FishEye Matlock: A Random Functional Encoding Mechanism for Secure Location Sharing

Pedro Wightman, *Senior Member, IEEE*, Nicolás Avilán, and Augusto Salazar

***Abstract*—Location tracking is difficult to protect due to the sequential nature of the data and the need for accuracy to offer a proper service and monetize the location information. Homomorphic encryption can partially solve the problem, but it typically has a minimal set of operations that are not feasible for geographical operations. Randomized Functional Encoding (RFE) allows the creation of random keys to decrypt data, according to the user's needs. Now, creating keys that only focus on a portion of the path while protecting the rest of the path has not been proposed in the literature, to the knowledge of the authors. This work proposes a RFE mechanism for Matlock-coded location data, called FishEye Matlock. This technique generates disposable random key matrices that only reveal a desired portion of the path, with the possibility for the user to add random noise to protect the revealed data and to control the amount of noise added to the rest of the path. This allows secure information sharing with particular actors, like law enforcement, so that the information of interest is shared without affecting the user's privacy. The algorithm is tested in two different path scenarios to show the technique's applicability, the level of protection, and the impact of the parameter value selection. Results show that the mechanism can be tailored to generate key matrices for different scenarios: at the lowest value of k, the level of noise reaches several thousands of kilometers of noise along the path, and between 60 and 100 times the level of noise, and with, the highest k value, between 40% to 80% of the maximum distance radius on average at the point of interest, and between 1.1 and 10 times the defined noise level at the path.**

Link to graphical and video abstracts, and to code:
https://latamt.ieeer9.org/index.php/transactions/article/view/8959

***Index Terms*—Location privacy, Access control, Location obfuscation, Matrix multiplication, Location-Based Services.**

I. Introduction

LOCATION privacy is defined in [1] as "a special type of information privacy which concerns the claim of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others". This definition works towards protecting the geolocated information from users, for them to be able to obtain quality contextualized services while keeping control of their information as much as possible. In addition, given that this information is valuable for many companies that are supporting their core business with it, it is impossible to ignore the importance of preserving the value of the data being protected for monetization.

There are many techniques that can be used to protect location information; however, they have not been widely adopted because of a lack of regulation and interest by service providers. Some techniques, like noise-based location obfuscation, can degrade the quality of the information, which makes it less profitable. Based on that, techniques that preserve the precision of the information are desired, like encryption or location coding, because the original information is secure and can be accessed anytime.

In most cases, a service provider cannot offer any functionalities over secure data without access to the key, which would give total access to the data, and facilitate a data breach. Homomorphic encryption offers some options [2]; however, there are some issues: the types of operations available are limited for complex operations, like the ones needed for geographical analysis; they may introduce errors in the data that may affect the precision; the algorithms need to format the data in a certain way that complicates storage. So, in scenarios where the user needs to share a precise location with a third-party actor, directly from the service provided to certify its authenticity (like when required by law enforcement), the user would put its privacy at risk.

This work presents FishEye Matlock, a functional encoding mechanism for secure location sharing in location-based information systems. It works over Matlock-coded locations and creates disposable key matrices that can decode a small portion of a path without revealing all the rest of the data. In addition, the algorithm allows adding random noise and defines how much information is revealed in terms of the whereabouts of the user. Thus, FishEye Matlock provides a tool that preserves the user's privacy while providing a service that the provider can use to offer functionalities. The technique is tested by applying the technique in two different scenarios, showing how the technique can be used at different parts of the paths, and how the parameters can be used to control the level of noise in the data.

II. Related Work

In the literature, many different techniques exist that can be used to protect location information. One way of doing this is obfuscation, which irreversibly alters each location in different manners before it is sent to the service provider. This method prevents eventual attackers or even the provider from having the exact original reported location of the user.

The associate editor coordinating the review of this manuscript and approving it for publication was Oscar Mauricio Caicedo (*Corresponding author: Pedro Wightman*).

Pedro Wightman, and N. Avilán are with the Universidad del Rosario, Bogota, Colombia (e-mails: pedro.wightman@urosario.edu.co, and nicolasg.avilan@urosario.edu.co).

A. Salazar is with the Universidad del Norte, Barranquilla, Colombia (e-mail: augustosalazar@uninorte.edu.co).

There are different ways to do this: adding random noise using different distributions and techniques [1], [3]–[7], replacing actual locations with alternate points in grids or points of interest in the city [8], aggregating locations based on time and areas [9], [10], etc.

The main drawback of these mechanisms is the loss of accuracy of the data, so the original path cannot be rebuilt for any future applications. In addition, if the level of noise defined by the user is too large, the quality of the service may be affected as well as the value of the data, or, if the amount of noise is too little, then an attacker would be able to infer the original location easily. In [4], the authors find an alternate route between points A and B, based on feasible paths, to obscure the location of a user. This solution solves the problem of noise-inducing techniques, but they are mostly useful in urban areas because there will be real options to generate alternate locations, compared to a highway in a remote area with no nearby roads. Also, these techniques imply a large computational cost for training the systems.

Another option to implement obfuscation is to transform the data by using a mathematical function [4], [11], [12] or homomorphic encryption [13]–[15], to store the location information and offer some level of service, but making it not understandable, and thus unusable by the service provider or any attacker without the key when the data is coded.

This technique is more advantageous than noise-adding techniques in terms of user privacy, because the user keeps control of the data, receives services, and preserves the accuracy of the data. This is especially useful if the data needs to be shared. Now, sometimes a full-homomorphic scheme may be expensive in terms of power and computation in small devices, thus lightweight techniques are still necessary. Also, some homomorphic techniques allow a fairly small number of operations that may not fit the needs of geographical data [2]. For example, in [13], the technique uses k-anonymity and homomorphic encryption to generate a value that can be operated, but it needs the presence of k other users to operate correctly.

