attack can be divided in two distinct phases:
in the preprocessing phase the goal is to derive from
the polynomial \( p \) enough linear relations containing
only the secret variables to create a solvable linear
system of equations; at this scope the attacker can
evaluate the polynomial by suitably choosing both
public and secret variables. The matrix associated to
the found linear system can easily be inverted. Then,
in the online phase, secret variables can be attacked
by using the ciphertext to compute known terms of
the system: these are combinations of known cipher-
text values, these combinations are formed in accord
to the terms which appear in the sums of cubes on
the public variables used in the offline phase to find
the linear relations, Equation (1).

In this way the secret variables, which are set to
the key bits, can be determined by multiplication of
the inverse matrix with the vector of combinations of
ciphertext.

Note that the offline phase has to be carried on
just once, since the linear relations (i.e., the matrix)
are proper to the enciphering function.

3 The attack

In the first part of this section, up to Theorem 3,
we do not need to distinguish between public and secret
variables, so we consider a polynomial in \( n \) variables,
\[ p(x_1, \ldots, x_n), \]
for the sake of simplicity of notations.

Let \( p \) be a multivariate polynomial over \( \mathbb{F}_2 \).
Due to the field equation \( x^2 = x \) we know that each variable
\( x_i \) appears in the polynomial with exponent at most 1.
so we can identify each monomial with the subset
of the indices of the variables appearing in it.

We denote the term \( x_{i_1} \cdots x_{i_k} \) as \( t_{I} \), where \( I = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\} \).
For each term \( t_I \) we can factor the polynomial \( p \)
as \( p = t_I \cdot p_{S(I)} + q_I \), where each term in \( p_{S(I)} \) does
not have any of the variables with indices in \( I \) and
the polynomial \( q_I \) is the sum of those terms which are not
divisible for \( t_I \). We call \( p_{S(I)} \) the superpoly of \( I \) in \( p \).
We are interested in those terms which have a linear
non-constant superpoly.

Definition 1 A term \( t_I \) is a maxterm if \( \deg(p_{S(I)}) = 1 \).

Given a subset \( I \) of indices of size \( k \), say \( I = \{i_1, \ldots, i_k\} \), we define the cube \( C_I \) as
the set of \( n \)-tuples such that the elements of index
\( j \in I \) takes all the possible combinations of values \( 0/1 \),
while the ones with index \( j \not\in I \) remain undetermined as a
variable \( x_j \). So each element of \( C_I \) is a formal combina-
tion of boolean values and variables.1 For each el-
ment \( v \in C_I \) we denote by \( p|_v \) the polynomial
in \( n - k \) variables \( p(v) \), which does not depend on the
variables \( x_{i_1}, \ldots, x_{i_k} \).

The main result of this section is stated in the
following theorem.

Theorem 3 For every polynomial \( p \) and for any sub-
set of variables \( I \) we define \( p_I := \sum_{v \in C_I} p|_v \). Then
\[ p_I = p_{S(I)}. \]

Proof: We want to show the equivalence
\[ p_{S(I)} = \sum_{v \in C_I} p|_v = \sum_{v \in C_I} (t_I \cdot p_{S(I)} + q_I)|_v. \]

We know that the variables in \( I \) do not appear in the
superpoly \( p_{S(I)} \), so, since \( t_I \) is different from 0 (and
equals 1) only in the top vertex \( v^* \) of the cube \( C_I \)
such that \( v_j^* = 1 \) for all \( j \in I \), the superpoly \( p_{S(I)} \)
remains unevaluated and is added only once, i.e.,
\[ \sum_{v \in C_I} p|_v = \sum_{v \in C_I} (t_I \cdot p_{S(I)})|_v + \sum_{v \in C_I} q_I|_v = p_{S(I)} + \sum_{v \in C_I} q_I|_v. \]

Moreover, every term \( t_j \) of \( q_I \) misses at least one
of the variables in \( I \), so \( t_j \) does not change its value
when it is calculated on elements of the cube which
differ only in the variables not in \( J \). This means that
each different evaluation of the term \( t_j \) is added an
even number of times, and so vanishes in the sum. \( \square \)

Now we come back to the original model, in which
we distinguish the variables in public and secret ones,
\( p(v_1, \ldots, v_m, x_1, \ldots, x_n) \).

