Université de Limoges
Faculté des sciences et techniques

Master 2 CRYPTIS

Rapport de projet

UE : Codes correcteurs

ISD algorithm and MDPC encryption

Dalal AL HALABI

Contents

1 ISD algorithm

1.1 Explanation

The idea behind the algorithm:
Given a code C with a generator matrix G and parity check matrix H, our goal is to decode y, where y=mG+e such that e is the error who has a small weight.

1. Guess k positions where the error is equal to zero.
2. Check if the error vector that we calculate satisfies the weight condition.
3. if we are correct, we managed to find the error vector, otherwise we restart with a new permutation.

The basics of the algorithm:
For any invertible matrix $U \in \mathbb{F}_2^{(n-k)\times(n-k)}$ and for any permutation matrix $P \in \mathbb{F}_2^{n\times n}$ we have: $(eH^T = s) \iff (e'H'^T = s')$ where $H' = UHP$ and $s' = sU^T$ and $e' = eP$.
Proof : $e'H'^T = (eP)(UHP)^T = (eP)(P^T H^T U^T) = eH^T U^T = sU^T = s'$
Using the logic above, our goal is to find e. By finding the correct permutation P, we manage to write $H' = UHP$ under the form $I|H$. Since $e'H'^T = s'$, if s' is of correct weight, $e' = (s', 0)$ and e can be recovered by $e = e'P^{-1}$.

1.2 The algorithm and code

This pseudo code is the one followed in the program Ex1 file contains ISD.c an independednt algorithm that you only need to feed it the parameters, the matrix H is gnerated on the inside using a special function and can be altered.

Algorithm 1 Prange-ISD

Input: $H \in \mathbb{F}_2^{(n-k)\times(n-k)}$.
Output: $e \in \mathbb{F}_2^n$ `such that` $eH^T = s$ `and` $||e|| \leq w$.

do
`Sample a uniformly random permutation` $P \in \mathbb{F}_2^{n\times n}$.
`Compute` HP.
`When it exists, find` $U \in \mathbb{F}_2^{(n-k)\times(n-k)}$ `such that` $UHP = (I_{n-k}|\widetilde{H})$
while $||sU^T|| > w$
`return` $(sU^T, 0)P^{-1}$

1.3 Complexity of the algorithm

The complexity of the Prange algorithm to decode an error of weight w associated to a codeword of a random [n, k] code is the cost of matrix inversion times the average work factor (the inverse of the probability to find an adequate set of columns which does not intersect with the error).
The number of cases for which the algorithm succeeds, corresponds to the number of choices of k positions among n-w positions (the total number of positions minus the w forbidden positions of the error) and the total number of choices is choosing k positions among n: $\frac{\binom{n-w}{k}}{\binom{n}{k}} = \frac{\binom{n-k}{w}}{\binom{n}{w}}$. Hence, the complexity of the algorithm is :

$$O((n-k)^3 \frac{\binom{n}{k}}{\binom{n-k}{w}})$$

2 The results + remarks

The computation time for ISD using the parameters 400, 200, 20 is :106 ms, as for the parameters 1000, 500, 10, the timing is : 1825 ms.

3 MDPC encryption

3.1 The BitFlip principle

The rows of the parity Check matrix H of a code C are refereed to as check sums, and elements of these check sums are called bits
In the hard-decision decoding algorithm BitFlipping, the decoder computes all parity-check sums and then flips bits with a number of unsatisfied check sums larger than a predetermined thresh hold.Parity checks are then recomputed using these new values. The process is repeated until all the check sums are satisfied or until some maximum number of iterations is reached

3.2 The BitFlipping algorithm

The first algorithm that i coded is the BitFlipping algorithm, using the following pseudocode. The code is contained in the file under the name bitflip.c.

Algorithm 2 BitFlipping(h_0, h_1, s, T, t)

Input: h_0, h_1 and $s = h_1e_1 + h_0e_0$, threshold value T required to flip a bit, weight t of e_0 and e_1.

