Lattice Cryptography based Geo-encrypted Contact Tracing for Infection Detection

Mayank Dhiman, Nitin Gupta, *Senior Member IEEE*, Ujjawal Gupta and Yashwant Kumar

Department of Computer Science and Engineering,

National Institute of Technology Hamirpur

Himachal Pradesh, India

Email: {mdhiman536, nitin3041, gujjawal29, yashwant2451}@gmail.com

Abstract—The world has already witnessed many epidemic diseases in the past years, like H1N1, SARS, and Ebola etc. Now, Covid-19 has also been added to list, which is declared as pandemic by World Health Organization. One of the most commonly used method to tackle the spread of such diseases is using mobile applications to perform contact tracing of the infected person. However, contact tracing applications involve transmitting sensitive location based data of the infected person to the government servers. Therefore, recently this has raised a lot of concerns regarding privacy of the infected persons. This work proposes a light-weight and secure encryption scheme, based on location based encryption which can be used to transfer the location data to the server without compromising its security. The main aim of the work is design an algorithm in such a way that the encrypted transferred data can only be decrypted at the server and in-between data leakage can be prevented. This work proposes to use location based encryption combined with Learning with Errors problems in Lattices, which can provide a solution to privacy concerns in contact tracing, which will even be applicable in the post quantum period.

Index Terms—COVID-19, Lattice based Cryptography, Contact Tracing, Location Data, Security and Privacy.

I. INTRODUCTION

In recent past, world has observed many infectious diseases. However, the Coronavirus disease (COVID-19), seems to be most fatal among all. COVID-19 is an infectious disease caused by SARS-CoV-2, was first believed to originated from Wuhan, China in December 2019 [1], [2]. The World Health Organization also declared this outbreak a pandemic on March 11, 2020. More than 28.4 million cases of COVID-19 have been reported in more than 188 countries and territories, resulting in more than 9,15,000 deaths, while approximately 20.4 million people have also been recovered [3].

Like other contagious diseases, it is believed that the COVID-19 infection spreads when a healthy person comes in close contact with the already COVID-19 infected person with symptoms [4]. Although work to develop a potential vaccine has already begun, however, till now major breakthrough in this area has not been achieved. Till any vaccine arrives which can be used to cure COVID-19, one of the possible preventive solution is to track and isolate the infected person. Further, all the persons which are primary contact of that person are also required to be identified [5], [6]. This is not a new technique, humans have already used manual contact tracing during previous pandemics, notably during Spanish Flu in

1918 [7]. However, with large population of the globe, it is not feasible to do contact tracing manually.

To achieve this objective of contact tracing, mostly all the countries have developed contact tracing apps for a smart phone. These apps are used for tracing and identifying the COVID-19 effected person [8]. Government of India has also developed 'Aarogya Setu' application [9], which reached 50 million users in just 13 days. Government of Singapore, uses 'TraceTogether' application [10], which relies on Bluetooth technology, for doing contact tracing. Honk-Kong Government has developed 'StayHomeSafe' [11] application, which pairs with wristbands to perform contact tracing. Other countries have also developed their indigenous contact tracing applications, many have mandated its citizens to install the application in their smart phone.

Although, contact tracing has proved to be effective in tackling COVID-19, however, many privacy advocates and security researchers have raised their concern about the privacy problems associated with these applications. Most contact tracing applications in use today, involve collecting location data from the user, and sending it to the government servers. Since sensitive location data is transmitted, one of the major concern is data leakage on the way to the government servers. Due to the data security concerns, many persons are not willing to install the contact tracing app to share their private data [12]. Therefore, a secure and privacy preserving software application is highly required to build the confidence of a contact tracing application user.

