# Massive Superpoly Recovery with Nested Monomial Predictions

Kai Hu, Siwei Sun, Yosuke Todo, Meiqin Wang, Qingju Wang

ASIACRYPT 2021

December 7, 2021

# Introduction

- At EUROCRYPT 2009 Dinur and Shamir proposed cube attack
- At CRYPT 2017 the conventional bit-based division property was first introduced to probe the structure of the superpoly
- At EUROCRYPT 2020 Hao et al. proposed the three-subset division property without unknown subsets

## Contribution

- Propose a new framework with nested monomial predictions which scales well for massive superpoly recovery.
- With help of the Möbius transformation, we present a novel key-recovery technique based on superpolies involving all key bits exploiting the disjoint properties

# Preliminaries

## Boolean Function

Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ be a Boolean function whose algebraic normal form (ANF) is

$$f(\boldsymbol{x}) = f(x_0, x_1, ..., x_{n-1}) = \bigoplus_{\boldsymbol{u} \in \mathbb{F}_2^n} a_{\boldsymbol{u}} \prod_{i=0}^{n-1} x_i^{u_i}$$

where $a_{\boldsymbol{u} \in \mathbb{F}_2}$ and

$$\boldsymbol{x}^{\boldsymbol{u}} = \pi_{\boldsymbol{u}}(\boldsymbol{x}) = \prod_{i=0}^{n-1} x_i^{u_i} = \begin{cases} x_i, if & u_i = 1, \\ 1, if & u_i = 0, \end{cases}$$

Example 1 Let $f(x_0, x_1) = x_0 x_1 \oplus x_0 \oplus 1$ ,then we have $x_0 x_1 \rightarrow f, x_0 \rightarrow f, 1 \rightarrow f, x_1 \not\rightarrow f$

## Vectorial Boolean Function

$\boldsymbol{f} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ be a vectorial Boolean function with $\boldsymbol{y} = (y_0, y_1, ..., y_{m-1}) = \boldsymbol{f}(\boldsymbol{x}) = (f_0(\boldsymbol{x}), ..., f_{n-1}(\boldsymbol{x}))$ . For $\boldsymbol{x} \in \mathbb{F}_2^n$,we use $\boldsymbol{y}^{\boldsymbol{v}}$ to denote the product of some coordinates of $\boldsymbol{y}$:

$$\boldsymbol{y}^v = \prod_{i=0}^{m-1} y_i^{v_i} = \prod_{i=0}^{m-1} (f_i(\boldsymbol{x}))^{v_i}$$

# Monomial Prediction

Let $\boldsymbol{f} : \mathbb{F}_2^{n_0} \to \mathbb{F}_2^{n_r}$ be a composite vectorial Boolean function of a sequence of r smaller function $\boldsymbol{f}^i : \mathbb{F}_2^{n_i} \to \mathbb{F}_2^{n_{i+1}}, 0 \leq i \leq r-1$ as

$$\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)} \tag{1}$$

**Definition 1 (Monomial Trail)**

Let $\boldsymbol{x}^{(i+1)} = \boldsymbol{f}^{(i)}(\boldsymbol{x}^{(i)})$ for $0 \leq i < r$.We call a sequence of monomails $(\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}), \pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)}), \ldots, \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}))$ an r-round monomial trail connecting $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ with respect to the composite function $\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)}$ if

$$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(1)}}(\boldsymbol{x}^{(1)}) \to \cdots \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$$

If there is at least one monomial trail connecting $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$,we write $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. Otherwise,$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$

## Example 2

Let $\boldsymbol{z} = (z_0, z_1) = \boldsymbol{f}^{(0)}(y_0, y_1) = (y_0y_1, y_0 \oplus y_1), \boldsymbol{y} = (y_0, y_1) = \boldsymbol{f}^{(0)}(x_0, x_1, x_2) = (x_0 \oplus x_1 \oplus x_2, x_0x_1 \oplus x_0 \oplus x_2)$ and $\boldsymbol{f} = \boldsymbol{f}^{(0)}\boldsymbol{f}^{(1)}$.