Since the goal of our attack is finding the values
of the \( n \) secret variables actually used to make the
encryption, and in the online phase we can tweak only
the public ones, we are interested in maxterms which
are product of only public variables, \( t_I = v_{i_1} \cdots v_{i_k}, \)
\( k \leq d - 1 \), while their superpolies are sums of only
secret variables. This last condition is easily satisfied
by setting all the public variables not in the maxterm
\( t_I \) to zero (or to any other chosen value).

To proceed with the attack, the first step is to find
enough (at least \( m \) maxterms, \( t_{I_1}, \ldots, t_{I_m} \), all con-
taining solely public variables. For each maxterm \( t_{I_i} \),
we calculate the superpoly \( p_{S(I_i)}(x_1, \ldots, x_n) \), which
can be evaluated in the key \( k = (k_1, \ldots, k_n) \) with
a chosen-plaintext attack, using Theorem 3, which states
that
\[ p_{S(I_i)}(k) = p_{I_i}(k) = \sum_{v \in C_{I_i}} p|_v(k). \]

For each superpoly \( p_{S(I_i)} \) we store the non con-
stant part \( p'_{S(I_i)}(x_1, \ldots, x_n) \) and the free term \( p_0' = p_{S(I_i)}(0, \ldots, 0) \), so that we can set up the linear system
\[ \begin{align*}
  p'_{S(I_1)}(x_1, \ldots, x_n) &= p_{S(I_1)}(k) - p'_0 = p'_{S(I_1)}(k) \\
  p'_{S(I_2)}(x_1, \ldots, x_n) &= p_{S(I_2)}(k) - p'_0 = p'_{S(I_2)}(k) \\
  &\vdots \\
  p'_{S(I_m)}(x_1, \ldots, x_n) &= p_{S(I_m)}(k) - p'_0 = p'_{S(I_m)}(k)
\end{align*} \]

which, if it has a unique solution, once solved gives
directly the key \( k \).

---

1 For instance: if \( n = 3 \) and \( I = \{1, 2\} \), then
\( C_I = \{(0, 0, x_3), (0, 1, x_3), (1, 0, x_3), (1, 1, x_3)\} \).
Note 1 Note that the search of the matrix of coefficients $A$ of system (2) (and its eventually inversion) can be done in a preprocessing phase, since it is independent of the key used in the encryption.

Example 1 Let us consider the polynomial

$$p(v_1, v_2, v_3, v_4, v, x_2) = v_1 v_2 v_3 + v_1 v_2 v_4 + v_2 v_3 v_4 + v_1 v_3 x_2 + v_1 v_2 x_1 + v_2 v_3 x_1 + v_2 v_4 x_2 + v_4 x_1 x_2 + v_2 v_1 + v_4 x_1 + v_1 + x_1 x_2 + 1$$

over $\mathbb{F}_2$. We want to recover the key $k = (0, 1)$.

In the preprocessing phase we look for maxterms which are product of only the public variables $v_1, \ldots, v_4$. Such maxterms are

$$t_{1,2} = v_1 v_2, \quad t_{2,3} = v_2 v_3,$$
$$t_{1,3} = v_1 v_3, \quad t_{1,4} = v_1 v_4,$$

while their superpolies, with the eventually other public variables set to zero, are

$$p_{S(1,2)} = x_2 + 1, \quad p_{S(2,3)} = x_1,$$
$$p_{S(1,3)} = x_1 + x_2 + 1, \quad p_{S(1,4)} = x_2.$$  

Among them, we choose two superpolies in order to create a square linear system with a unique solution. In this example we choose $p_{S(1,2)}$ and $p_{S(1,3)}$.