The main aim of this work is to secure contact tracing application. All contact tracing applications mostly have similar underlying technology [13]. As mentioned above, mostly all the contact tracing applications send the user's private data to the remote server and prone to the inbetween data leakage. The objective of this work is to ensure that data obtained from the contact tracing application can only be decrypted at the specified server. To achieve this objective, the proposed work introduces *Learning with Errors* (LWE) problem of Lattices with location based encryption scheme to prevent data leakage in contact tracing applications. Moreover, the proposed lattice based encryption, is a strong candidate for post quantum cryptography [14]. Therefore, the proposed scheme will be a promising approach even in the post quantum period. Further, a geo-encryption scheme is implemented by incorporating the

location parameters in the key generation process of standard LWE based encryption. It is shown through analysis that the proposed encryption algorithm is light-weighted and fast enough to be effectively used in contact tracing applications.

The rest of the paper is organized as follows. In Section II, previous works which have been done in this field are discussed. In Section III, brief background about LWE problem and geocryption is provided. In Section IV, proposed approach with LWE based cryptography system is discussed followed by describing the various ways in which location parameters in the key generation process can be incorporated. Further, various methods of generating key are also discussed. Finally, in Section V, performance of the proposed scheme is evaluated followed by the conclusions.

II. RELATED WORK

Contact tracing applications typically involve sending the location based data of the infected person to a server, so that the data can be analyzed and the people who have been in contact with him or her can be notified about the potential risk. In recent times, there have been some works to analyze what kind of privacy concerns may arise and how they could be tackled. Authors in [15] discussed various types of contact tracing applications and their privacy concerns. The authors have highlighted data leakage problem in this work also. James Bell et al. in [16], analyzed the security concerns about 'TraceTogether' application. The authors analyzed the use of Additive Homomorphic encryption as an effective measure to secure contact tracing. Ni Trieu et al. [17], considered the application where random tokens are exchanged between users' smart phones when they come within proximity of each other. These tokens then, get stored locally on users' phones. These tokens are used to learn if any user has recently been in the contact of an infected person. The authors proposed a two-party private set intersection cardinality based algorithm where private information is not exchanged between phones. Authors in [18], discussed an approach to secure contact tracing, by using Multi-Party Computation. This approach allows a group of participants to evaluate a particular function. An individual can only learn final result and could not see private input of the users. This approach is however has a larger run time. Thamer Altuwaiyan et. al. in [19], attempted to secure the information of the user using matching techniques over the encrypted content, with enhancements using the weight-based-matrix. Authors in [20], [21] proposed to collect and stores users' data based on proximity-based protocols, which enables two users to match their profile without disclosing any personal information.

Although all of these methods provide secure lightweight encryption, we believe these methods will cease to be productive after the post quantum period. That is, once the Quantum Computers take over in the near future, these cryptographic schemes will not prove to be a valid solution.

Contribution of the work: To the best of knowledge, the proposed lattice based Geocryption scheme has not been

used till now for securing data in the contact tracing applications. The proposed lattice-cryptosystem based scheme can prove to be an effective answer even in the post quantum period. The main objective of the work is to design a decryption algorithm which ensures that data is decryptable only at the specific location. Furthermore, this work also proposes a few methods to make the proposed scheme light-weighted which may be used to increase the speed of proposed algorithm, and so that can be effectively used in these applications.

III. BACKGROUND

Before proceeding to explanation of the proposed lattice based algorithm for contact tracing, some key techniques used in the proposed scheme are discussed in this section.

LWE problem introduced by Oded Regev is a versatile basis for cryptographic constructions [22], and the cryptographic constructions based on it are claimed to be secure. The LWE problem asks to recover a secret $s \in \mathbb{Z}_n^q$ given a sequence of 'approximate' random linear equations on s , and each equation is correct up to some small additive error. Recovering s from these equations would have been quite easy with Gaussian Elimination Algorithm [23] if there was no additive error, however introduction of errors make this problem significantly difficult. Further, the best known algorithms for the lattice problems require $O(2^n)$ time [24], [25]. Since, there is no polynomial time quantum algorithm, even quantum computers won't be able to solve this problem. This hardness of LWE problem set basis for many cryptographic constructions. However, LWE problem can be reduced to many easier problems. This flexibility of creating variants of LWE problem is one of the reason for it's large number of applications in cryptography. In addition to this, LWE can be implemented efficiently as it involves low complexity operations (mainly additions).

