A New Class Of Attacks On McEliece Public-Key And Related Cryptosystems

A Kh. Al Jabri (Prof)
Elect. Eng. Dept., King Saud University,
P.O.Box 800, Riyadh 11421, Saudi Arabia
e-mail: aljabri@ksu.edu.sa

Abstract- Different attacks are proposed in the literature to cryptanalyze McEliece public-key cryptosystem. Most of these attacks are based on some form of the information set decoding approach. In this paper, we propose a new attack based on an algorithm that can be viewed as a generalized threshold decoding algorithm. Applying this algorithm on the McEliece system shows that the system with its original parameters is not secure.

1 Introduction

The decoding problem of general linear block codes is known to be **NP** hard. In practice, codes are usually designed with certain algebraic structures that can be exploited to speed up the encoding and decoding processes. This is, however, at the expense, for most codes, of a decreasing code efficiency for correcting errors.

Randomly generated codes, on the other hand, are typically good [4]. In fact, the minimum distance, d, of an (n, k) random code is related to its rate, k/n, by

$$1 - \frac{k}{n} \approx h(\frac{d}{n}),$$

where $h(x)$, $0 \leq x \leq 1$, is the binary entropy function defined by $x \ \log_2(1/x) + (1 - x) \ \log_2(1/(1-x))$. For half rate codes, for example, the above relation can be approximated by $d \approx 0.11n$ or up to $0.055n$ errors can be corrected by these codes.

The problem with these codes, however, is the absence of efficient decoding algorithms. Based on this McEliece proposed a public-key cryptosystem [4].

McEliece System:

In this system, the public key, $\mathbf{G}$, is a $k \times n$ matrix that is a product of three private keys: $\mathbf{S}$, $\mathbf{G}'$, and $\mathbf{P}$ where $\mathbf{S}$ is a $k \times k$ scrambling matrix, $\mathbf{G}'$ is $k \times n$ Goppa code generator matrix, and $\mathbf{P}$ is a $n \times n$ permutation matrix. Both $\mathbf{S}$ and $\mathbf{P}$ provide the required randomization effect on $\mathbf{G}'$. For Goppa codes, a fast decoding algorithm exists[4]. For the equivalent code $\mathbf{G}$, however, this is not the case if the private keys are kept secret.

The ciphertext vector $\mathbf{c}$ corresponding to an information vector $\mathbf{u}$ is given by

$$\mathbf{c} = \mathbf{uG} \oplus \mathbf{e},$$

where $\mathbf{e}$ is an n dimensional binary vector of weight t or less where t is the error correcting capability of the code. To decode, the received vector $\mathbf{y}$ is first multiplied by $\mathbf{P}$ and the fast decoding algorithm of the Goppa is then used to remove the errors. The result is then multiplied by $\mathbf{S}^{-1}$ to obtain the information vector $\mathbf{u}$.

Previous Attacks:

Different attacks on this system have been proposed in the literature. Most of them are based on the information set decoding algorithm [4]. The

algorithm is based on a random selection of k bits from $\mathbf{y}$ on the hope that these bits are not in error. The selected subvector is then multiplied by the inverse of the $k \times k$ matrix composed of the k columns of $\mathbf{G}$ corresponding to the selected positions of $\mathbf{y}$. The result is the decrypted vector. To check its validity, one can use the approach proposed in [4]. If successful then stop else repeat all processes all over again.

2. The New Approach: (Statistical Decoding)

The algorithm is of a statistical nature and can be viewed as a generalized threshold decoding algorithm [4]. In classical threshold decoding, a set of appropriately selected vectors from the dual space of the code is used to detect and locate all the error positions. The proposed algorithm, however, is based on generating a "balanced" projection set $\mathcal{H}_w$. This is composed of randomly generated vectors from the dual space of the code with small weight w (large weights works as well)[1]. To perform decoding, a new set, $\mathcal{H}_r$, is formed by restricting $\mathcal{H}_w$ into the subset containing vectors satisfying the condition $\mathbf{y}\mathbf{h}^T = 1$. The set $\mathcal{H}_r$ will be called the restricted set. If the vectors of $\mathcal{H}_r$ are summed in R^n, then the t erroneous positions will be the ones corresponding to the maximum (minimum) t values of the resulting vector. Justification for why the algorithm works will be discussed in the next section. The outline of the algorithm is given below.