In [12], the authors propose three transformation techniques. The Ellipsoidal Random Obfuscation Method uses a mechanism similar to Matlock's in terms of matrix multiplication and ellipsoidal transformation to alter the positions. The other two techniques alter the location based on a chain of circles or by using a grid to improve the selection of feasible points, trying to preserve the distance between the points. This characteristic keeps the path in the same coordinate space, which may allow attackers to know the user's whereabouts. Thus, it is not suitable for the proposed scenario.

In [14], the authors propose using homomorphic encryption to generate an obfuscated path that preserves time and distance between points but does not reveal the original path. Now, the resultant path is near the original path, which still allows an attacker to know the general area where the user is moving. In [15], the authors propose using slight perturbation of the locations and encryption to protect the original location information when used to calculate a common meeting point among several users. In general, most of these techniques reveal the user's whereabouts or enough information to recreate part of the path, which may not be acceptable for many users.

Classic encryption techniques, as well as homomorphic encryption, despite their strength in protecting the data, are not normally suitable for some special scenarios. For example, if a user wants to share a small segment of a coded path with the service provider or a trusted third-party actor, to improve the quality of service, to comply with a judiciary order, in case of a medical emergency, or any other reason to confirm presence at a certain point at a certain time, without having to reveal any information about the rest of the path, then sharing the key will not be secure enough.

An effective privacy protection mechanism should conceal the user's location information while guaranteeing a good level of contextualized service, allow the user to have a high level of control over the data, permit access to trusted and untrusted actors with differentiated levels of trust, and reduce data quality degradation after the protection mechanism has been applied. Functional Encryption (FE) solves this scenario.

In [16], the authors formalize the definition of FE scheme where "an authority holding a master secret key can generate a key sk_k that enables the computation of the function $F(k,)$ on encrypted data. More precisely, using sk_k, the decryptor can compute $F(k, x)$ from an encryption of x." In other words, new keys can be generated to decrypt a specific portion of the cipher.

A Randomized Functional Encryption scheme, as defined in [17], considers that "an encryptor can hide an input within a ciphertext in such a way that authorized decryptors can only recover the result of applying a randomized function to it." This means that the decrypt cyphertext will have an added randomness that protects the shared context.

There have been many applications of these concepts. In [18], an application was developed to support VANETs secure communications, also using a blockchain to preserve the consistency of the data. The authors of [19] used FE to solve the cross-domain problem of distributing classified information among different actors in a shared network. In [20], the authors propose a technique to protect data where order must be preserved, like medical lab results.

In [21], the authors applied FE to secure geographical keyword-based queries via a three-actor-based system: Location-Based Services (LBS) server, LBS user, and Cloud server. The LBS server sends the encrypted data to the cloud and sends secret keys to the users to query the encrypted data. The user will build the query structure using a matrix-based technique and a K value representing the maximum number of relevant query results. In this way, the cloud server does not know the real geographical information or what the user is requesting. This scheme has a few limitations: the user fully trusts the LBS, and there is a need for secure communication between the LBS server and the user to exchange private keys, which the service provider knows.

III. The Matlock Algorithm

The Matlock algorithm [11] is a transformation-based location privacy protection mechanism that uses matrix multiplication to change the geometrical projection and scale of

the locations. This change makes the resultant coordinates useless for geographical purposes, producing values way over the $\pm 90^o$ for latitude and $\pm 180^o$ for longitude coordinates, as shown in Fig. 1. It is a very lightweight mechanism that does not require complex computations while providing good protection. Also, this is a great advantage when working with untrusted service providers because the user controls the data and many potential services over coded data.

The general Matlock-based location-based service works as follows:

- There is a user U which sends coded tracking location information to an untrusted server S, using a key matrix m. The encoding algorithm runs on the user's end device, so none of the key's original information reaches the service provider.
- Key matrix m is a 3x3 matrix $\in \Re$ defined by the user manually or using a key generator. In the original version, it must be invertible to be able to recover the original data.
- Original location information of a path P is a tuple composed of (Latitude, Longitude, Time). The time unit is assumed in seconds, starting in 1 from when the path initiates. The original full timestamps may be stored in the database.
- User U may have different coded paths P_i on the server's database
- Each path P_i was coded with a key matrix m_i. This matrix can be the same for all paths or different every time. On the rest of the definition, a single path and a single key matrix will be referred to as P and m, respectively.
- The service provider provides access to the data by receiving a matrix and sending back data multiplied by the matrix. The server does not provide raw, encoded data.

The example, Fig. 1 shows the encoding of spatial-time coordinates of a certain route. As can be seen, the morphology of the path is substantially changed, obscuring the shape of the previous path and the coordinates domain in which the user moved. In this manner, the service provider does not know the real location of the users, who are the only actors who know their whereabouts and can decode their data. This is a unique characteristic of Matlock, given that all the information about the path will be coded into a line despite its shape and location. The formal definition of this behavior will be presented in a future work.

For example, tuples like (-1.752014322, 4.887933,4) and (-1.744073836, 4.885695754, 469) would become (24.47211968, 47.59943536, 19.63189003) and (4209.484535, 1442.597418, 2344.653474), respectively, when multiplied with the matrix m on Fig. 2. Notice that the coordinates of the coded values do not correspond to the regular geographical scale; however, these values are still geometrically valid for some geographical services. The original values can be recovered by multiplying the tuples by the inverse of matrix m.