Next, the main component of the proposed work is *Geocryption*. Logan Scott and Dorothy E. Denning in [18] have discussed the general mechanism of Geocryption. Location-based-Encryption is referred to a method of encryption using which an encrypted text is decryptable at the specific location only. Usually, standard encryption algorithms like AES or RES have been used a lot in location based encryption. Location-based-encryption typically involves, taking a standard cryptography algorithm and *trying* to incorporate the location parameter in it's key. The receiver using its location data to generate a secret key. The decryption is possible only if the "key generated by the receiver" is same as "key sent by the sender". Hence adding an *extra layer of security* in the algorithm. One of the application of location based encryption is the effective movie distribution, where the movie is available at only those theaters, which have actually paid for the movie. Further, the main part of the location-based-encryption algorithm is the key generation process where the location is incorporated in the key, however, it should be difficult to retrieve the location back from the key.

Let us understand LWE problem with an example. There is a secret vector $S = (s_1, s_2, s_3, s_4)^T \in \mathbb{Z}_{13}^4$. Following equations

are correct up to some small additive error.

$$\begin{aligned}
5s_1 + 5s_2 + (-3)S_3 + 7s_4 &\approx 6(\text{mod}13) \\
(-1)s_1 + 1s_2 + 2S_3 + (-5)s_4 &\approx -4(\text{mod}13) \\
(-3)s_1 + 3s_2 + 7S_3 + 4s_4 &\approx 2(\text{mod}13) \\
5s_1 + 4s_2 + (-1)S_3 + 2s_4 &\approx -5(\text{mod}13) \\
(-4)s_1 + 6s_2 + 3S_3 + (-2)s_4 &\approx 5(\text{mod}13) \\
(-2)s_1 + 3s_2 + 1S_3 + 6s_4 &\approx -3(\text{mod}13)
\end{aligned}$$

Using these equations, secret vector S is to be solved. In Fig. 1, Challenger is having the secret S and generates LWE samples (A, b) from the LWE distribution $A_{s,n,q,\chi}$. For secret vector $S \in Z_q^n$ and distribution χ , LWE distribution $A_{s,n,q,\chi}$ generates samples $(a, b) \in Z_q^n \times Z_q$ where a sampled uniformly from Z_q^n and $b = \langle a, s \rangle + e$ where $e \leftarrow \chi$. Challenger passes LWE samples (A, b) to the adversary and then the adversary tries to find secret $S \in Z_q^n$.

Fig. 1. Learning with errors problem (search)

Next, brief introduction of LWE based encryption and decryption is discussed [22]. For an encryption using LWE, one private key (secret key S), and two public keys (A, B) are required, where a private key is used for decryption of the data and public keys are used for encryption of the data. The various parameters in LWE crypto-system are as follows:

- 1) Security parameter - n
- 2) Number of equations - m
- 3) Modulus - q
- 4) Noise parameter - α (real number)

For having both security and correctness, q needs to be a prime number between n^2 and $2n^2$, $m = 1.1 \cdot n \log q$, and $\alpha = \frac{1}{\sqrt{n} \log_2 n}$.

Key generation, encryption & Decryption in general Learning with Errors based Cryptosystem is described as (all additions are performed modulo q)

- **Private Key:** It is vector s chosen uniformly from Z_q^n .
- **Public Key:** The public key consists of m samples $(a_i, b_i)_{i=1}^m$ from the LWE distribution with secret s , modulus q , and error parameter α .
- **Encryption:** For each bit of the message, select a random set S uniformly among all 2^m subsets of $[m]$. The

Fig. 2. Overview of Contact-Tracing Apps

encryption is $(\sum_{i \in S} a_i, \sum_{i \in S} b_i)$, if the bit is 0 and $(\sum_{i \in S} a_i, \lceil \frac{q}{2} \rceil + \sum_{i \in S} b_i)$, if the bit is 1.