The preprocessing phase ends by setting out the system

$$\begin{align*}
x_2 &= p_{S(1,2)}(k) + 1 \\
x_1 + x_2 &= p_{S(1,3)}(k) + 1
\end{align*}$$

and calculating the inverse of the matrix of coefficients,

$$A^{-1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \in M_2(\mathbb{F}_2).$$

In the online phase we can evaluate the polynomial in some chosen values of only the public variables, and it is sufficient to find the free terms of the system (3), as

$$p_{S(1,2)}(k) = \sum_{v \in C_{1,2}} p|_{v}(k)$$

and

$$p_{S(1,3)}(k) = \sum_{v \in C_{1,3}} p|_{v}(k).$$

Finally, we are able to recover the key used by solving system (3) in polynomial time with the use of the matrix $A^{-1}$ previously calculated,

$$k = A^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = (0, 1).$$

Generic ciphers usually implement the Shannon idea of confusion/diffusion and they have very complicated and huge polynomial representations; as a consequence, any polynomial representing a cipher should be so chaotic that it is correct to suppose that its structure is indistinguishable from a random polynomial of a certain degree $d$, i.e., polynomial in which each monomial of degree at most $d$ can occur with probability $\frac{1}{2}$. We actually need only a weaker condition on the polynomial, since we are merely interested in maxterms which are products just of public variables.

Definition 2 A polynomial $p$ of $m$ public variables, $n$ secret variables and degree $d$ is a $d$-random polynomial if each term of degree $d$ which is the product of $d - 1$ public variables and one secret variable is independently chosen to occur with probability $\frac{1}{2}$.  

Note that in a $d$-random polynomial each term being product of $d - 1$ public variables is a maxterm with probability $1 - 2^{-n}$, as it is not a maxterm only if all the terms which contain the same $d - 1$ public variables and any secret variables do not appear in $p$.

Thus, with the hypothesis above, written the system (2) as $Ax = b$, we can suppose that every entry in the binary matrix $A$ is chosen randomly. So, in order to estimate the probability that $A$ is invertible, we use the following lemma:

Lemma 1 The probability that a random matrix $A \in M_n(\mathbb{F}_2)$ is invertible is $\prod_{i=1}^{n} \left(1 - \frac{1}{2^n}\right)$.

Proof: We recall that a square matrix $A$ of order $n$ is invertible if and only if its rank is maximum and equals to $n$, i.e., if all its rows are linearly independent. This means that, for $i = 1, \ldots, n$, the $i$th row must be linearly independent from all the previous ones, so that it can be chosen in $2^n - 2^{i-1}$ different ways out of the $2^n$ possible $n$-tuples of elements in $\mathbb{F}_2$. Thus, the probability that $A$ is invertible is

$$\prod_{i=1}^{n} \left( \frac{2^n - 2^{i-1}}{2^n} \right) = \prod_{i=1}^{n} \left(1 - \frac{1}{2^{i+1}}\right) = \prod_{i=1}^{n} \left(1 - \frac{1}{2^n}\right).$$

\hfill $\square$

Remark 1 It is easy to show that the sequence $S_n = \prod_{i=1}^{n} (1 - \frac{1}{2^n})$ is decreasing and converge to a positive value, as stated by the equivalence

$$\prod_{i=1}^{\infty} (1 - \theta_i) > 0 \iff \sum_{i=1}^{\infty} \theta_i < \infty ,$$

where $\theta_i \in [0, 1]$. As you can see from Figure 1, this value is approximated by 0.28879.

Figure 1: Probability $\Pr[\det(A) \neq 0]$ expressed as a function of the order of the matrix $A$.  

This means that the probability that system (2) has a unique solution can be made arbitrary close to 1 increasing the number of maxterms taken into account.
4 The “black box” attack

If the polynomial expression is not available, for instance because we have to deal with a huge polynomial or because the internal structure of the cipher has been kept secret, we have to find the required maxterms in a more complex way. In this case, in fact, we have to proceed with a random walk tweaking both public and secret variables, which can always be done in the preprocessing phase. This is possible because of the following theorem, in which we do not distinguish between public and secret variables.