One advantage of this technique against location obfuscation and geomasking is that Matlock allows recovering the original path, just by multiplying the data with the inverse of the key

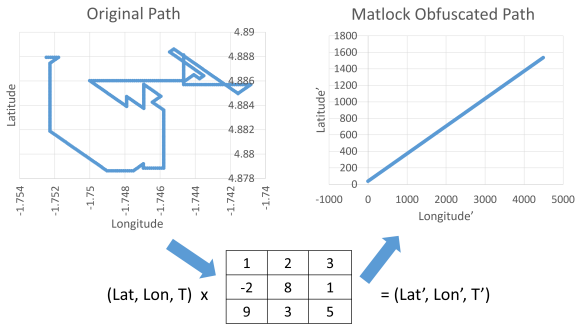

Fig. 1. Example of data transformation using Matlock.

matrix, thus the quality of the data is preserved. Now, the simplicity of Matlock comes at a price: in its current form, it cannot be considered a fully fleshed encryption mechanism due to its linear nature. If an attacker can grab hold of the coded dataset and estimate the corresponding locations to three points, then an equation system can find the matrix values and reverse engineer the technique. However, in [22], it was shown that the Matlock algorithm responded well to these attacks because the sensitivity to noise in the attacker's estimation could throw the decoded path very far from the original one. This sensitivity was tested by inducing noise to the locations to represent some inaccuracy by the attacker when trying to estimate the original locations. Results showed that the estimated paths showed maximum errors around 1000 times larger than the induced noise; this is, a very small noise in the estimation from the attacker would create a matrix that will generate a very different path from the original one, but with some similarity around the estimated points. The creation of a non-linear version of the algorithm is also part of future work.

However, this same characteristic can be used as an advantage. Recreating the same process in a systematic manner, can allow the user to create key matrices that will allow partial decoding of the path. And, as it will be seen later, this process can be tailored to generate decoding keys to generate controlled noise along the path, while being able to share a specific portion of the path. None of the consulted techniques can provide the user with the possibility of sharing just a portion of their location information without the intervention of a trusted LBS server or a third-party actor.

IV. The FishEye Matlock Algorithm

FishEye Matlock is an RFE-based protection mechanism for location data coded with the Matlock [11] algorithm, based on the findings in [22]. The user, who is always in control of the original key, can define the terms of how the data will be shared with others or even the service provider, using a disposable key matrix that will only reveal a portion of the path around a selected point. In addition, the user can define the parameters to select the amount of random noise added to the decoded points of interest and the level of accuracy of the complete path, depending on the level of trust in the other actor. In this sense, this technique follows the concept of Functional Encryption in [17]. It even provides a way to

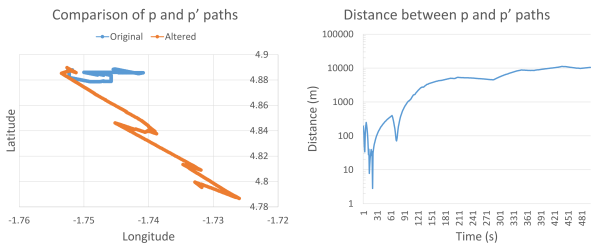

Fig. 2. Example of the altered paths and distances after applying the FishEye Matlock algorithm.

parameterize how the user shares the rest of the information, which is not commonly seen in the existing techniques.

The main idea is that the technique works like fish-eye lenses, returning an accurate version of the point of interest that the user wants to share but obscuring the rest of the path so that it is practically unusable.

This is done by generating an altered key matrix, different than the original one that the user uses to code/decode the complete path. This new matrix, when used to decode the coded data, shows the locations surrounding the desired points with precision, while the rest of the path appears as noisy data. In addition, the user can implement another level of protection to the data by adding noise to the revealed portion to not show only the exact location. This parameter can be defined depending on the user's level of trust in the recipient so that even the revealed portion can be protected.

Fig. 2 shows an example of the application of FishEye Matlock on a path. The user is assumed to want to share the 20th point, with 10 meters of added noise. The original path in blue can be seen on the left figure and the decoded path with FishEye in orange. Both paths intersect just in a small section at the beginning of the path, around the point of interest, and then the paths distance themselves substantially. The right figure shows the distance between each pair of corresponding points of the paths. Notice that the closest points have very little error, between 1 and 10 meters, while the rest of the path shows noise levels between hundreds and thousands of meters.

A. Definition

A general scenario for FishEye Matlock can be defined as follows: a user U wants to share the location at the ith second of a given path P.

1) Setup: The FishEye Matlock technique is run on the user's device to create a disposable key for the given scenario. It needs an array of Matlock-coded geolocation 3-component tuples (Lat, Lon, Time) and the master private key.

The process of creating an altered key matrix $m'_{i,k}$ starts by setting up the following parameters:

- Define the exact point i that will be shared.
- Define parameter k, which defines the distance between the 3 samples points. The noise level in the decoded full path is inversely proportional to the value of k. The maximum noise will be obtained with contiguous points $k = 1$.

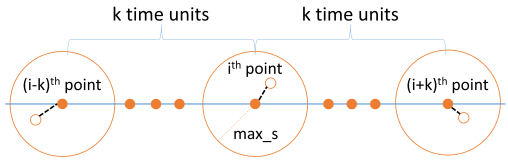

Fig. 3. Example of the noise generation process in FishEye Matlock.

- Define parameter max_s, which will define the maximum amount of noise added to the geographical coordinates of sample points. This noise will be transmitted to the altered path at the ith point.