The Statistical Decoding Algorithm:

Input: $\mathcal{H}_w$, $\mathbf{y}$.

Output $\mathbf{u}$, the information vector.

1. Calculate the error-locating vectors $\mathbf{v}$.

$$\mathbf{v} = \sum_{\mathbf{h} \in \mathcal{H}_w} (\mathbf{y}\mathbf{h}^T)\mathbf{h}.$$

2. Calculate $\mathbf{u}_i$ where

$$\mathbf{u}_i = \mathbf{y}_{ki}\, \mathbf{G}_{ki}^{-1} \quad i = 1, 2,$$

where $\mathbf{y}_{ki}$, $\mathbf{G}_{ki}$, $i = 1, 2$ are the subvectors of $\mathbf{y}$ and the corresponding submatrices of $\mathbf{G}$ corresponding to the t positions in $\mathbf{v}$ with the maximum and minimum t values, respectively.

3. Check

$$\text{weight}(\mathbf{u}_i\mathbf{G} \oplus \mathbf{y}) \quad i = 1, 2.$$

Choose $\mathbf{u}_i$ that yields weight $\leq t$.

Consider the following example:

Example: $(32, 16, 5)$ **Random code:**

This is a randomly generated $(32, 16, 2)$ code. It can be shown that the minimum distance for this code is 5. That is the code can correct up to 2 errors. The $\mathbf{P}$ part of the generator matrix $\mathbf{G} = [I : \mathbf{P}]$ for this code is given by

$$P = \begin{bmatrix}
0&1&1&1&1&1&1&0&1&0&1&1&0&0&1&0\\
0&0&1&1&0&1&0&0&1&0&0&1&1&0&0&0\\
0&0&0&1&0&0&0&0&0&1&0&1&1&1&1&1\\
0&1&1&0&0&1&0&1&1&0&1&0&0&1&1&0\\
1&1&0&1&1&0&0&1&1&0&0&1&1&1&1&1\\
1&0&0&1&0&1&0&0&0&1&1&1&1&0&0&1\\
1&0&1&1&0&0&1&0&0&0&0&1&1&0&0&1\\
1&1&1&1&0&0&1&1&0&1&1&1&0&1&1&0\\
1&0&1&1&1&1&0&0&0&1&1&1&1&1&1&1\\
0&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1&1\\
0&0&1&0&1&1&0&0&1&0&1&1&1&1&1&1\\
1&1&1&0&0&1&1&0&0&0&1&0&0&1&0&1\\
1&1&1&1&1&0&1&1&1&1&1&0&0&1&0&1\\
1&1&1&1&1&1&0&0&1&1&1&1&1&0&0&0&1\\
0&1&1&0&1&1&0&1&1&1&1&0&1&0&0&1\\
0&1&1&1&1&1&1&1&1&1&0&1&0&1&0&1
\end{bmatrix}$$

A random search is performed for $\mathcal{H}_w$ and it is found that the following 28 vectors are sufficient for decoding.

$$\begin{bmatrix}
0&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&0&1&0\\
1&1&1&1&0&1&0&1&1&1&1&1&1&0&1&1&1&0&0&1&1&1&0&1&1&1&1&1&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&0&1&0&1&0&0&1&1&1&0\\
1&1&1&1&0&1&1&1&1&1&1&0&1&1&1&1&0&1&1&0&1&1&1&1&0&0&1&0&1&1&1&0\\
1&1&1&0&1&0&1&0&0&1&0&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&0\\
0&0&1&1&1&0&0&1&0&1&1&1&1&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&1&1&1&0\\
1&1&1&1&1&1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&0&0&1&0&1&1&0&0&1&1\\
1&1&1&0&1&1&0&1&1&1&1&1&0&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&0&1&0&1\\
0&1&1&1&0&1&1&0&1&1&1&1&0&1&1&1&1&1&1&1&1&0&1&0&1&1&1&1&0&1&1&0&1\\
1&1&0&1&1&1&1&1&0&0&1&1&1&0&1&0&1&1&1&1&1&1&1&1&1&1&0&1&0&1&1&0&1\\
1&0&1&1&0&1&1&1&0&0&1&0&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&0&1\\
1&0&1&1&0&1&1&1&1&1&0&1&1&1&0&1&1&1&0&0&1&1&1&0&1&1&1&1&1&1&1&0&1\\
0&0&0&0&1&0&1&1&1&1&1&1&1&0&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&0&1&0&1\\
1&1&0&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&0&1&1&0&1&0&1&1&1&1&0&1&0&1&1
\end{bmatrix}$$