$\boldsymbol{y}$

$$(y_0, y_1)^{(0,0)} = 1, (y_0, y_1)^{(1,0)} = y_0 = \underline{x_0} \oplus x_1 \oplus x_2, (y_0, y_1)^{(0,1)} = y_1 = x_0x_1 \oplus \underline{x_0}$$

$\oplus x_2$,

$(y_0, y_1)^{(1,1)} = y_0y_1 = x_0x_1x_2 \oplus x_0x_1 \oplus x_1x_2 \oplus \underline{x_0} \oplus x_2$.

Then

$$x_0 \rightarrow y_0, x_0 \rightarrow y_1, x_0 \rightarrow y_0y_1$$

Similarly

$$(z_0, z_1)^{(0,0)} = 1, (z_0, z_1)^{(1,0)} = z_0 = y_0y_1, (z_0, z_1)^{(0,1)} = z_1 = y_0 \oplus y_1,$$

$(z_0, z_1)^{(1,1)} = z_0z_1 = 0$

Then connecting $x_0$ and monomials of z:

$$x_0 \rightarrow y_0 \rightarrow z_1, x_0 \rightarrow y_1 \rightarrow z_1, x_0 \rightarrow y_0y_1 \rightarrow z_0$$

## Lemma 1

$\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$. if $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$,and thus $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\rightsquigarrow \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ implies $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \not\to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$

Considering Example 2, although $x_0 \rightsquigarrow z_1$,we have $x_0 \not\to z_1$ since

$$z_1 = y_0 \oplus y_1 = \underline{x_0} \oplus x_1 \oplus x_2 \oplus x_0x_1 \oplus \underline{x_0} \oplus x_2 = x_0x_1 \oplus x_1.$$

## Definition 2 (Monomial Hull)

For $\boldsymbol{f}$ with a specific composition sequence, the monomial hull of $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)})$ and $\pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$, denoted by $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$,is the set of all monomial trails connecting them. The number of trails in the monomial hull is called the **size** of the hull and is denoted by $|\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})|$.

## Example 3

Consider Example 2, the monomial hull of $x_0$ and $z_1$ is the set

$$x_0 \bowtie z_1 = \{x_0 \to y_0 \to z_1, x_0 \to y_1 \to z_1\}$$

Thus the size of $x_0 \bowtie z_1$ is 2.Furthermore, since $x_0 \not\rightsquigarrow z_0 z_1$, $x_0 \bowtie z_0 z_1 = \emptyset$ and $|x_0 \bowtie z_0 z_1| = 0$.

## Theorem 1

Let $\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)}$ defined as above. $\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \to \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)})$ if and only if

$$|\pi_{\boldsymbol{u}^{(0)}}(\boldsymbol{x}^{(0)}) \bowtie \pi_{\boldsymbol{u}^{(r)}}(\boldsymbol{x}^{(r)}) \equiv 1 \pmod 2.$$

# Cube Attack

For a cipher with a secret key $k \in \mathbb{F}_2^m$ and a public input $x \in \mathbb{F}_2^n$, a Boolean function $f(x, k)$

$$f(\boldsymbol{x}, \boldsymbol{k}) = p(\boldsymbol{x}[\hat{\boldsymbol{u}}], \boldsymbol{k}) \cdot \boldsymbol{x}^{\boldsymbol{u}} + q(\boldsymbol{x}, \boldsymbol{k})$$

Let $\mathbb{C} = \boldsymbol{x} \in \mathbb{F}_2^n : \boldsymbol{x} \preceq \boldsymbol{u}$

$$\bigoplus_{\boldsymbol{x} \in \mathbb{C}_u} f(\boldsymbol{x}, \boldsymbol{k}) = \bigoplus_{\boldsymbol{x} \in \mathbb{C}_u} (p \cdot \boldsymbol{x}^{\boldsymbol{u}} + q(\boldsymbol{x}, \boldsymbol{k})) = p$$

It is easy to check that the superpoly of $\mathbb{C}_u$ is just the coeffcient of $\boldsymbol{x}^{\boldsymbol{u}}$ in the parameterized Boolean function $f(\boldsymbol{x}, \boldsymbol{k})$

$$p(\boldsymbol{x}[\hat{\boldsymbol{u}}], \boldsymbol{k}) = Coe(f(\boldsymbol{x}, \boldsymbol{k}), \boldsymbol{x}^{\boldsymbol{u}}).$$