Imagine that the user wants to share location $i = 250$, which corresponds to the following tuple (-1.747818, 4.884640, 250). The user does not trust the user provider and does not want to obfuscate much of the rest of the path. In this case, the k parameter has to be large. If $k = 100$, the other sample values are (-1.745789, 4.879667, 150) and (-1.743830, 4.886750, 350). This set of three coded points is denoted c. As seen in the experimental results, this large value of k implies a small error along the whole path. Assuming that the user does not have the coded points at hand, the three coded coordinates, the ith, $i-k$th, and $i+k$th, are requested to the server. These correspond to the point of interest and the other two sample points needed to calculate the random key matrix $m'_{250,100}$.

2) Key generation: To generate the key, the set of coded points $c_{i,k}$ is decoded by the user into the set $p_{i,k}$, using the private original key matrix m. For this example, the coded tuples in $c_{i,k}$ are (1347.494875, 488.5457646, 754.6422996), (2247.4829, 788.581491, 1254.641187), and (3147.482668, 1088.606344, 1754.655259).

The set of original points $p_{i,k}$ will be obfuscated to alter their location and time by adding random noise, based on the *max_s* parameter, as shown in Fig. 3. For each of the original points in orange, a new random point will be calculated inside the circumference defined by *max_s*. This could be done by generating random distances, angles, or points that fall inside the circumference. When applying a maximum of $max_s = 1$ meter of error to the locations, one possible result of altered tuples is (-1.745750493, 4.879613084, 150), (-1.747780772, 4.884656258, 250) and (-1.743891392, 4.886685055, 350). This new set of altered points $p'_{i,k}$ is the input for generating the altered key matrix. The changes performed here will affect the resultant matrix and transfer the alteration to the decoded path.

Then, the set of obfuscated decoded points $p'_{i,k}$ and the three coded points from the database $c_{i,k}$ will be used to calculate an obfuscated key matrix $m'_{i,k}$, by using Equation IV-A2 to solve the equation system. This will create a 3x3 key matrix that will differ from the original one and cannot be used to recover the original. Even if there is no or very little noise added to the sample points, the distance between the samples, k, will induce an error in the new key matrix that will protect the rest of the path.

$$m'_{i,k} = {p'_{i,k}}^{-1} c_{i,k} \tag{1}$$

187473.8571	62492.98635	104154.5708
67039.91382	22355.31669	37246.50161
9.842058739	3.280686812	5.467814024

Fig. 4. Altered example matrix $m'_{i,k}$ and $k = 100$

TABLE I
RESULTS OF APPLYING THE ALTERED KEY MATRIX $m'_{i,k}$ IN c

k	Original Time (s)	Lon	Lat	Time (s)	Distance from original (m)
100	19	-1.75214	4.88689	104.11876	13.8609
	50	-1.75226	4.88320	161.60296	5.3913
	150	-1.74570	4.87979	151.27317	9.9441
	250	-1.74798	4.88476	252.46145	11.1678
1	19	-1.75219	4.88699	15.32709	8.5307
	50	-1.75439	4.88787	74.16316	564.2524
	150	-1.75335	4.86640	202.91634	1,508.4206
	250	-1.75826	4.85520	385.93563	3,622.7693

With the obfuscated values $p'_{i,k}$ and the corresponding coded values $c_{i,k}$, the altered key matrix would be as in Fig. 4. Another example is shown in Fig. 5, where the sample points are taken with a $k = 1$, thus increasing the distance between paths.

The estimated obfuscated key matrix $m'_{i,k}$, when applied to the coded data, will generate a path p^* that differs from the original one, as in Fig. 2. This will protect the user's location information outside the area of interest while providing accuracy to the points used to estimate it. Table I shows the decoded coordinates and their distance from the original points, using the two example altered key matrices.

As seen in Table I, with a value of $k = 100$, distances are very small, between 3 and 15 meters, which matches the added noise. In the case of $k = 1$, the distance around the point of interest $i = 20$ remains small according to the added noise parameter. In contrast, the distances between the paths increase substantially in points far from the point of interest to the order of thousands of meters. This behavior will be explored in Section VI.

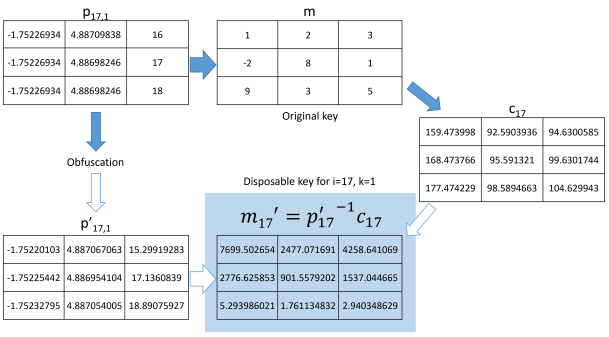

Fig. 5. Example of the altered key matrix generation in FishEye Matlock.

3) Protocol: The user U will send the new matrix $m'_{i,k}$ to the actor that needs to access the data. The third-party actor will query the service provider using the matrix and an invertible randomly generated one. In this way, neither the disposable matrix m' nor the decoded path p^* will travel on potentially unprotected channels. The third-party actor will decode the data with the inverse of the sent matrix.

If the third-party actor's permission expires or, if any major security issues are detected, the user can revoke the permissions for certain actors or request recodification of the complete path with a new key matrix.

V. ATTACK MODELS

The main goal of FishEye Matlock is for users to securely share specific location information with third-party actors, trusted and untrusted, by creating disposable keys that can decrypt only portions of a location trace.