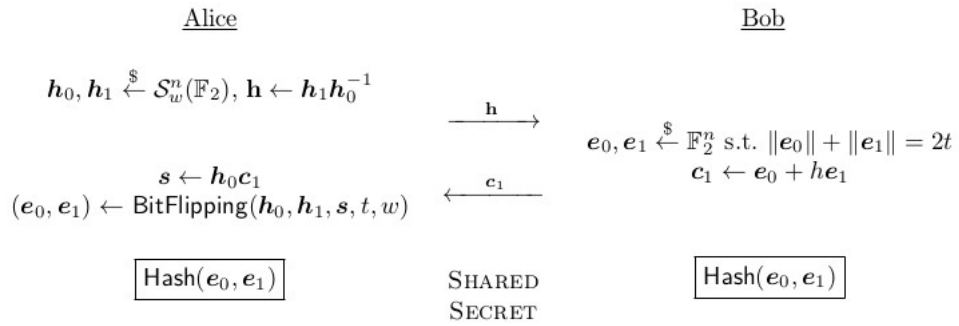
Output: (e_0, e_1) if the algorithm succeeds, $\perp$ otherwise.

$(u, v) \leftarrow (0, 0) \in (\mathbb{F}_2^n)^2$, $H \leftarrow (rot(-h_0)^T, rot(h_1)^T) \in \mathbb{F}_2^{n\times 2n}$
$syndrome \leftarrow s$.
while [$||u|| \neq$ t `or` $||v|| \neq$t] `and` $||syndrome|| \neq 0$ **do**
  $sum \leftarrow$syndrome $\times H$. `/No modular reduction, values in` $\mathbb{Z}$`/.`
  $flipped\_positions \leftarrow 0 \in \mathbb{F}_2^{2n}$.
  for $i \in [[0, 2n-1]]$ **do**
    if $sum[i] \leq T$ **then**
      $flipped\_positions[i] = flipped\_positions[i] \oplus 1$.
    end if
  end for
  $(u, v) = (u, v) \oplus flipped\_positions$.
end while
$syndrome = syndrome - H \times (flipped\_positions)^T$.
if $s - H \times (u, v)^T \neq 0$ **then**
  `return` $\perp$.
else
  `return` (u, v).
end if

4 The MDPC encryption

4.1 Key exchange

$S_w^n(\mathbb{F}_2)$ is the set of all words of weight w in $\mathbb{F}_2$.
The key exchange protocol is described as follows:



Why it works: $\boldsymbol{h}_0\boldsymbol{c}_1 = \boldsymbol{e}_0\boldsymbol{h}_0 + \boldsymbol{e}_1\boldsymbol{h}_1$, a syndrome for the small weight vector $(\boldsymbol{e}_0, \boldsymbol{e}_1)$ associated to the MDPC matrix derived from $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$, which can be decoded by the BitFlipping algorithm

[!htb]

In the file name ex2, key exchange file, the code corresponding to key exchange is represented in the op.c file , when ran, the file will create a file and print the output inside of it for the next operation. the code is somewhat complex as i didn't rely on prexisting code and implementing everything by myself, to follow the operation, the first Alice interaction is presented in the function Alice1, first bob interaction is presented in bob1, second alice interaction is in Alice 2, again this code is fully independent m just feed it the requested input, like the parameters of the key exchange n, t, w. The average timing is around 6.7 seconds.

4.2 Encryption

Using the same notations as above, Alice generates a public key h, Bob uses this public key and does the exchange with Alice so that they both have the secret hash(e_0, e_1), except Bob encrypts a message m by Xoring it with this secret i.e. the ciphertext is $c = m \oplus Hash(e_0, e_1)$, and sends it to ALice with his interaction
The code is in the encrypt.c file in the EX2 file, it is extremly similar to key exchange code, as it is the same procedure, with only minor tweaking. The average timing is around 6.8 seconds for the realistic parameters

4.3 Decryption

After receiving c, ALice then xor it with her calculated secret $Hash(e_0, e_1)$ and re obtain the message m. The code is the decrypt.c file in the decryption folder that belongs to the EX2 folder.. The average timing is around 6.8 seconds.

5 Remarks

The project is included in the **pr_co** zip file along with s copy of this report, it contains 2 folders ex1 , that is focused on ex1 problem and ex2 folder that focuses o ex2 problems.
For reference, i used inria pseudo code for matrix inversion, as for binary operation i relied on pseudo code that i found online and logic. the code inside the files is divided into several parts each parts deals with one or multiple tasks that i felt are necessary to accomplish the tasks , but maybe made the code a little too long to read, hopefully the comments solve this problem.
Thank you.