COMP3697 Codes and Cryptography 2023-24 Summative Report

Alex Read jjxc38

---

2. Attacking 'Learning Without Errors'

By consistently setting the error distribution χ to be 0, the LWE secret s can be compromised by finding a solution to a system of linear equations.

Given the equation, where $e = 0$

$$b = A \cdot s + e$$

$$b = A \cdot s.$$

Considering b and A as known entities from the public key, we can solve the system of linear equations

To solve for s, we apply the LU decomposition method using the formula.

$$s = (A^T A)^{-1} A^T b.$$

To ensure that the results conform to Gaussian fields bounded by a modulus q, this computation is performed within the context of Gaussian field matrices.

Decryption:

By utilizing the discovered secret key, denoted as s, we can input the ciphertext and s into the decryption function to successfully decrypt the ciphertext.

Correctness, discussion of other attacks

This method succeeds when error values are omitted. If we let $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ where $e_i \in \{0, 1, 2, \ldots\}$. This method would either lead to an unsolvable system of linear equations or result in a different solution and secret key s. Any method of solving a system of linear equations would have been valid to crack this instance of LWE. The time complexity is $O(n^3)$, the same time complexity as using LU Decomposition.

3. Attacking 'Learning With A Few Errors'

.

As the error distribution χ is usually 0 with a very low probability of otherwise being 1 or -1, most linear equations within the system of linear equations are expected to be error-free. Thus, we may select a random subset of size n from the equations and hope that this smaller system of linear equations is free of errors. - i.e. in the subset, every $e_i = 0$. We can then solve this set of linear equations. If consistently chosen subsets yield the same solution, this

likely represents the actual secret key s - which compromises the entire set of equations.

I have already shown that finding a solution to LWE can be viewed as finding a solution to $b = A \cdot s + e$ and that finding s allows for decryption of the ciphertext.

In this case, the system of linear equations may appear as follows

$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}$$

My algorithm generates all possible subsets of n rows from a system of linear equations and solves them iteratively. A solution is returned when it recurs θ times among these subsets, i.e. a subset of equations produces the same secret key s. If no solution recurs θ times, the most frequently occurring solution across all subsets is returned.

As an example, one random subset that could be selected may look as follows:

$$\begin{bmatrix} b_5 \\ b_3 \\ \vdots \\ b_9 \end{bmatrix} = \begin{bmatrix} a_{51} & a_{52} & \cdots & a_{5n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{91} & a_{92} & \cdots & a_{9n} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$b = A \cdot s.$$

This system would then be solved to get s and produce a solution, which is then saved to count future occurrences. In this case, all of the error values e_i are 0, and thus, this smaller system of linear equations would most likely produce the correct secret key s - more on 'most likely' later.

Design Choices

The frequency at which a solution must be observed is denoted by θ. More specifically, it refers to the number of instances where the secret key s must be determined while resolving a subset system of linear equations. I have chosen θ to be 4, reasoning that it's highly improbable to encounter three subsets with identical yet incorrect solutions before finding the actual solution.

Correctness, discussion of other attacks

This approach is effective because the system of linear equations typically exhibits a low error rate. Particularly when a subset of more than n error-free equations exists, this algorithm's effectiveness and time complexity improve with the increase in the number of equations where $e_i = 0$.

If $\theta \times n$ exceeds the number of possible combinations of equations that do not contain a zero, the algorithm fails to function effectively.

The worst-case time complexity of this algorithm is $O\left(\binom{m}{n} \cdot n^3\right)$

To loop over every possible combination of n equations is $O\left(\binom{m}{n}\right)$ and to solve each subset (combination) of linear equations is $O(n^3)$.

The algorithm might fail if the threshold θ is too low or if it samples subsets of linear equations solving for the same, incorrect value s. A more reliable but computationally intensive alternative is to generate all possible error vectors e, ordered by length. Applying these errors to the public key b and solving for s. This method has a computational complexity of $O(3^m)$ and would have been chosen if the presence of -1 and 1 were more common in the error vector e.

4. Attacking 'Learning With Errors'

The previously mentioned algorithm works consistently for the test cases presented in this problem. I've chosen to use this method based on the assurance that employing this algorithm will not impact my marks.

Why does this work / How can it be fixed?

Although no details were provided about the error distribution χ in this question, I presume that this method is consistently effective for reasons similar to those mentioned in the previous question. Its efficiency likely stems from the system of linear equations generally having a low error rate, meaning that error values are frequently set to zero $e_i = 0$. This implementation can be fixed by making sure the presence of 0's in the error vector e is smaller than n.

Discussion of other attacks

I opted for this approach instead of alternatives like enumerating all lattice points to identify the smallest vector and determine the values of e and s (enumeration) Simply because my approach is less computationally expensive ie $O\left(\binom{m}{n} \cdot n^3\right)$ is better than $O(2^{m+1})$. In the implementation I used, my method delivers superior and more rapid outcomes. Nonetheless, in practical scenarios, I would employ the enumeration technique or a different strategy such as sieving, as they are more dependable and don't require a specific floor in the test cases.