# The Nested Framework

Given a parameterized Boolean function which consists of a sequence of simple vectorial Boolean functions as

$$\boldsymbol{f} = \boldsymbol{f}^{(r-1)} \circ \boldsymbol{f}^{(r-2)} \circ \cdots \circ \boldsymbol{f}^{(0)}$$

$$f(\boldsymbol{x}, \boldsymbol{k}) = \bigoplus_{\boldsymbol{t} \in \mathbb{F}_2^n, \pi_{\boldsymbol{t}}(r-r_0)(\boldsymbol{s}^{(r-r_0)}) \in \mathbb{S}^{(r-r_0)}} \pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)})$$

where $\mathbb{S}^{(r-r_0)} = \{\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}) : \pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}) \rightarrow f\}$

**Compute Coe($\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}), \boldsymbol{x}^{\boldsymbol{u}}$)**

$\boldsymbol{s}^{(r-r_0)}$ is the output vector of a new composite vectorial Boolean function as

$$\boldsymbol{s}^{(r-r_0)} = \boldsymbol{f}^{(r-r_0-1)} \circ \boldsymbol{f}^{(r-r_0-2)} \ldots \boldsymbol{f}^{(0)}$$

then $\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)})$ is a polynomial of $(\boldsymbol{x}, \boldsymbol{k})$

Hence we can construct the MILP model to enumerate all feasible trails representing
$\boldsymbol{k}^{\boldsymbol{v}}\boldsymbol{x}^{\boldsymbol{u}} \rightsquigarrow \pi_{\boldsymbol{t}(r-r_0)}(\boldsymbol{s}^{(r-r_0)})$
set a time limit $\tau^{(r-r_0)}$ for thr MILP model.We use

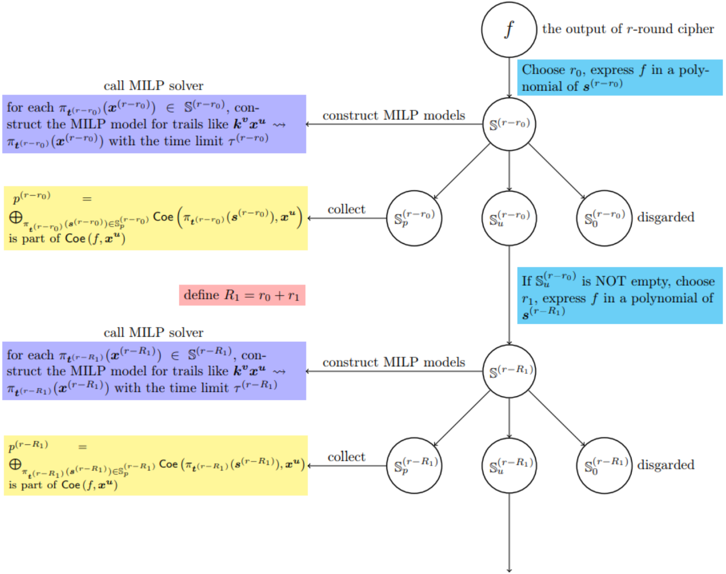
$$\mathcal{M}.TimeLimit \leftarrow \tau^{(r-r_0)}$$

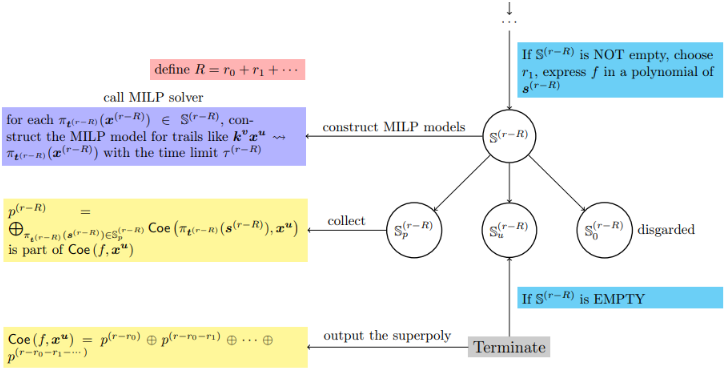
For each element in $\mathbb{S}^{(r-r_0)}$,the model of enumerating the trails will end up with three different kinds of status

1. The model is solved and infeasible, then Coe$(\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}))$
2. The model is solved and feasible, and all the solutions has been enumerated,then Coe$(\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}))$ are obtained
3. The model is not solved in the time limit $\tau^{(r-r_0)}$

$$\mathbb{S}^{(r-r_0)} = \mathbb{S}_0^{(r-r_0)} \bigcup \mathbb{S}_0^{(r-r_0)}{}_p \bigcup \mathbb{S}_0^{(r-r_0)}{}_u$$