Potential adversaries can include a) Malicious Third-party individuals or entities with malicious intent who seek to obtain sensitive location data for purposes such as surveillance, stalking, or targeted attacks; b) A mal intentioned service provider or employees who may attempt to access or misuse users' location data for unauthorized purposes; and c) adversaries with advanced technical capabilities who may attempt to exploit vulnerabilities in the FishEye Matlock mechanism to gain access to users' precise location data.

Some possible attack vectors are key compromise, parameter modification, compromise of the recipient's system. or side-channel attack.

On the first one, attackers gain unauthorized access to the user's original key, allowing them to decrypt the entire path data without any noise or obfuscation. If the provider or the user detects unusual activity on the system, a recodification can be easily performed on the data by multiplying by a new matrix composed of the inverse of the current key and the new key. In addition, some robust key management practices, including secure storage, access control, and revocation procedures, are important to prevent unauthorized access to the user's original key.

Second, a malicious third party may intercept communication between the user and the third-party key recipient. With no protection, the attacker could use the matrix to access a small portion of the full path. In this case, end-to-end encryption can be applied to protect the communication between the user and the recipient. If the attacker wants to tamper with the matrix to alter the result, then mechanisms for verifying the integrity of parameters can be used in the FishEye Matlock mechanism to detect and prevent tampering by malicious entities, like creating dependencies between parameters, matrices, and private information between user and recipient, etc. such that just the recipient can use the sent matrix properly.

In the third case, an attacker may gain access to the decoded location data if the recipient's system is compromised. The amount of compromised information will depend on the generated matrix. Recodification is the main tool to counteract this attack, which would make the existing generated matrix useless.

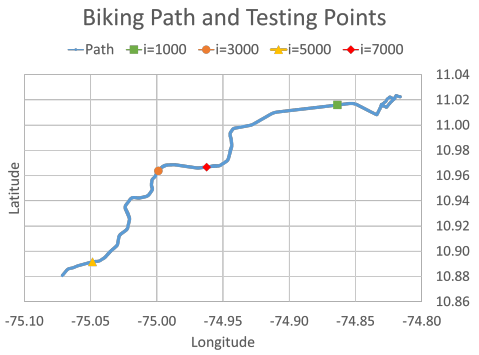

Fig. 6. Full round-trip biking path and testing points used in the experimental design.

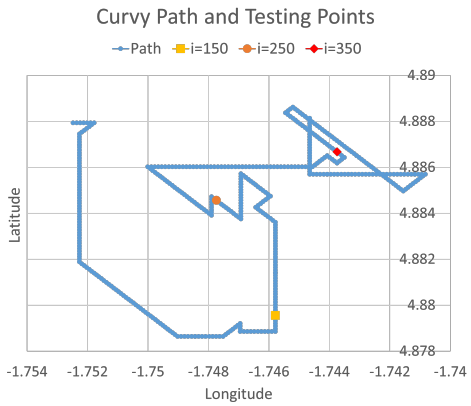

Fig. 7. Synthetically generated curvy path and testing points used in the experimental design.

The fourth option involves sophisticated attackers who exploit side-channel vulnerabilities to implement the FishEye Matlock mechanism to infer sensitive information about the user's location data. FishEye Matlock inherits the possibility of reverse engineering the key matrix by using estimated locations and encoded data. One solution being explored is the use of a non-reversible matrix, which only allows encoding but not decoding. Preliminary results have shown that the FishEye Matlock technique still works under this condition.

VI. Performance Evaluation

In this section, the FishEye Matlock technique will be evaluated to characterize its behavior and evaluate the impact of the parameter selection on providing a full differential privacy service.

A. Data Description

Two datasets are used in the experiments: a round-trip biking route in the metropolitan area of Barranquilla, Colombia, consisting of 9956 points, taken once every second. Both start and finish are at the same location, and the biker follows the same route twice. The path can be seen in the following figure (Fig. 6). The timestamps of the points on the path start on one and are all one second apart. This path starts at (11.02094, -74.82014) and finishes at (11.02249, -74.81611).

The second dataset is a synthetically generated Markov-chain-based random walk in Takoradi, Ghana. This path, consisting of 500 points a second apart from each other, shows a curvier path along the city, as can be seen in the next figure (Fig. 7). This path does not follow the actual roads of the city. The curvy characteristic is given by a higher chance of changing direction on the Markov chain. This path tries to show that, even though the path is concentrated in an area and different parts of the path may be close to each other, the technique still protects the location data far from the point of interest i. This path starts at (-1.752478, 4.887933) and finishes at (-1.744653, 4.888130).

B. Experimental Design

A set of points was selected from each scenario to perform the experiments, simulating the ones that a user may want to share. Three parameters are defined to evaluate the performance of the FishEye Matlock algorithm: Point position i, Space noise s, and Distance between sample points k, following the input parameters defined on the algorithm. Table II shows the values for each parameter. The PinWheel [3] technique was used to generate the noise, with $\phi = 123^o$ and the sampling method of N-Rand [23], with $N = 4$. The original key matrix used in the experiments is the same as the figure explaining Matlock (Fig. 1).

TABLE II
Definition of Parameters for Experimental Testing

Parameter	Values-Biking	Values-Curvy
Point positions	1000, 3000, 5000, 7000	150, 250, 350
Space noise - s (m)	1, 5, 10, 50, 100, 500, 1000	
Sample points distance - k	1, 10, 100, 1000	1, 10, 100

Given that the Curvy path has fewer than 1000 points, the k values will only be between 1 and 100.

The experiments will measure the distance between the estimated points p^* and the original points p as a measure of the impact of the algorithm. Five different indicators are shown for each scenario:

- Distance at the ith point.
- Maximum distance over the complete path.
- Minimum distance over the complete path.
- Average distance over the complete path.
- Average of the ± 25 points contiguous to the ith point.