Theorem 4 Let \( t_I \) be a maxterm in a polynomial \( p(x_1, \ldots, x_n) \), so that its superpoly is \( p_{S(I)}(x_1, \ldots, x_n) = a_0 + a_1 x_1 + \ldots + a_n x_n \), and let \( X \) and \( X_j \) be the sets \( X = \{ x \in \mathbb{F}_2^n: x_i = 0 \text{ for all } i \notin I \} \) and \( X_j = \{ x \in \mathbb{F}_2^n: x_i = 0 \text{ for all } i \notin I \cup \{ j \} \} \) and \( x_j = 1 \}. \) Then

1. \( a_0 = \sum x p(x) \)
2. \( a_j = a_0 + \sum x_j p(x) \) for all \( j \notin I \).

Proof: We recall that, given the maxterm \( t_I \), the polynomial \( p \) can be written as \( p = t_I p_{S(I)} + q_I \).

\[
\sum x p(x) = \sum x [t_I(x) p_{S(I)}(x) + q_I(x)] = \sum x t_I(x) p_{S(I)}(x) + \sum x q_I(x) = p_{S(I)}(0, \ldots, 0) = a_0
\]

since the second sum is zero modulo 2 because, as already mentioned, each term in \( q_I \) is a monomial that lacks at least a variable with index in \( I \). As a consequence, the values obtained when the monomial is evaluated on \( C_I \) are summed an even number of times. On the other hand, in the first sum, the result is that only in one case the terms \( t_I(x) p_{S(I)}(x) \) take a value different from 0, namely when \( t_I \) is evaluated to 1: in this case, since the superpoly does not contain any variable with index in \( I \), it is evaluated only with its \( n-k \) arguments all equal to zero.

\[
\sum x_j p(x) = \sum x_j [t_I(x) p_{S(I)}(x) + q_I(x)] = \sum x_j t_I(x) p_{S(I)}(x) + \sum x_j q_I(x) = a_0 + a_j
\]

since the second sum is zero and in the first one the only term summed is the one for which \( t_I = 1 \), but in this case the top vertex element \( v^* \) has a 1 also in the \( j \)th position, so, with the free term of the superpoly \( p_{S(I)} \) also the coefficient of the term \( x_j \) summed.

In Algorithm 4.1, we report the strategy presented in (Dinur & Shamir 2009), which can be applied when the polynomial representation of the cryptosystem can not be assumed to be a \( d \)-random polynomial.

For what concern the testing of the linearity of the superpoly, Dinur and Shamir suggest to use a probabilistic linearity test, as for instance the BLR test, which consists in choosing independently and randomly vectors \( a, b \in \mathbb{F}_2^n \) and verifying the condition

\[
p_{S(I)}(a) + p_{S(I)}(b) + p_{S(I)}(0, \ldots, 0) = p_{S(I)}(a + b).
\]

5 Cube attack in \( \mathbb{F}_q \)

We consider a polynomial \( p \in \mathbb{F}_q[x_1, \ldots, x_n] \) of degree \( d = \deg(p) \) (without distinguishing between public and secret variables) and a monomial \( t = x_{i_1}x_{i_2}x_{i_3} \), where \( 0 \leq r_i < q \) for \( 1 \leq i \leq k \). For any monomial \( t \), we can factor \( p \) as before

\[
p = t \cdot p_{S(t)} + q_t
\]

where \( q_t \in \mathbb{F}_q[x_1, \ldots, x_n] \) is the sub-polynomial of \( p \) which is the sum of all the terms of \( p \) which are not divisible by \( t \). Note that, differently from the case \( \mathbb{F}_2 \), the superpoly \( p_{S(t)} \) can actually contain some of the variables of \( t \), while both \( p_{S(t)} \) and \( q_t \) can even have terms containing all the variables \( x_{i_1}, \ldots, x_{i_k} \); we denote this sub-polynomial of \( q_t \) as \( q'_t \).

