A method for finding codewords of small weight *

Jacques Stern
Équipe de logique
Université Paris 7
et
Département de mathématiques et informatique
École Normale Supérieure

****

Abstract

We describe a probabilistic algorithm, which can be used to discover words of small weight in a linear binary code. The work-factor of the algorithm is asymptotically quite large but the method can be applied for codes of a medium size. Typical instances that are investigated are codewords of weight 20 in a code of length 300 and dimension 150.

1 Introduction

It is known that the problem of finding a codeword with a given weight in a linear binary code is NP-complete [1] .Therefore, it seems possible to use this problem in order to build signature and authentication algorithms in the spirit of the Fiat-Shamir protocol [2] . Such an idea has recently been considered by Harari [3]. The validity of such a scheme would presumably be based upon the secret knowledge of codewords with a given small weight. Thus, it is important to investigate methods that can efficiently discover such codewords in order to estimate lower bounds ensuring the security of such a secret information .

In the present paper, we describe a probabilistic algorithm for finding a codeword with a given weight w in a linear binary code of length n and

*research supported by the P.R.C. Mathématiques et Informatique

dimension k. The algorithm consists of a basic attack, whose iteration eventually produces the required codeword. Unfortunately, the probability of success of the basic attack is asymptotically very low. Nevertheless, a precise analysis shows that it can still be significant for codes of a medium size. Typical values that are investigated are $n = 300, k = 150, w = 20$. Thus, our algorithm puts security restrictions on any cryptographic scheme based upon the secret knowledge of codewords with small weight.

2 Description of the basic attack

We assume that a code of length n and dimension k is given and we let H denote the parity check matrix of this code .

$$H = (H_{j,m}) \begin{array}{ccccc} 1 & \leq & j & \leq & n-k \\ 1 & \leq & m & \leq & n \end{array}$$

What we have to do is to find a solution of the equation

$$Hx = 0$$

whose weight $\mid x \mid$ is exactly w.

Our algorithm uses as parameters two positive integers p, l which are assumed to take small values. Typical values of p, l will be discussed further on. The basic attack includes the following steps.

- step 1 Perform gaussian elimination in the following way:
  for $i := 1$ to $n - k$ choose randomly a column C of matrix H among those which have not yet been chosen; let m_i be the index of this column;
  select, if possible, a pivot belonging to column C and some unmarked row numbered k_i ;
  mark this row;
  make the other elements of the column equal to 0 by linear combinations of rows;
  in case no pivot has been found revise the choice of column C.

- step 2 Randomly split the indices of the columns of H that have not been selected in step 1 into two subsets denoted by X and Y respectively. Each index is added either to X or to Y with probability 1/2 so that X and Y do not have necessarily the same size.

- step 3 Randomly choose a set J consisting of l indices $\leq (n-k)$ say $j_1 \cdots j_l$.
- step 4 For every subset A of X with p elements, compute the l-bit vector $\pi(A)$, obtained by adding up the columns of the matrix

$$(H_{j,m})_{j \in J, m \in A}$$

and perform a similar computation for every subset B of Y with p elements.

- step 5 List all pairs A, B such that $\pi(A) = \pi(B)$; for every pair A, B in the above list, compute the sum of all members of $A \cup B$; if one of the vectors that has just been computed , say V, is exactly of weight $w - 2p$, then return the solution x defined by

$x_m = 1$ iff $(m \in A \cup B$ or for some $i \leq n-k$ $(m = m_i$ and $V_{k_i} = 1))$;

if no vector V of weight $w - 2p$ can be found then the basic attack has failed.

3 Correctness and probability of success of the basic attack

proposition 1 *Any solution returned by the basic attack is a codeword of weight w .*

Proof We keep the notations introduced in the description of the previous paragraph.Given a solution x returned by the algorithm and coming from some V computed in step 5 , we define n-bit vectors y, z by the following

$$y_m = 1 \text{ iff } m \in A \cup B$$

$$z_m = 1 \text{ iff } m = m_i \text{ and } V_{k_i} = 1$$

Clearly, y is of weight $2p$, z is of weight $w - 2p$ and we have

$$x = y + z$$

$$y.z = 0$$

so that the weight of x is precisely w. Now, if Γ is the value of matrix H at the end of the algorithm and if Γ^m denotes the m-th column of Γ , we have

$$V = \sum_{m=1}^{n} y_m \Gamma^m$$

and

$$V = \sum_{m=1}^{n} z_m \Gamma^m$$

so that

$$\sum_{m=1}^{n} x_m \Gamma^m = 0$$

hence, x is a word of the given code.

In the reverse direction, we have

proposition 2 *Let x be a codeword of weight w . x will be returned by the basic attack provided the following properties hold*

i) *x has exactly p non-zero coordinates with indices in X*

ii) *x has exactly p non-zero coordinates with indices in Y*

iii) *if N_J is the set of indices m_i such that k_i is a member of J, then, x takes value 0 on all of N_J*

Proof We let A (resp. B) be the set of indices in X (resp. Y) such that the corresponding coordinate of x is non-zero. Both A and B have p elements and, because x is a codeword, we have

$$\sum_{m \in A \cup B} \Gamma^m = \sum_{m \notin X \cup Y} x_m \Gamma^m = \sum_{i=1}^{n-k} x_{m_i} \Gamma^{m_i}$$

so that, by property iii, $\pi(A) + \pi(B)$ has all its coordinates equal to 0. It follows that step 5 of the algorithm will consider vector

$$V = \sum_{m \in A \cup B} \Gamma^m$$

and therefore find x as a corresponding solution.