C. Impact of Space Noise

The first set of experiments shows the impact of different levels of space noise s on the resultant path generated with the obfuscated key matrix m'. All scenarios showed a similar trend, so four examples were chosen to illustrate this behavior: Figs. 8 and 9 which show $i = 5000$ and $i = 7000$ for the biking path, and Figs. 10 and 11 which show $i = 250$ and $i = 350$ for the curvy path.

Experiments show that, in general, the noise levels increased with the maximum space noise, which means that the scale of this variable is reflected on the resultant obfuscated path.

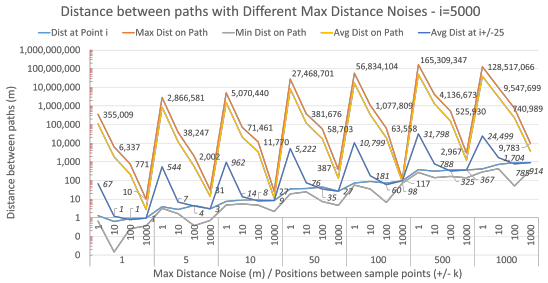

Fig. 8. Maximum, Minimum, Average, and Distance at point 5000, with no time noise and different maximum space noise - Biking Path.

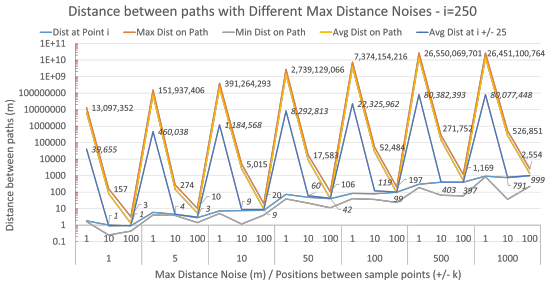

Fig. 10. Maximum, Minimum, Average, and Distance at point 250, with no time noise and different maximum space noise - Curvy Path.

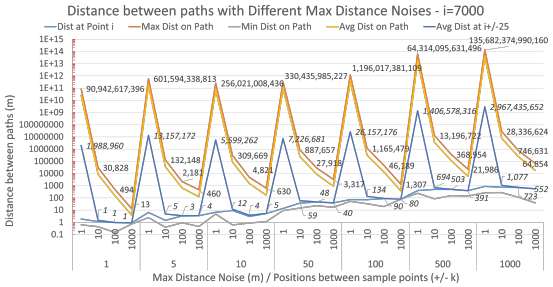

Fig. 9. Maximum, Minimum, Average, and Distance at point 7000, with no time noise and different maximum space noise - Biking Path.

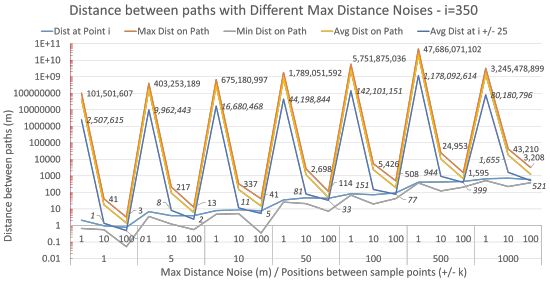

Fig. 11. Maximum, Minimum, Average, and Distance at point 350, with no time noise and different maximum space noise - Curvy Path.

Figs. 9 and 10 show a continuous increase in the noise. In some cases, the increase is close to being fully proportional, but this ratio is not linear in others. The main reason is that the randomness associated with the Pinwheel mechanism does not guarantee a given distance but a uniformly distributed added noise, which may impact some individual values.

For example, in Fig. 8, with a maximum noise of 1000m, the maximum distance on the path and the average distances of all the paths and at the ± 25 contiguous points are less than with a maximum noise of 500m. The noise distribution in that particular experiment was somewhat lower than in the previous one; however, the noise at point i is still larger than what was obtained in all previous scenarios. This means that despite the variability of the noise along the path (which remains on the scale of millions), the parameter selected by the user to protect the specific point still alters the data as intended. A similar behavior was seen in Fig. 11.

In terms of general protection, the maximum and average noise show a very high distance compared to the one obtained around the point of interest, showing that most obfuscated points stay very far from the original ones. On the other hand, the average distance of the ± 25 contiguous points gets very close to the distance at i when k increases, showing that the estimated matrix m' can provide a fairly accurate approximation to the original path while keeping a maximum level of noise far from i. Now, the minimum distance is not always exactly at the ith position, which can be attributed to the obfuscation of the points; however, the surrounding area of the point of interest is the one that gets the lower amount of added noise. This can also be attributed to the random noise distribution.

D. Impact of the k Parameter

The k parameter, or the distance between the sample points, was shown to have a great impact on the final amount of noise generated by the obfuscated key matrix m'. In the previous figures, it can be seen that when the k value is 1, the noise values are at their maximum level, and when the value increases, the noise level decreases drastically, up to a point in which it is very close to the max space noise value s.

This means that, when the sample points are distributed along the path, the estimated random matrix captures the behavior of the general path. On the other hand, if the points are contiguous, it is harder for the equation system solution to capture the general behavior of the path and, thus, obtain a matrix that will produce a path p^* similar to the original one, except in the vicinity of the point of interest i.