Example 2 Let us consider the monomial \( t = x_1^3x_2^3 \) and the polynomial \( p \in \mathbb{F}_8[x_1, \ldots, x_3]:

\[
p = x_1^3x_2^3x_3^3 + x_1^4x_2^3x_4 + x_1^2x_2^3 + x_2^2x_3^3 + x_1x_2^3x_4 + x_1^2x_2x_3 + x_1^4x_2^3x_3^3 + x_2^3 + x_2^3 + 1 = x_1^3x_2^3x_3^3 + x_2x_4x_3 + x_3 + x_4 + x_2^3 + x_3^3 + x_1^4x_2x_3 + x_1^2x_2^3 + x_3^2 + x_2^3 + 1.
\]

In this case

\[
p_{S(t)} = x_1^3x_2^3 + x_2x_4x_3 + x_3 + x_4,
\]

\[
q_t = x_1^4x_2 + x_2x_3 + 1
\]

and

\[
q'_t = x_1^4x_2 + x_1x_2x_3.
\]

We have a similar result as in (Dinur & Shamir 2009), by considering the \( k \)-dimensional Boolean cube

\[
C_t = \{ (x_1, \ldots, x_n) : x_i \in \{0,1\} \subset \mathbb{F}_q \}
\]

for each variable appearing in \( t \).

For each element \( v \in \mathbb{C}_t \) we denote by \( p_{S(t)} \) the polynomial in \( n-k \) variables \( p(v) \). Moreover, we define the top vertex \( v^* \) of the cube \( C_t \) as

\[
v^*_t := \begin{cases} x_i & \text{if } x_i \text{ does not appear in } t \\ 1 & \text{if } x_i \text{ appears in } t \end{cases}
\]
Let us denote by $W_H(v)$ the number of ones in $v$, i.e., it would be like the Hamming weight if we were in $F_2^n$, but here we have vectors with elements which are 1, 0 or variables. Note that the weight of the top vertex is $k$ by definition $W_H(v^*) = k$.

Analogously to Theorem 3 we have the following result:

**Theorem 5** Let us define

$$p_t := \sum \limits_{v \in C_t} (-1)^w(v)p_{\text{v}}$$

with $w(v) := W_H(v^*) + W_H(v) = k + W_H(v)$.

Then

$$p_t = (p_{S(t)} + q_t)|v^*.$$  

**Proof:** In a way similar to the proof of Theorem 3, we obtain

$$\sum \limits_{v \in C_t} (-1)^w(v)p_{\text{v}} = \sum \limits_{v \in C_t} (-1)^w(v) \left[ t \cdot p_{S(t)} + q_t \right]_{v}$$

For all $v \in C_t$ except $v^*$, $t|v = 0$, so, the contribution of the first sum in $p_t$ corresponds exactly to the evaluation of $p_{S(t)}$ on $v^*$,

$$\sum \limits_{v \in C_t} (-1)^w(v) \left[ t \cdot p_{S(t)} \right]_{v} = p_{S(t)}(v^*).$$

For what concerns the second sum, any term appearing in $q_t$ contains all variables in $t$, so it is summed only once (in the top-vertex $v^*$). All the other terms of $q_t$ lack at least one of the variables in $t$, so they contribute an even number of times to the sum, half times with a positive sign and the other half with a negative one, thus globally they vanish and therefore

$$\sum \limits_{v \in C_t} (-1)^w(v)q_t|_v = q_t^*(v^*).$$

\[\square\]

**Note 2** Note that $w(v)$ has been defined with the “correcting value” $k$ so that the element of the sum which corresponds to the top-vertex $v^*$ has always a positive sign. This correction is also present in the definitions of $w$ in Theorem 6.