- **Decryption:** The decryption of a pair (a, b) is 0 if $b - \langle a, s \rangle$ is closer to 0 than to $\lceil \frac{q}{2} \rceil$ modulo q , and 1 otherwise.

IV. PROPOSED LATTICE CRYPTOSYSTEM

Let us consider an user *Alice* comes in contact with an infected person *Bob*. A contact tracing application sends *Alice*'s information to a server, say *Servy*. Based on the contact tracing application, the *Servy* then notifies *Alice* about the contact. However, this data may be attacked by a middle-man or hacker, say *Trudy*. The example scenario is shown in Fig. 2. The main objective of the proposed work is to make the location data only available to *Servy*. The data should not be visible to any person who is not in the same *location* as of *Servy*. The proposed work considers a LWE based scheme to encrypt the data sent by the *Alice* to the *Servy*. The main idea is that the data is only encrypted using a key derived from the location data of *Servy*, thus it can only be decryptable at the *Servy* itself.

The main components of the proposed scheme, the key generation, encryption and decryption are discussed next.

A. Secret Key Generation using Location Parameters

As mentioned above, the main objective of the work is that the encrypted data can only be decrypted at a particular location (at server). The considered location (or encryption) parameters in the proposed scheme are latitude, longitude and tolerance Distance of the server. The tolerance distance represents the radius around a particular value of latitude and longitude in which the data can be decrypted. Further, while discussing standard LWE based encryption, a secret key S is required. Depending on the number of considered parameters, S is a vector of dimensions $1 \times s$. To show the complexity

involved in different dimensions, four cases are considered where s can be either 1, 2, 3 or in general n .

- 1) $[1 \times 1]$: For different locations, different keys must be generated to make system more secure. Therefore, some hash function can be used to generate secret key using latitude, longitude and tolerance distance.

$$S = (u \times P_1 + v) \times P_2 + t, \quad (1)$$

where u is latitude, v is longitude and t is a tolerance distance. Further, P_1 and P_2 are selected as some big prime numbers such that chances of collision are minimized.

- 2) $[1 \times 2]$: Consider three constants or weights a , b and c , which are multiplied with u , v and t respectively to get the normalized results. For, the first value of S , S_1 is defined as,

$$S_1 = \frac{au}{au + bv}, \quad (2)$$

and S_2 is defined as,

$$S_2 = \frac{bv}{au + bv}. \quad (3)$$

- 3) $[1 \times 3]$: Here the above process is extended by including tolerance to generate three values. For, the first value of S , S_1 is defined as,

$$S_1 = \frac{au}{au + bv + ct}, \quad (4)$$

S_2 is defined as,

$$S_2 = \frac{bv}{au + bv + ct}, \quad (5)$$

similarly, S_3 is defined as,

$$S_3 = \frac{ct}{au + bv + ct} \quad (6)$$

This process, can be combined with standard key generation algorithms like Password-Based Key Derivation Function 2 (PBKDF2) [26].

- 4) $[1 \times n]$ or *Generalized Secret Key Generation*: To generate a secret key matrix of size $1 \times n$, n random values from the Normal Distribution are selected. Mean of this normal distribution function is weighted average of latitude, longitude and tolerance distance. Further, standard deviation will be a constant. Generation of secret key is summarized in Algorithm 1.

B. Generating public key

After the secret key $S_{n \times 1}$ of size $n \times 1$ is generated, public key $A_{m \times n}$ of size $m \times n$ needs to be generated. In the proposed scheme, public key $A_{m \times n}$ is created by generating a random matrix of $m \times n$. Then, public key $A_{m \times n}$ is multiplied with secret key $S_{n \times 1}$ and the result is added with error $e_{m \times 1}$ to get public key $B_{m \times 1}$. The error to be added is a matrix of size $m \times 1$ which follows distribution χ .

$$B = (A \times S + e) \% q \quad (7)$$

Algorithm 1 GenerateSecretKey(u, v, t, n)

Assumptions:

- *Random Class* is a Class for generating random variables.
- *Random class* also contains functions for generating numbers using some probability distributions like Normal Distribution.
- A seed value can be set for an instance of a random class such that same pseudo-random numbers may be generated again, if required in future.
- Implementation of this class may vary in different programming languages.

string u : Latitude of the location

string v : Longitude of the location

string t : Tolerance of the location

int n : Length of the secret key vector

1: Begin

2: Initialize r with some random value

3: Initialize Standard Deviation **S.D** for normal distribution with a constant value

4: Calculate **mean** as weighted average of latitude, longitude and tolerance:

$$mean = \frac{au}{au + bv + ct},$$

where a, b, c are weights

5: Get an Instance of Random class: **rand** = **new Random()**

6: Set seed of instance as r so that same random numbers can be generated again if required in future: **rand.seed(r)**

7: Call Normal Distribution of Random class with parameters mean, S.D, n : **S** = **rand.normal(mean, S.D, n)**

8: **Output**: Secret Key, S (a vector of size $n \times 1$)

C. Encryption

Data that needs to be transmitted is first converted into a binary format. Then binary data is encrypted using different keys. Two public keys A and B are generated, where A and B are matrices. Further, a value q is selected which is a prime number used for modulus operation. Secret key S (a private key), is generated using Geo-encryption that considers three parameters latitude, longitude and tolerance. Then using A , B and q each bit is converted to (u, v) pairs, and hence, encryption is completed.

The public key consisting of m samples $(a_i, b_i)_{i=1}^m$, has 2^m possible subsets. For every bit of the message, a random set F is selected uniformly among all the 2^m subsets. For generating (u, v) pair for every bit, following LWE scheme as discussed in [22] is followed (in which all additions are performed modulo q):

- 1) *When bit value is 0*: For every (a_i, b_i) pair in the set F , summation of all a_i is done to generate u and summation

of all b_i is done to generate v .

$$(u, v) = \left(\sum_{i \in F} a_i, \sum_{i \in F} b_i \right). \quad (8)$$

- 2) When bit value is 1: For every (a_i, b_i) pair in the set F , summation of all a_i is done to generate u . For generating v , summation of all b_i is added with $\left\lceil \frac{q}{2} \right\rceil$, where q is a modulus number.

$$\left(\sum_{i \in F} a_i, \left\lceil \frac{q}{2} \right\rceil + \sum_{i \in F} b_i \right). \quad (9)$$

After each bit is converted to (u, v) pair, encryption is completed and this encrypted data is then, ready to be transmitted from *Alice* to the *Servy*. Further, the seed value used for secret key generation, modulo number q and location tolerance value are also required to be send to the *Servy* along with the encrypted data. The proposed encryption process is summarized in Algorithm 2.

Algorithm 2 Encryption()

- 1: Begin
 - 2: Generate secret key S using location parameters.
 - 3: Generate public key A using Random sampling.
 - 4: Select modulo number q .
 - 5: Generate public key B , as given in (7).
 - 6: Convert data to binary format.
 - 7: Convert each bit of binary data to (u, v) pair using encryption scheme.
 - 8: **OUTPUT:** Encrypted Data.
-

D. Decryption

Alice (sender) sends data to the *Servy*(receiver) in the encrypted form. For decryption to happen on the *Servy*'s side, *Servy* also receives some parameters mentioned above, through the secure channel. One trivial mathematical operation which can be used to make data decryptable only at *Servy*'s location, is using XOR operation and shown in Fig. 3. In the proposed

Fig. 3. Standard Location Based Encryption Using XOR gate

approach, a decryption scheme discussed below, is followed for converting each (u, v) pair in the encrypted data to its binary form. After conversion of whole encrypted data into the binary format, conversion is done to retrieve the original

text/data. The decryption scheme used to check if the data can be decrypted at location (u, v) is as follows:

- 1) Location of *Servy* is fetched using a location device. Let's consider this location value to be (x, y) , where x is the latitude and y is the longitude.
- 2) The location of the *Servy* (x, y) , is then added with location tolerance value (t) (received from *Alice*) to generate say P different samples of the locations such that $(x, y) + t = (x_i, y_i)_{i=1}^P$.
- 3) For every location $(x_i, y_i)_{i=1}^P$, $[(x_i, y_i), t]$ pair is used to generate a secret key S' using the same scheme employed on the *Alice*'s side.
- 4) The following scheme is used to generate a secret key S' using (x_i, y_i) [22]:
 - For secret key S' and pair (u, v) in the encrypted data, calculate $v - \langle u, S' \rangle$
 - If $v - \langle u, S' \rangle$ is closer to 0 than to $\left\lceil \frac{q}{2} \right\rceil$ modulo q , the pair (u, v) is decrypted as 0.
 - If $v - \langle u, S' \rangle$ is closer to $\left\lceil \frac{q}{2} \right\rceil$ modulo q , the pair (u, v) is decrypted as 1.
- 5) This secret key S' is used for decryption of the encrypted data. If decryption fails for S' then, other secret keys generated using $(x_i, y_i)_{i=1}^P$ is used. If *Servy* is in correct desired position, data would get decryptable for some S' generated using (x_i, y_i) . If data doesn't get decrypted for any of the generated secret keys, that means the location of the *Servy* is invalid i.e *Servy* is not at correct desired location for decryption of the data to take place.

Summary of proposed decryption algorithm is shown in algorithm 3.

Algorithm 3 Decryption()

- 1: Begin
 - 2: Fetch *Servy*'s location (x, y) .
 - 3: Set $(x, y) + t = (x_i, y_i)_{i=1}^P$.
 - 4: Set $i = 1$.
 - 5: **for** $i \leq P$ **do**
 - 6: $S' = (x_i, y_i) + t$.
 - 7: Try Decryption using S' .
 - 8: **if** Success **then**
 - 9: **BREAK**
 - 10: **end if**
 - 11: **end for**
 - 12: **if** Still Decryption fails **then**
 - 13: Invalid Location of *Servy*.
 - 14: **else**
 - 15: Convert each (u, v) pair in encrypted data into binary format.
 - 16: Convert to Original data.
 - 17: **end if**
 - 18: **OUTPUT:** Encrypted Data.
-

V. PERFORMANCE ANALYSIS

In this section, analysis of the proposed algorithm on various parameters is done.

A. Time Complexity Analysis

The performance speed of the algorithm relies on two major parts, first is the key generation part, and second is matrix or vector creation, which is further used for subsequent calculations with other matrices A and B. Both of these scenarios are described separately below, and the final complexities are inferred.

1) *Key Generation Phase*: In Section IV-A, various ways of generating key for the proposed algorithm like, generating a $[1 \times 1]$, $[1 \times 2]$ or $[1 \times N]$ key have been discussed. Overall, the complexity varies as the N varies. The method, of generating keys for $N \leq 3$ require $O(1)$ time, however, as the value of N increases so does the time complexity of key generation algorithms. In the method, for $N > 3$, taking the help of a normal distribution, generating key for specific values could take up to $O(N)$.

2) *Calculations after key generation*: Once the key is generated, matrix calculations at encryption and decryption stages are required to be performed. It is observed that this T varies with a cubic factor as N increases, or as $T \propto N^3$. The reason of dependency is that as the N increases, so does the dimensions of the corresponding matrices. The reason being that the standard matrix generation algorithms take a time and space complexity of $O(N^3)$.

B. Security Analysis

1) *Security on the basis of structure of lattice*: In lattice cryptography, considered lattice is a regularly spaced grid of points or vectors, and also can be extended to infinity. However, since there is a memory limit, *lattice basis* [27] (collection of vectors that can be used to reproduce any point in grid that forms lattice) is used in the proposed work to define lattice, which produces the final grid. Moreover, to increase the security of the proposed algorithm the dimensions of the *lattice basis* can be increased as much as possible to enhance security. However, increasing too much dimensions will also increase the required number of mathematical operations, which will finally increase the complexity. Therefore, there is a trade-off between dimension and security and one has to make the balance.

