Preventing Information Leakage from encoded Data in Lattice Based Cryptography

**Abstract**

With the advent of quantum computers in today's world has led us to improvise our cryptography, we need to improvise the existing cryptography so as to secure the data. In this research paper they have discussed about the security Dimension of lattice-based cryptography including the hardness of lattice problems based on Goldreich-Goldwasser-Halevi (GGH) public-key cryptosystem and its weaknesses. This is widely used in encoding and decoding of data. There is a significant flaw in GGH cryptosystem so in this paper they have enhanced the GGH cryptosystem which prevents information leak.

**Introduction**

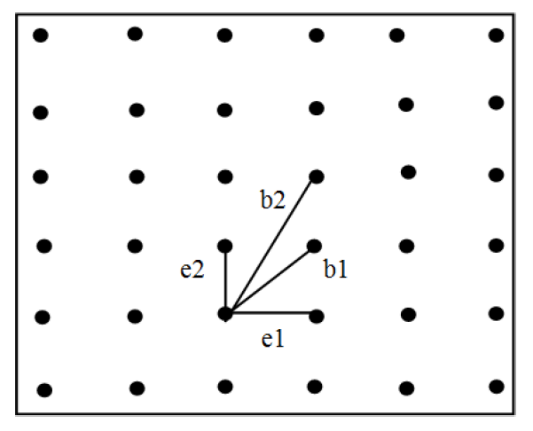
Due to the rise in cryptanalytic attacks and new technology such as quantum computer conventional cryptographic schemes are obsolete.  So, we need a alternate security mechanism which remain secure in these times. lattice-based cryptography is one of the good post quantum cryptographies. Two Lattice-based public key cryptosystem have been accepted and widely used

1. Ajtai-Dwork cryptosystem (AD) and 2. the Goldreich-Goldwasser-

Halevi cryptosystem (GGH). In a scheme the public key is Hermite normal form (HNF) of the Private key is smaller than the public key size of the GGH cryptosystem. In Phong Nguyen’s research paper, he showed that the ciphertext of GGH leaks information about plain text and is a major flaw to be fixed. In this paper they have suggested some methods by which this problem can be fixed. Introductory part follows by an overview on lattices, GGH cryptosystem, Its weakness that observation and proposed methodology.

**Lattice theory**

That is based cryptography is a new approach introduction of data and is widely used in Cloud data security. Miklos Ajtai and Micciancio made the use of lattices in cryptography. A lattice could be a set of points within the n dimensional Euclidean space with a robust property of cyclicity.



Here the set of vectors b1, b2

and e1, e2 are called a basis for the lattice. Here in below

figure e1, e2 forms a “good basis” with nearly orthogonal

vectors and b1, b2 a “bad basis”.

The two problems, shortest vector problem (SVP) and closest vector problem (CVP) are lattice-based problems. CVP states that given a basis of a lattice and a vector c find a vector v ∈ L that is closest to c among all points of L. The Goldreich–Goldwasser–Halevi (GGH) cryptosystem is

based on basis of the fact of a hard lattice problem to find the closest vector in lattice.

**GGH CRYPTOSYSTEM**

The Goldreich–Goldwasser–Halevi (GGH) cryptosystem is an equated asymmetric public-key cryptosystem based on lattices cryptosystem makes use of the nearest vector problem is with a “bad basis”. It’s simple to have a vector that is near to a lattice point, as an instance taking a lattice point and adding a short low error vector. But to get back from this incorrect vector to

the initial lattice point has a crux and a special basis is required.

**Algorithm**

The cryptosystem is based on two variables: dimension n and security parameter. Given a target point and a lattice with two bases, the Private key and the public key are two different types of keys. The Private Key is a secret key that only you have access to. Bgood (roughly orthogonal basis) and Bbad (public key) is a bad basis that is far away from being orthogonal. For the purposes of the Babai's rounding off algorithm, the good foundation, the target point Bgood is capable of locating the closest match. Bbad is unable to solve a lattice point with a high probability. In the lattice, CVP. The data will be sent in this manner. Successfully while maintaining security.

**Security**

To begin with, if a third party listens to the ciphertext c, the only basis available to him is the Public-key B, which is a hardly reliable source of information. The lattice L has an orthogonal basis. However, little is known about it.to solve CVP in a polynomial time algorithm, or to Using a bad basis, we can approximate it to within a polynomial factor. As a result, the decrypted plaintext is inaccurate. It's simple to do. Find the closest vector with a "good basis," however this can be tricky.so with a “bad basis”.

Second, the public basis B has a high orthogonality defect by design, but the private basis R is far from orthogonality. As a result, an eavesdropper would have to solve another hard problem in order to deduce the private key given only the public key. It's easier to create a "bad basis" from a "good basis" than it is to create a "good" from a "bad" one.

Phong Nguyen demonstrated in his study that the GGH scheme's design has several fundamental problems. His "leaky remainder" attack took advantage of the error vector's unique shape to retrieve plaintext remainders: c + mB-1 (mod 2) This streamlined the CVP instance, allowing for easier plaintext recovery and ciphertext testing for varied plaintexts. Aside from attacks like the Round-off Attack, the Nearest-plane Attack, and the embedding Attack, Nguyen uncovered the following flaws in his research:

* Short error vector

Because GGH error vectors are much shorter than lattice vectors, GGH CVP instances are much easier to implement than ordinary Closest Vector instances.

* Error vector in a unique form

The GGH cryptosystem chooses between + and - for the error vector e, where is the security parameter.

As a result, the original is particularly vulnerable to a wide range of attacks.

This σ = (σ, σ,…,σ) ϵ Zn will reveal partial data regarding the plaintext.

* Attack of the "Leaky Remainder"

A well-chosen modulus could be used to remove the error vector from the ciphertext. By adding σ to every element of the ciphertext, the error varies from {-σ, σ} n to {0, 2σ} n. By decreasing modulo 2 *σ*, the error vector can be fully erased.

**PROPOSED METHODOLOGY**

Here they have introduced two functions - The *pixelPermutation()* function is

included to the GGH encryption algorithm and *reconstruct()*

function is appended to GGH decryption algorithm.

The new ciphertext generated by pixelPermutation() when leaked to third party then it will not reveal any information about the plaintext because pixels of ciphertext c are shuffled using random shuffling in MATLAB. Cout.B-1 now returns no plaintext information. In the following lines, we'll go through how both functions work.

* pixelPermutation(): This method takes a ciphertext C produced by the GGH encryption algorithm as an input and returns a new ciphertext C out  with jumbled pixel values.
* reconstruct(): From C out  , it reconstructs the ciphertext C (here symbolized as C rec).

**Conclusion**

In our work, we implemented the GGH cryptosystem with the help of several advanced functions that prevent information leakage. Because of the increased efficiency, a larger range of security parameters can be used while keeping the scheme reasonably feasible. This new technique is demonstrated for grayscale photos, but work on text and colourful images can really be done in the future. Also, because the size of the public key and its matching ciphertext is substantially greater, the system can be improved in terms of space complexity.