**Example 3** Let us continue with Example 2; we already have the superpoly

$$p_{S(t)}(x_1, \ldots, x_5) = x_1^5 + x_2x_4 + x_2 + x_3 + x_4,$$

and

$$q_t^*(x_1, \ldots, x_5) = x_2^2x_2 + x_1x_2x_3.$$  

Then we consider the evaluation of $p$ on the cube

$$C_t = \{ (0, 0, x_3, x_4, x_5), (0, 1, x_3, x_4, x_5), (1, 0, x_3, x_4, x_5), (1, 1, x_3, x_4, x_5) \}$$

in order to compute $p_t$. Thus

$$p_t = p(0, 0, x_3, x_4, x_5) - p(0, 1, x_3, x_4, x_5) + \ldots + p(1, 1, x_3, x_4, x_5)$$

Thus

$$p_t = 1 (x_3^5 + 1) - (x_3^5 + x_3^2 + x_3 + 1) + \ldots + (x_3^5 + x_3 + 1) = \ldots$$

$$(x_3^5 + x_3 + 1) = p_{S(t)}(v^*) + q_t^*(v^*).$$

Hereby, we consider the problem of determining all the coefficients of the sub-polynomial $p_{S(t)} + q_t^*$ when it is linear and $p$ is given as a black box function. With respect to the case in $F_2^n$, the main difference is that whatever the monomial is, the polynomial we obtain evaluating $p$ as in Theorem 4 is always the linear part of $p_{S(t)}$, where $t_0$ is the monomial which is product of all the variables in $t$ taken with exponent 1, $t_0 = x_{i_1} \cdots x_{i_k}$. This is due to the fact that, called $I = \{i_1, \ldots, i_k\}$ the set of the indices of the variable appearing in $t$, evaluating $p$ on the sets $X = \{x \in F_2^n : x_i = 0 \text{ if } i \not\in I \text{ and } x_i \in \{0, 1\} \text{ if } i \in I\}$ and $X_j = \{x \in F_2^n : x_i = 0 \text{ if } i \not\in I \cup \{j\}, x_i \in \{0, 1\} \text{ if } i \in I \text{ and } x_j = 1\}$ we can only obtain the coefficients of the linear part of $(p_{S(t)} + q_t^*)|_{v^*}$ (which is a polynomial in $n-k$ variables, all but $x_{i_1}, \ldots, x_{i_k}$), i.e., this sum can distinguish if a variable is present in the monomial but cannot determine its degree.

We denote with $x^*$ (and respectively $x^{*j}$) the element of $X$ (respectively of $X_j$) such that $x^*_i = 1$ (respectively $x^{*j}_i = 1$) for all $i \in I$. Then, we have:

**Theorem 6** Let $t = x_{i_1}^* \cdots x_{i_k}^*$ be a monomial, and let $I$ be the subset of indices which appear in $t$ as denoted above. If the polynomial $(p_{S(t)} + q_t^*)|_{v^*}$ is linear, $(p_{S(t)} + q_t^*)|_{v^*} = a_0 + a_1 x_1 + \ldots + a_n x_n$, then

1. $a_0 = \sum \limits_X (-1)^w(x)p(x)$

   where $w(x) = W_H(x^*) + W_H(x)$;

2. $a_j = -a_0 + \sum \limits_{X_j} (-1)^w(x)p(x)$

   where $w(x) = W_H(x^*) + W_H(x)$ and $j \not\in I$.