To show this behavior clearer, Figs. 12 and 13 show how, for a value of $k = 1$, the amount of noise on the rest of the path is maximized and there is a clear spike of distance reduction at the ith point. This implies that the path, not at the point of interest, is very well protected. Confirming the results from Figs. 8 through 11, when k increases, the distance between the paths reduces substantially along the whole trajectory, and the number of points with minimum noise increases. This means that the matrix uses points with greater differences and can capture the behavior at a certain scale but can still produce an adequate level of noise in the rest of the path. Now, when the k parameter reaches the maximum tested value of 1000 and 100 on each of the scenarios, it can be seen how the absolute distance between the estimated and the original points softens up to an almost linear, around the maximum added noise s. This can be seen especially in the second half of the biking route scenario, which follows the same direction instead of the

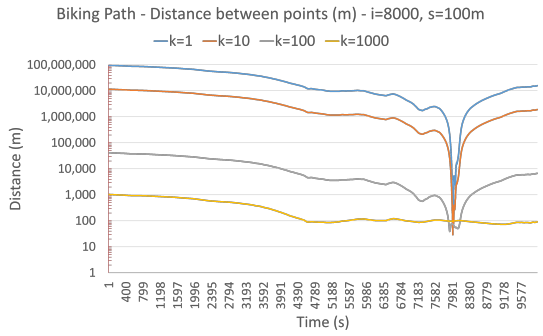

Fig. 12. Distance between the points along the path with maximum space noise of 100m and different values of k - Biking Path.

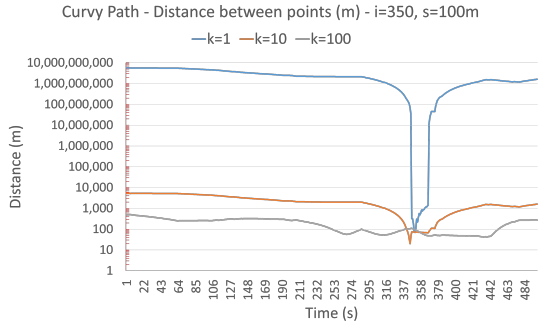

Fig. 13. Distance between the points along the path with maximum space noise of 100m and different values of k - Curvy Path.

second half that follows the inverse orientation.

The next figure (Fig. 14) shows the comparison among generated paths, with different values of k, confirming the impact of this parameter on the path-wide accuracy. In the zoomed-in version, it can be seen how when using $k = 1$, just a little set of points (in yellow) are in the range of the original path, while the others go beyond the geographical coordinate domain. However, the paths are almost identical when $k = 100$ (in orange), showing a difference very close to the space noise s.

VII. Discussion

Despite the good results of this technique, there are a few elements to consider. First, not all obfuscations of $p_{i,k}$ will

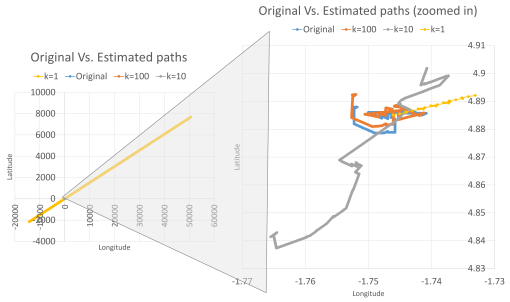

Fig. 14. Comparison of generated paths with different values of k - Curvy Path.

generate a valid matrix that can generate the desired noise. Sometimes, the noise at point i can grow beyond the maximum defined noise s. The program verifies that the $m'_{i,k}$ matrix will produce at most the maximum space noise. For example, in one experiment for the Biking path, at $i = 7000$, $k = 1$, and $max_s = 1$, it took more than 3.716.864 trials before finding a good combination of random values of $p_{i,1}$ to obtain the noise. In all other scenarios, the expected results were obtained on the first trial.

Two elements pending exploration in depth are the selection of the sample points, which may require an asymmetrical selection, especially when the point to be shared is at any of the paths' endpoints, and the nature of the k parameter, which still needs more study to determine its impact. For example, in the Biking and Curvy paths, a value of $k = 100$ produced very different levels of protection.

VIII. Conclusions and Future Work

This work introduces the FishEye Matlock technique, which provides a privacy-enforced way for users to share a selected portion of a path coded with the Matlock algorithm. The user generates an obfuscated key matrix to decode the path so that just the points of interest have the desired accuracy while the rest of the path is substantially distorted.

The mechanism uses two main parameters to generate the obfuscated key matrix m': parameter s, or the maximum level of space noise, which will affect each point to distance them from the original path, and the k parameter, or the distance between sample points, which determines the impact on the path as a whole, in terms of distortion. When the value of s increases, so does the minimum distance between estimated and original points, and when the value of k increases, the level of noise for all the points of the path or, in other words, the focus of the fish eye lens looking at the path, decreases. Thus, selecting these parameters is critical and depends on the information the user would like to reveal to the third-party actor. Results show an evident convergence of the average and minimum values towards the value at position i or point of interest.

Results show that inducing a controlled amount of error in the key matrix generation process will induce a similar error around the point of interest and a larger amount of noise in all the other path segments, making the information unusable for any real application. The total error in the decoded paths with the altered key matrix shows a "V" shaped behavior, with large errors to all points far from the points of interest and very small errors around the points that the user wants to share, as in Figs. 12 and 13. These results confirm the desired targeted protection or, in other words, users will be able to provide access to their data under their own conditions, which is imperative for a user-centered location privacy protection mechanism.

Results also showed that some points may have smaller noise than the one at i, which was assumed to be a consequence of the obfuscation process, but this was not explored in this work. In addition, future work will evaluate other aspects, like asymmetric values of k, their impact on different paths, and the value selection for the matrix.

ACKNOWLEDGMENTS

This work was partially supported by Universidad del Rosario and the Internal Research Project "Mecanismos de protección de privacidad de localización, basado en agregación y geocodificación" - IVTFA052.