In order to evaluate the probability that our algorithm discovers a given solution x with weight w , we have to compute the probability that the random choices performed by the algorithm yield a situation where statements i, ii, iii of the previous proposition are satisfied. This can be checked from the triple (X, Y, N_J) and the corresponding triples will be called favourable with respect to x . What will be actually evaluated is the probability of getting a favourable triple by choosing randomly a triple (X, Y, N) , where X, Y, N are disjoint sets of indices at most n, $X \cup Y$ has cardinality k and N cardinality l. We think that it is a reasonable measure of the probability of finding the solution x although it neglects the role of the possible revisions in the choices of the columns that are performed during step 1.

theorem 1 *The probability of getting a triple favourable w.r.t. x is the product of the three following real numbers*

$$\binom{w}{2p}\binom{n-w}{k-2p}/\binom{n}{k}$$

$$\binom{2p}{p}/4^p$$

$$\binom{n-k-w+2p}{l}/\binom{n-k}{l}$$

Proof The first number is the probability of the event
" $X \cup Y$ has exactly $2p$ elements such that the corresponding coordinates of x are non-zero".
We note that, once this event is realized, properties i and ii of the definition of a favourable triple become equivalent and independent of property iii. The second and third numbers can be interpreted as the conditional probabilities of these two independant events , whose intersection is the set of favourable triples . The result follows.

If we take $n = 300$, $k = 150$, $w = 20$, we find that the probability of success of the basic attack corresponding to $p = 3$, $l = 12$ is 0.0031 which is a quite significant value as approximately 320 trials should lead to a favourable triple and therefore to the disclosure of a given solution. Of

course, many other solutions may have been discovered in between so that the average number of trials needed to disclose at least one codeword of weight w can be considerably smaller than expected. We also note that the probability of success decreases slowly as l increases. For example, if we change l to 16 and keep the other parameters as above, it only goes down to 0.0020.

Clearly, the asymptotic behaviour of the expected number of trials, and therefore the overall complexity of the full algorithm is quite bad. Nevertheless, as shown by the following theorem, it can still be much better than the brute force algorithm which requires a number of trials essentially equal to the binomial coefficient

$$\left(\begin{array}{c} n \\ w \end{array} \right)$$

theorem 2 *Assume that p is fixed and that, for each n, the parameters l, w, k are chosen in such a way that*

- *k/n has a limit α distinct from 0 and 1*
- *l and w are $o(\sqrt{n})$*

then, for any β, $0 < \beta < (1 - \alpha)$,there is a fixed constant K, depending on p, such that the probability of the set of favourable triples is bounded from below by $K\beta^w$.

We only sketch the **proof** of this result. If w is $o(\sqrt{n})$, then, it is easily seen that

$$(n - w)! \sim \frac{n!}{n^w}$$

Using repeatedly analogous relations, one finds that the first number appearing in the previous theorem is equivalent to

$$\left(\begin{array}{c} w \\ 2p \end{array} \right) \frac{k^{2p}(n-k)^{w-2p}}{n^w}$$

which is equal to

$$\left(\begin{array}{c} w \\ 2p \end{array} \right) \left(\frac{k}{n}\right)^{2p} \left(\frac{n-k}{n}\right)^{w-2p}$$

so that, asymptotically, it is larger than β^w.
The second number is a constant depending on p. As for the last one, using

the same type of approximations as above, it can be seen to be equivalent to

$$\frac{(n-k-l)^{w-2p}}{(n-k)^{w-2p}}$$

that is

$$\left(1-\frac{l}{n-k}\right)^{w-2p}$$

whose limit is 1. Finally, in view of theorem 1, the probability of success is bounded from below by $K\beta^w$, for some K.

4 Complexity of the basic attack

In this section we show that the number of operations required by our algorithm is polynomial, provided p is fixed. Of course, the degree of the polynomial which bounds the complexity increases with p so that the algorithm can be of practical value only for small p.

The time complexity of the algorithm is dominated by steps 1,4 and 5.

- The gaussian elimination performed in step 1 requires a number of bit operations of order

$$1/2(n-k)^3 + k(n-k)^2$$

- Step 4 consists of binary additions whose number is approximately

$$2lp\begin{pmatrix} k/2 \\ p \end{pmatrix}$$

- In order to estimate the complexity of step 5, we make the assumption that the function π has roughly a uniform distribution. Any l-tuple of bits is hit approximately by $\begin{pmatrix} k/2 \\ p \end{pmatrix}/2^l$ many values of A and similarly for B. Thus, the algorithm has to consider a number of pairs (A,B) , which is close to

$$\begin{pmatrix} k/2 \\ p \end{pmatrix}^2/2^l$$

and this leads to

$$2p(n-k)\left(\begin{array}{c} k/2 \\ p \end{array}\right)^2 /2^l$$

bit operations.

In the case of the typical values considered in the previous section, namely

$$n = 300\ ,\ k = 150\ ,\ w = 20\ ,\ p = 3\ ,\ l = 12$$

one finds a number of operations which is $5\ 10^6$ for step 1, $5\ 10^5$ for step 4 and 10^9 for the final step. If we replace l by 16 and keep the other values, the last step goes down to $6.2\ 10^7$, a figure which clearly allows the implementation of the algorithm.

References

[1] E.R.Berlekamp, R.J.Mc Eliece and H.C.A. Van Tilborg, On the inherent intractability of certain coding problems, *IEEE Trans. Inform. Theory*, (1978) 384-386.

[2] A. Fiat and A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, *Proceedings of Crypto 86*, Santa-Barbara (1986), 181-187.

[3] S. Harari, *This volume.*