- $\mathbb{S}_0^{(r-r_0)}{}_0$ is called a solved-0 set ,case 1
- $\mathbb{S}_0^{(r-r_0)}{}_p$ is called a solved-p set ,case 2
- $\mathbb{S}_0^{(r-r_0)}{}_u$ is called a undecided set ,case 3


$f$
the output of $r$-round cipher
Choose $r_0$, express $f$ in a polynomial of $\boldsymbol{s}^{(r-r_0)}$
call MILP solver
for each $\pi_{\boldsymbol{t}^{(r-r_0)}}(\boldsymbol{x}^{(r-r_0)}) \in \mathbb{S}^{(r-r_0)}$, construct the MILP model for trails like $\boldsymbol{k}^{\boldsymbol{v}}\boldsymbol{x}^{\boldsymbol{u}} \rightsquigarrow \pi_{\boldsymbol{t}^{(r-r_0)}}(\boldsymbol{x}^{(r-r_0)})$ with the time limit $\tau^{(r-r_0)}$
construct MILP models
$\mathbb{S}^{(r-r_0)}$
$p^{(r-r_0)} = \bigoplus_{\pi_{\boldsymbol{t}^{(r-r_0)}}(\boldsymbol{s}^{(r-r_0)}) \in \mathbb{S}_p^{(r-r_0)}} \mathsf{Coe}\left(\pi_{\boldsymbol{t}^{(r-r_0)}}(\boldsymbol{s}^{(r-r_0)}), \boldsymbol{x}^{\boldsymbol{u}}\right)$ is part of $\mathsf{Coe}(f, \boldsymbol{x}^{\boldsymbol{u}})$
collect
$\mathbb{S}_p^{(r-r_0)}$
$\mathbb{S}_u^{(r-r_0)}$
$\mathbb{S}_0^{(r-r_0)}$
disgarded
If $\mathbb{S}_u^{(r-r_0)}$ is NOT empty, choose $r_1$, express $f$ in a polynomial of $\boldsymbol{s}^{(r-R_1)}$
define $R_1 = r_0 + r_1$
call MILP solver
for each $\pi_{\boldsymbol{t}^{(r-R_1)}}(\boldsymbol{x}^{(r-R_1)}) \in \mathbb{S}^{(r-R_1)}$, construct the MILP model for trails like $\boldsymbol{k}^{\boldsymbol{v}}\boldsymbol{x}^{\boldsymbol{u}} \rightsquigarrow \pi_{\boldsymbol{t}^{(r-R_1)}}(\boldsymbol{x}^{(r-R_1)})$ with the time limit $\tau^{(r-R_1)}$
construct MILP models
$\mathbb{S}^{(r-R_1)}$
$p^{(r-R_1)} = \bigoplus_{\pi_{\boldsymbol{t}^{(r-R_1)}}(\boldsymbol{s}^{(r-R_1)}) \in \mathbb{S}_p^{(r-R_1)}} \mathsf{Coe}\left(\pi_{\boldsymbol{t}^{(r-R_1)}}(\boldsymbol{s}^{(r-R_1)}), \boldsymbol{x}^{\boldsymbol{u}}\right)$ is part of $\mathsf{Coe}(f, \boldsymbol{x}^{\boldsymbol{u}})$
collect
$\mathbb{S}_p^{(r-R_1)}$
$\mathbb{S}_u^{(r-R_1)}$
$\mathbb{S}_0^{(r-R_1)}$
disgarded




The solved-0 set is discarded naturally since the elements in it have no contribution to $\mathsf{Coe}(f, \boldsymbol{x}^{\boldsymbol{u}})$.

For the solved-p set ,

$$p^{(r-r_0)} = \bigoplus_{\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)})\in\mathbb{S}_p^{(r-r_0)}} Coe(\pi_{\boldsymbol{t}}^{(r-r_0)}(\boldsymbol{s}^{(r-r_0)}))$$

# Key-Recovery Attacks Exploiting Massive Superpolies

## offline

We have recovered the exact ANF of superpoly $p(\boldsymbol{f})$ for the cube term $\boldsymbol{x}^{\boldsymbol{u}}$ (the corresponding cube is denoted by $\mathbb{C}_{\boldsymbol{u}}$).

