Input: $\mathcal{P}$: $\{p_j(\mathbf{x})\}_{i=0}^{m-1}$, m: Integer, n: Integer, n_1: Integer
Result: A solution to the system $\mathcal{P}$

PREPROCESS($\mathcal{P}$)
$\ell \leftarrow n_1 + 1$
$\mathit{PotentialSolutions} \leftarrow []$
foreach $k = 0, \ldots$ **do**
  $A \leftarrow$ MATRIX(l, m)
  $\tilde{\mathcal{P}}_k \leftarrow \{\sum_{j=0}^{m-1} A_{i,j} \cdot p_j(\mathbf{x})\}_{i=0}^{\ell-1}$
  $w \leftarrow (\sum_{i=0}^{\ell-1} \tilde{\mathcal{P}}_k.\text{degrees}()[i]) - n_1$
  $\mathit{CurrPotentialSolutions} \leftarrow$ OUTPUT_POTENTIALS($\tilde{\mathcal{P}}_k$, n, $n1$, w)
  $\mathit{PotentialSolutions}[k] \leftarrow \mathit{CurrPotentialSolutions}$
  foreach $\hat{y} \in \{0,1\}^{n-n_1}$ **do**
    if $\mathit{CurrPotentialSolutions}[\hat{y}][0] = 1$ **then**
      foreach $k_1 = 0, \ldots k-1$ **do**
        if $\mathit{CurrPotentialSolutions}[\hat{y}] = \mathit{PotentialSolutions}[k_1][\hat{y}]$ **then**
          $\mathit{sol} \leftarrow \hat{y} \parallel \mathit{CurrPotentialSolutions}[\hat{y}]$
          if *TEST_SOLUTION($\mathcal{P}$, sol)* **then**
            return *sol*
          end
        end
      end
    end
  end
end

Algorithm 1: SOLVE($\mathcal{P}$, m, n, n_1)

Input: $\tilde{\mathcal{P}}$: $\{r_i(\mathbf{x})\}_{i=0}^{\ell-1}$, n: Integer, n_1: Integer, w: Integer
Result: A two-dimensional list of size $2^{n-n_1} \times (n_1+1)$ containing the z_i bits, y bits and $U_0(y)$ bit.

$(V, ZV[0, \ldots (n_1-1)]) \leftarrow$ COMPUTE_U_VALUES($\tilde{\mathcal{P}}$, n, n_1, w)
$U_0 \leftarrow$ MOB_TRANSFORM($V[0 \ldots |W_w^{n-n_1}|-1]$, $n-n_1$)
foreach $i = 1 \ldots n_1$ **do**
  $U_i \leftarrow$ MOB_TRANSFORM($ZV[i][0, \ldots, |W_{w+1}^{n-n_1}|-1]$, $n-n_1$)
end
$\mathit{Evals}[0 \ldots n_1][0 \ldots 2^{n-n1}-1] \leftarrow \{0\}$
foreach $i = 0 \ldots n_1$ **do**
  $\mathit{Evals}[i][0 \ldots 2^{n-n1}-1] \leftarrow$ MOB_TRANSFORM(U_i.as_array(), $n-n_1$)
end
$\mathit{Out}[0 \ldots 2^{n-n1}-1][0 \ldots n_1] \leftarrow \{0\}$
foreach $\hat{y} \in \{0,1\}^{n-n_1}$ **do**
  if $\mathit{Evals}[0][\hat{y}] = 1$ **then**
    $\mathit{Out}[\hat{y}][0] \leftarrow 1$
    foreach $i = 1 \ldots n_1$ **do**
      $\mathit{Out}[\hat{y}][i] \leftarrow \mathit{Evals}[i][\hat{y}] + 1$
    end
  end
end
return *Out*

Algorithm 2: OUTPUT_POTENTIALS($\tilde{\mathcal{P}}$, n, n_1, w)

Input: $\tilde{\mathcal{P}}$: $\{r_i(\mathbf{x})\}_{i=0}^{\ell-1}$, n_1: Integer, w: Integer
Result: Lists V and ZV containing evaluations of $U_i(y), \forall i \in \{0, \ldots n_1\}, \forall y \in \{y \mid y \in \{0,1\}^{n-n_1}, hw(y) \leq w\}$

$\mathit{Sols}[0 \ldots L-1] \leftarrow$ BRUTEFORCE($\mathcal{P}$, n, $n1$, $w+1$)
$V[0 \ldots |W_w^{n-n1}|-1] \leftarrow \{0\}$
$ZV[0 \ldots n_1][0 \ldots |W_{w+1}^{n-n_1}|-1] \leftarrow \{0\}$
foreach $s \in \mathit{Sols}$ **do**
  $\hat{y}, \hat{z} \leftarrow s[0 \ldots n-n_1-1], s[n-n_1 \ldots n-1]$
  if *HAMMING_WEIGHT($\hat{y}$)* $\leq w$ **then**
    $\mathit{idx} \leftarrow$ INDEX_OF($\hat{y}$, $n-n_1$, w)
    $V[\mathit{idx}]$++
  end
  foreach $i = 1 \ldots n_1$ **do**
    if $z_i = 0$ **then**
      $\mathit{idx} \leftarrow$ INDEX_OF($\hat{y}$, $n-n_1$, $w+1$)
      $ZV[i][\mathit{idx}]$++
    end
  end
end
return $V, ZV[1 \ldots n_1]$

Algorithm 3: COMPUTE_U_VALUES($\tilde{\mathcal{P}}$, n, n_1, w)

Input: For some polynomial p, the degree d, amount of variables n, and a sparsely filled truth-table S.
Result: The full truth-table of p, stored in R.

$R[0 \ldots 2^n-1] \leftarrow \{0\}$;
$D \leftarrow$ DICT(default: 0);
$R[0], D[0] \leftarrow S[0], S[0]$;
foreach $i = 1 \ldots, 2^n - 1$ **do**
  $\mathit{Depth} \leftarrow \min(\text{HAMMING\_WEIGHT}(i), d)$;
  $K \leftarrow$ BITS(i, *Depth*);
  if *HAMMING_WEIGHT(i)* $> d$ **then**
    foreach $j = \mathit{Depth} \ldots, 1$ **do**
      $D[K_{0 \ldots j-1}] \leftarrow D[K_{0 \ldots j-1}] \oplus D[K_{0 \ldots j}]$;
    end
  else
    $Q \leftarrow D[0]$;
    $D[0] \leftarrow S[\text{GRAY}(i)]$;
    foreach $j = 1 \ldots, \mathit{Depth}$ **do**
      if $j < \mathit{Depth}$ **then**
        $\mathit{Tmp} \leftarrow D[K_{0 \ldots j}]$;
      end
      $D[K_{0 \ldots j}] \leftarrow D[K_{0 \ldots j-1}] \oplus Q$;
      if $j < \mathit{Depth}$ **then**
        $Q \leftarrow \mathit{Tmp}$;
      end
    end
  end
  $R[\text{GRAY}(i)] = D[0]$;
end
return R;

Algorithm 4: FES_RECOVER(d, n, S)