2) *On basis of secret key*: For generating a secret key, location parameters are used in the proposed work, which includes latitude, longitude and tolerance discussed in Section IV-A. Further, location is converted into a binary format and then PBKDF2 is applied, which is a simple cryptographic key derivation function. This function takes parameters as binary data, salt or seed, and counts. Salt is randomly generated bytes which is generally 16, 32, 64 or 128 bytes long. The proposed work considers 32 bytes salt, and this size can be further increased to enhance the security as total number of possible combinations will also be increased. However, it will also increase the computation time slightly. Further, count

parameter which denotes number of iterations, to generate key can also be manipulated accordingly.

3) *Compromise between speed and security*:: One of the objective of the algorithm, is to provide a good and efficient mechanism for contact tracing, which is used in mobile applications. Therefore, it is of utmost importance that the algorithm should be light weight, at the same time since location data is involved security should not be compromised. With some experimentation, it is found that the equilibrium point for both to work efficiently is, by using a key of size $N \in [3, 10]$. It can be deduced that the key is generated for $N \leq 5$ using various $O(1)$ methods. However, as the size for N increases it is better to use some uniform probability distributions, like Normal distribution, as discussed in previous section.

C. Possible Optimization

For the decryption process, two possible cases can be considered. One without considering the tolerance distance and the other with the tolerance distance. In the former, the region or space in 3D, where the decryption is possible is determined by the accuracy of the GPS. However, this is not same in every scenario, as one GPS may vary from other.

In the other case where tolerance distance is considered, each point in the hypothetical sphere or circle (if only two dimensions are considered) formed by the tolerance distance, is searched. Further, certain optimization can be made here. Instead of dividing the circle, into fixed size squares, and searching in it; a lot of computational time may be saved, if the size of squares is less, near the centre and increases as we move outwards from the centre. Thus, there will be more density of search points at the centre and this will decrease as we move out.

Another factor that can decide the density of the squares, is the accuracy of the receiver. If the considered GPS, has low accuracy, it is better to search more rigorously in a larger part, otherwise it is sufficient to search in the areas closer to the center.

VI. CONCLUSIONS

In the Covid-19 pandemic, one of the effective way to tackle the infection is to identify the infected person and to alert all the nearby persons who could possible come in the contact of that person. Large number of contact tracing applications are available worldwide to meet this objective. In this work, a secure cryptographic algorithm for contact tracing applications has been discussed. The proposed method can effectively be used even in the post quantum period. The proposed scheme can be effectively applied to all the contact tracing applications where private data of the user is sent to the server for contact tracing of an infected person. The analysis shows that the proposed scheme is light weighted and effective in preserving privacy as data of the user can only be decrypted at the server's location and hence data leakage in between can be well avoided. For the future work, mobile server can be considered, where location data is directly transferred between two mobile

Fig. 4. Example of dynamic search space. The square size is low and thus density high inside the inner circle. The inner circle radius can be varied

devices. Hence location data and coordinates need to be fetched in real time. Further, multiple server scenario may be considered, data fragments are sent to different servers for parallel processing. This would require an altogether different mathematics for key generation as the keys have to be different for different servers and yet should contain some information about the fragment being sent to that server. Although, there are numerous schemes which entirely skip the server, and thus are more secure. Yet we developed the algorithm in this form so that it can secure those applications (like Indian application, Aarogya Setu) which already involve sending data to the server, and entirely skipping server is not a viable option. The main effort of this work is to help mankind fight the pandemic, which had already taken many lives, and is not showing any signs of decrease in numerous regions.