## online

In the online phase, we first call the cipher oracle to encrypt all elements in the cube and get the value of the superpoly with time complexity $2^{wt(\boldsymbol{u})}$.

Next,we try to obtain some information of the secret key from the equation:

$$p(\boldsymbol{k}) = \bigoplus_{\boldsymbol{x} \in \mathbb{C}_{\boldsymbol{u}}} f_{\boldsymbol{k}}(\boldsymbol{x}).$$

It is well known that Möbius transformation is available for the conversion between the ANF and the truth table of any Boolean function.

**Algorithm 1** Möbius transformation

```
procedure MöbiusTransformation((a[i], 0 ≤ i ≤ 2^n) :)
    for k = 1 to n do
        for i = 0 to 2^(n−k) do
            for j = 0 to 2^(k−1) − 1 do
                a[2^k i + 2^(k−1) + j] = a[2^k i + j] ⊕ a[2^k i + 2^(k−1) + j]
    return a
```

It requires $n \times 2^{n-1}$ 1-bit XORs and $2^n - bit$ memory complexity.
Considering the sparse property,the efficient algorithm, the Möbius transformation costs only $n \times 2^{n-2}$ XORs for the superpolies we consider in this paper

# Divide-and-Conquer Method Using the Disjoint Set

**Definition 1 (Disjoint set)**

Given a superpoly $p(\boldsymbol{k})$ with n variables, if for $0 \leq i \neq j < n$,$k_i$ and $k_j$ are never multiplied mutually in all monomials of $p(\boldsymbol{k})$, then we say $k_i$ and $k_j$ are disjoint. If for a subset of variables $D \subseteq \{k_0, k_1, ..., k_{n-1}\}$, every pair of variables like $k_i, k_j \in D$ are all disjoint,we call D a disjoint set.

Search for a disjoint set of $p(\boldsymbol{k})$ A matrix $M \in \mathbb{F}_2^n$ is called the disjoint matrix of $p(\boldsymbol{k})$,if $M[i][j] = 0$ when $k_i, k_j$ are disjoint ,$M[i][j] = 1$ otherwise.

1. sort the variables in $\{k_0, k_1, ..., k_{n-1}\}$ in certain order,an increasing order according to the value $\sum_{0 \leq j < n} M[i][j]$ for $k_i$.The sorted variables are denoted as$\{k'_0, k'_1, ..., k'_{n-1}\}$;
2. initialize a set $D = \{k'_0\}$
3. for $1 \leq i < n$,if $k'_i$ is disjoint with all variables in D,put $k'_i$ into D; otherwise,process the next variable
4. after all the variables are processed,D is one of the disjoint sets.

## Key recovery attacks with single balanced superpoly

If the balanced superpoly $p(\boldsymbol{k})$ has a disjoint set $D$ with m variables and $J = \{k_0, k_1, ..., k_{n-1}\}/D$, then $p(\boldsymbol{k})$ can be written as the form

$$p(k_0, k_1, ..., k_{n-1}) = \left( \bigoplus_{0 \le i < m} k_i \cdot p_i(J) \right) \oplus p_m(J)$$

**Complexity**

Every $p_i(J)$ involves at most n-m variables, Möbius transform to compute the truth tables of $p_0, p_1, ..., p_m$ over all possible values of variables in J

For the linear equation, we can remove 1-bit key guessing efficiently after guessing m $-$ 1 key bits additionally

$(m+1) \times (n-m) \times 2^{n-m-2}$ XORs to construct the truth tables

$2^{(m-1)}$ guesses for the values of any m $-1$ variables in the linear equations

## Key recovery attacks with multiple balanced superpolies

Suppose we have recovered N balanced superpolies $p^{(0)}, p^{(1)}, ..., p^{(N-1)}$,we call $D$ their common disjoint set
The complexity of the case then consists of

1. constructing the truth tables, which costs $N \times (m+1) \times (n-m) \times 2^{n-m-2}$ XORs;
2. constructing the linear equations, which is $N \times 2^{n-m} \times (m+1)$ truth table lookuos;
3. guessing the value of $m - N$(we always let $m > N$) cariables,then the remaining N variables can be determined by solving a set of simple linear equations.This step costs $2^{n-m} \times x^{m-N}$ guesses.