[1] The set $\mathcal{H}_w$ has to be generated and stored in advance. There are many efficient algorithms for such generation. (See for example [1]).

```
1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1
1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1
1 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1
1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1
0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1
1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1
1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1
1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 1
1 0 1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1
1 0 0 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1
1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1
0 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Now suppose a single error has occurred in position 10. In this case, $\mathbf{v}$ will be

```
14  16  18  14  15  16  13  15  17  21  15  15  15 16  17  15
15  14  17  15  17  17  15  13  15  17  17  15  17  17  16  15
```

Note that position 10 has the highest value (21 in this case) among the values of $\mathbf{v}$. On the other hand suppose that two errors occurred in position 10 and 20, then $\mathbf{v}$ will be

```
9   9   7  11   9  10  10  12  10   6   9   8   8  10   9  10
8  10   8   6  10   8   9   9   9   8   9   9   9   9   9  11
```

Note again that position 10 and position 20 have the smallest values (6 in this case) among the values of $\mathbf{v}$.

Similar calculations can be done for other error patterns. One can then set different thresholds to detect and locate different kind of correctable errors patterns. From this one can find a single threshold for every error pattern of weigh i, $0 \leq i \leq t$.

3. Asymptotic Behavior of the Algorithm:

Large codes such as the one proposed by McEliece, generally, behave like randomly generated codes and one can reasonably assume that the binomial distribution is a good approximation for the weights [4]. We now observe that the probability of observing a value of one in any position of the binary vector $\mathbf{h} \in \mathcal{H}_w$ is w/n. Restricting the set $\mathcal{H}_w$ to the new set $\mathcal{H}_r = \{\mathbf{h} : \mathbf{e}\mathbf{h}^T = 1\}$ will bias these probabilities in both the erroneous and the error free positions into two different values, p and q respectively. In fact, one can prove the following theorem.

Theorem 1:

For vectors in $\mathcal{H}_r$, the probability, p, of obtaining a value of 1 in an erroneous position is given by

$$p = \frac{\sum_{m\ odd} \binom{n-t}{w-m}\binom{t-1}{m-1}}{\sum_{m\ odd} \binom{t}{m}\binom{n-t}{w-m}}$$

and the probability, q, of obtaining a value of 1 in an error free position is given by

$$q = \frac{\sum_{m\ odd} \binom{n-t-1}{w-m-1}\binom{t}{m}}{\sum_{m\ odd} \binom{t}{m}\binom{n-t}{w-m}} \qquad \blacksquare$$

The idea of the algorithm is based on the observation that some of $\mathbf{h} \in \mathcal{H}$ provide error detection information. In the detection process (say odd detection; $\mathbf{y}\ \mathbf{h}^T$=1), the $\mathbf{h}$ vector acts like a mask on the vector $\mathbf{y}$. That is, if $\mathbf{y} = (y_1, y_2, \ldots, y_n) = \mathbf{c} \oplus \mathbf{e}$ is the received vector where $\mathbf{e}$ is the error vector of weight less than or equal to t, and $\mathbf{h} = (h_1, h_2, \ldots, h_n)$, then the sum $\sum_{i=1}^{n} \mathbf{y}_i\mathbf{h}_i$ will only include those terms for which $h_i = 1$, $1 \leq i \leq n$. Since $\mathbf{c}\mathbf{h}^T = 0$, the vector $\mathbf{y}$ can be replaced by $\mathbf{e}$.

Now we make an important observation that, if the weight of $\mathbf{h}$ is large, then it is likely that most of the errors will be masked by $\mathbf{h}$. This has a statistical implication. If one recursively add such $\mathbf{h}$ vectors, then he will end with numbers that points out to the positions of the errors since these positions are common to most of these vectors.