Besides, denoting with $t_0$ the monomial $x_{i_1} \cdots x_{i_k}$, then

$$p_{S(t_0)}|_{v^*} = (p_{S(t_0)} + q_t^*)|_{v^*}.$$  

**Proof:**

1. $\sum \limits_X (-1)^w(x)p(x) = \sum \limits_X (-1)^w(x) [t(x)p_{S(t_0)}(x) + q_t(x)] = \sum \limits_X (-1)^w(x)t(x)p_{S(t_0)}(x) + \sum \limits_X (-1)^w(x)q_t(x)$

   in the first sum the only (possibly) nonzero term is the one corresponding to $x^* \in X$, which is summed with a positive sign thus we have

$$\sum \limits_X (-1)^w(x)t(x)p_{S(t_0)}(x) = p_{S(t_0)}(x^*).$$
in the second sum all the terms containing other variables than \( x_i, \ldots, x_k \) are obviously zero, and all the other terms are summed an even number of times (and vanish since they take opposite signs in pairs), except the ones which contains all (and only) variables \( x_i, \ldots, x_k \), which are summed only once (in \( x^* \)) with a positive sign:

\[
\sum_X (-1)^{w(x)} q(x) = q^*_t(x^*).
\]

Therefore we obtain the claimed result:

\[
\sum_X (-1)^{w(x)} p(x) = p_{S(t)}(x^*) + q^*_t(x^*).
\]

2.

\[
\sum_{X_j} (-1)^{w(x)} t(x) p_{S(t)}(x) = \sum_{X_j} (-1)^{w(x)} t(x) [p_{S(t)}(x) + q_t(x)] = \sum_{X_j} (-1)^{w(x)} t(x) p_{S(t)}(x) + \sum_{X_j} (-1)^{w(x)} q_t(x) = \]

\[
p_{S(t)}(x^{*j}) + q^*_t(x^{*j})
\]

similarly to the previous case, only coefficients of the terms which contain all the variables in \( t \) and possibly \( x_j \) are summed.

Besides, to show the last equality, it is sufficient to notice that

- \( t \cdot p_{S(t)} + q^*_t = t_0 \cdot p_{S(t_0)} \)
- the sums above return only the free terms and the coefficients of the linear terms \( x_j, j \notin I \), once that all the variables \( x_i, \ldots, x_k \) are set to 1.

\[\Box\]

Example 4 Let us consider the polynomial \( p \in \mathbb{F}_3[x_1, x_2, x_3, x_4] \), then we have

\[
p = x_1^2 x_2 x_3 + 3x_1^2 x_2 x_4 + x_1^3 x_2 + x_1 x_2^2 + 2x_2 x_3 + x_3^2 x_4 + x_3 + 1
\]

and the monomial \( t = x_1^3 x_2^2 \), so that

\[
p_{S(t)} = x_3 + 2x_4 + x_1 \quad \text{and} \quad q^*_t = x_1 x_2^2.
\]

With the sums above we obtain \( a_0 = 2, a_2 = 1 \) and \( a_4 = 2 \), which are the free term and the coefficients of the polynomials

\[
(p_{S(t)} + q^*_t)\{x^* = p_{S(t_0)}\} = x_3 + 2x_4 + 2 \in \mathbb{F}_3[x_3, x_4].
\]

We conclude by stating the following lemma, which can be used to estimate the probability of success of the attack.

Lemma 2 Given a random matrix \( A \in M_n(\mathbb{F}_q) \), then the probability that \( A \) is invertible is

\[
\prod_{i=1}^n \left( 1 - \frac{1}{q^i} \right).
\]

Note that the proof of this lemma is analogous to the one of Lemma 1, and that the conclusion of Remark 1 remains valid (see Figure 2).

6 Conclusions

The cube attack has similarities with the AIDA\(^2\) (Vielhaber 2007), published a year before the one by Dinur and Shamir, but which was presented as a technique directed to the analysis of the Trivium cipher. In this paper, we presented the necessary modifications of the cube attack in order to be applied to a system working on a generic finite field \( \mathbb{F}_q \). The applicability of this version of the cube attack has to be further investigated from the point of view of implementation and it will probably share difficulties with the implementation of the original version on \( \mathbb{F}_2 \). A technique similar to the efficient brute force method recently presented in (Bouillaguet et al. 2010), once applied on cubes seems to provide better results. This technique has to be considered in the higher order setting we showed in this work.

References


URL: http://dx.doi.org/10.1201/9781439070033


\(^2\)Algebraic IV Differential Attack.