REFERENCES

- [1] Z. Y. Zu, M. D. Jiang, P. P. Xu, W. Chen, Q. Q. Ni, G. M. Lu, and L. J. Zhang, "Coronavirus disease 2019 (covid-19): a perspective from china," *Radiology*, p. 200490, 2020.
- [2] A. Deslandes, V. Berti, Y. Tandjaoui-Lambotte, C. Alloui, E. Carbonnelle, J. Zahar, S. Brichler, and Y. Cohen, "Sars-cov-2 was already spreading in france in late december 2019," *International Journal of Antimicrobial Agents*, p. 106006, 2020.
- [3] A. J. H. University., "Covid-19 dashboard by the center for systems science and engineering (csse) at johns hopkins university (jhu)," 2020 (accessed June 5, 2020).
- [4] Centre For Disease Control and Prevention, "Symptoms of coronavirus," 2020 (accessed June 5, 2020).
- [5] M. A. Azad, J. Arshad, A. Akmal, S. Abdullah, F. Ahmad, M. Imran, and F. Riaz, "A first look at contact tracing apps," *arXiv preprint arXiv:2006.13354*, 2020.
- [6] P. Ish, S. Agrawal, A. D. Goel, and N. Gupta, "covid19 contact tracing: Unearthing key epidemiological features of covid-19," *SAGE Open Medical Case Reports*, vol. 8, p. 2050313X20933483, 2020.
- [7] "Contact tracing at the times of spanish flu," <https://csahq.org/news/blog/detail/csa-online-first/2020/04/06/history-and-covid-19-part-i-lessons-from-1918>.
- [8] P. C. Ng, P. Spachos, and K. Plataniotis, "Covid-19 and your smartphone: Ble-based smart contact tracing," *arXiv preprint arXiv:2005.13754*, 2020.
- [9] "Aarogya setu by indian government," <https://www.mygov.in/aarogya-setu-app/>.
- [10] "Trace together by singapore government," <https://www.tracetoegether.gov.sg/>.
- [11] "Stayhomesafe application by honk kong government," <https://www.coronavirus.gov.hk/eng/stay-home-safe.html>.
- [12] L. Horvath, S. Banducci, and O. James, "Citizens' attitudes to contact tracing apps," *Journal of Experimental Political Science*, pp. 1–27, 2020.
- [13] F. Legendre, M. Humbert, A. Mermoud, and V. Lenders, "Contact tracing: An overview of technologies and cyber risks," *arXiv preprint arXiv:2007.02806*, 2020.
- [14] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 147–191.
- [15] H. Cho, D. Ippolito, and Y. W. Yu, "Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs," *arXiv preprint arXiv:2003.11511*, 2020.
- [16] J. Bell, D. Butler, C. Hicks, and J. Crowcroft, "Tracesecure: Towards privacy preserving contact tracing," *arXiv preprint arXiv:2004.04059*, 2020.
- [17] N. Trieu, K. Shehata, P. Saxena, R. Shokri, and D. Song, "Epi-one: Lightweight contact tracing with strong privacy," *arXiv preprint arXiv:2004.13293*, 2020.
- [18] L. Scott and D. E. Denning, "A location based encryption technique and some of its applications," in *ION National Technical Meeting*, vol. 2003, 2003, pp. 730–740.
- [19] T. Altuwaiyan, M. Hadian, and X. Liang, "Epic: efficient privacy-preserving contact tracing for infection detection," in *IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.
- [20] R. Zhang, J. Zhang, Y. Zhang, J. Sun, and G. Yan, "Privacy-preserving profile matching for proximity-based mobile social networking," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 656–668, 2013.
- [21] M. Li, S. Yu, N. Cao, and W. Lou, "Privacy-preserving distributed profile matching in proximity-based mobile social networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 5, pp. 2024–2033, 2013.
- [22] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, pp. 1–40, 2009.
- [23] J. F. Grcar, "Mathematicians of gaussian elimination," *Notices of the AMS*, vol. 58, no. 6, pp. 782–792, 2011.
- [24] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001, pp. 601–610.
- [25] D. Micciancio and P. Voulgaris, "A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations," *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1364–1391, 2013.
- [26] S. Josefsson, "Pkcs# 5: Password-based key derivation function 2 (pbkdf2) test vectors," *Internet Engineering Task Force (IETF), RFC Editor, RFC*, vol. 6070, 2011.
- [27] C. Peikert, "A decade of lattice cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.