The Work Factor:

We have interest in estimating the number of $\mathbf{h}$ vectors, N, required for there to be a 0.95 probability that the relative frequency estimate for the probability of an error event would be within ϵ of p. For large codes, it is noticed that the difference between p and q is very small. This puts some restriction on the value of ϵ. One can use the Central Limit Theorem to bound this number. Let $f_e(N)$ be the relative frequency of the error event in some position. Since $f_e(N)$ has mean p and variance $p(1-p)$, then it can be shown [5] that for a 0.95 reliability

$$N = 625 \times 10^{-6} p(1-p)\epsilon^{-2}.$$

This is the number of vectors in $\mathcal{H}_w$ required to detect all error patterns. Now assume that this set is available. To find the number of computation required to perform the second step in the statistical decoding algorithm, let m be the number of vectors that can be tested and added per second. In this case, the time, t_d, required to decrypt is given by

$$t_d = \frac{|\mathcal{H}_w|}{m} \quad \text{seconds.}$$

4. Results and Discussion:

As can be seen from the above discussion, the proposed attack involves a time-space trade of. One first needs to perform some precomputation to construct the set $\mathcal{H}_w$. Once this set is available, then all decryptions can be performed using the same set.

Let us consider the McEliece system with its original parameters $(1024, 512, 51)$. In this case we have

$$\begin{aligned} \mathrm{p} &= 0.8994152623 \\ \mathrm{q} &= 0.8994139996 \end{aligned}$$

Using these probabilities, the amount of $\mathbf{h}$ vectors required to identify the erroneous places in the McEliece system is 2^{38}. Our experimentation shows that for a single 700 MHz PENTIUM II processor, this will require around 2^9 hrs per decryption and can be significantly reduced using parallel computations. It is worth mentioning that storage of such sizes are within the reach of today's technology.

5. Conclusions

This paper has introduced a new decoding algorithm for general linear block codes called here "statistical decoding". The algorithm is of a different flavor from the conventional threshold decoding algorithm and it exploits, in a statistical sense, the statistical information provided by the classical syndrome decoding. The algorithm is then used to cryptanalyze McEliece public-key cryptosystem. Results show that all the computation and storage requirements to break the system are within the reach of today's technology. This suggests that the parameters of the system should be increased. A possible choice is $(2048, 1024)$. The size of the public-key, however, will be too large for any practical consideration.

Acknowledgment

The author would like to thank Prof. Paddy Farrell for the discussion regarding the subject of the paper.

References:

1. A. Canteaut and F. Chabaud,"A New Algorithm for Finding Minimum Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 611", IEEE Trans. Inform. Theory, vol. IT-44(1), pp. 367-378, 1998.

2. P.J. Lee and E.F. Brickell,"An Observation on the Security of McEliece's Public-Key Cryptosystem", in Lecture Notes in Computer Science 330, Advances in Cryptology: Proc. Eurocrypt'88, C.G. Gunther, Ed., Davos, Swizerland, May 25-27, 1988, pp. 275-280, Berlin: Springer-Verlag, 1988.

3. R. J. McEliece, "A Public-Key Cryptosystem Based on Algebraic Coding Theory, DSN Progress Report 42-44, pp. 114-116, Jet Propulsion Laboratory, CA, Jan-Feb 1978.

4. F.J. McWilliams and N.J. Sloane," *The theory of error correcting codes*", North Publishing Co. 3rd ed., North Mathematical Library , Vol. 16, Netherlands 1983.

5. A. Papoulis, "*Probability, Random Variables and Stochastic Processes*", 2nd ed.,McGraw-Hill Book Company, New York, 1984.