DATA AND CODE AVAILABILITY

The Biking path can be found in IEEE Dataport at the following address: https://dx.doi.org/10.21227/6ddj-4v34 and the Takoradi synthetic path can be found in IEEE Dataport at the following address: https://dx.doi.org/10.21227/vn20-hb71. The code can be found in https://github.com/pedrowightman/FishEyeMatlock

REFERENCES

[1] M. Duckham and L. Kulik, "Location privacy and location-aware computing," in *Dynamic & mobile GIS: investigating change in space and time*, R. Billen, E. Joao, and D. Forrest, Eds. CRC Press, 2006, ch. 3, pp. 35–51. [Online]. Available: https://doi.org/10.1201/9781420008609

[2] S. Bai, G. Yang, J. Shi, G. Liu, and Z. Min, "Privacy-preserving oriented floating-point number fully homomorphic encryption scheme," *Security and Communication Networks*, vol. 2018, no. 1, p. 2363928, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/2363928

[3] P. Wightman, M. Zurbarán, and A. Santander, "High variability geographical obfuscation for location privacy," in *2013 47th International Carnahan Conference on Security Technology (ICCST)*, 2013, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CCST.2013.6922079

[4] C. Pang, C. Qiu, and N. Wang, "A geo-obfuscation approach to protect worker location privacy in spatial crowdsourcing systems," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, 2019, pp. 166–167.

[5] A. Dua, P. Singh, and J. Bapat, "Location privacy-preserving mechanism - a data-driven approach," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CONECCT52877.2021.9622699

[6] W. Cheng, R. Wen, H. Huang, W. Miao, and C. Wang, "Optdp: Towards optimal personalized trajectory differential privacy for trajectory data publishing," *Neurocomputing*, vol. 472, pp. 201–211, 2022. [Online]. Available: https://doi.org/10.1016/j.neucom.2021.04.137

[7] Y. Chen, A. Machanavajjhala, M. Hay, and G. Miklau, "Pegasus: Data-adaptive differentially private stream processing," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1375–1388. [Online]. Available: https://doi.org/10.1145/3133956.3134102

[8] I. Ullah, M. A. Shah, A. Wahid, A. Mehmood, and H. Song, "Esot: a new privacy model for preserving location privacy in internet of things," *Telecommunication Systems*, vol. 67, no. 4, pp. 553–575, Apr 2018. [Online]. Available: https://doi.org/10.1007/s11235-017-0352-x

[9] Z. Gao, Y. Huang, L. Zheng, H. Lu, B. Wu, and J. Zhang, "Protecting location privacy of users based on trajectory obfuscation in mobile crowdsensing," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6290–6299, 2022. [Online]. Available: https://doi.org/10.1109/TII.2022.3146281

[10] M. Zurbaran and P. Wightman, "Voka: Voronoi k-aggregation mechanism for privacy in location-based information systems," in *2017 International Carnahan Conference on Security Technology (ICCST)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CCST.2017.8167857

[11] P. M. Wightman, M. A. Jimeno, D. Jabba, and M. Labrador, "Matlock: A location obfuscation technique for accuracy-restricted applications," in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 1829–1834. [Online]. Available: https://doi.org/10.1109/WCNC.2012.6214082

[12] V. A. Kachore, J. Lakshmi, and S. Nandy, "Location obfuscation for location data privacy," in *2015 IEEE World Congress on Services*, 2015, pp. 213–220. [Online]. Available: https://doi.org/10.1109/SERVICES.2015.39

[13] X. Zhu, Y. Lu, X. Zhu, and S. Qiu, "A location privacy-preserving protocol based on homomorphic encryption and key agreement," in *2013 International Conference on Information Science and Cloud Computing Companion*, 2013, pp. 54–59. [Online]. Available: https://doi.org/10.1109/ISCC-C.2013.17

[14] S. Gupta and G. Arora, "Use of homomorphic encryption with gps in location privacy," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 2019, pp. 42–45. [Online]. Available: https://doi.org/10.1109/ISCON47742.2019.9036149

[15] H. Shen, M. Zhang, H. Wang, F. Guo, and W. Susilo, "A lightweight privacy-preserving fair meeting location determination scheme," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3083–3093, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.2965065

[16] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *Theory of Cryptography*, Y. Ishai, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 253–273. [Online]. Available: https://doi.org/10.1007/978-3-642-19571-6_16

[17] J. Alwen, M. Barbosa, P. Farshim, R. Gennaro, S. D. Gordon, S. Tessaro, and D. A. Wilson, "On the relationship between functional encryption, obfuscation, and fully homomorphic encryption," in *Cryptography and Coding*, M. Stam, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 65–84. [Online]. Available: https://doi.org/10.1007/978-3-642-45239-0_5

[18] C. Yang, P. Jiang, and L. Zhu, "Accelerating decentralized and partial-privacy data access for vanet via online/offline functional encryption," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8944–8954, 2022. [Online]. Available: https://doi.org/10.1109/TVT.2022.3174888

[19] A. Kaminsky, M. Kurdziel, S. Farris, M. Łukowiak, and S. Radziszowski, "Solving the cross domain problem with functional encryption," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 49–54. [Online]. Available: https://doi.org/10.1109/MILCOM52596.2021.9652958

[20] D. Sharma and D. C. Jinwala, "Encrypted data ordering with functional encryption," in *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, 2018, pp. 1–8. [Online]. Available: https://doi.org/10.1109/RAIT.2018.8389084

[21] D. Li, J. Wu, J. Le, X. Liao, and T. Xiang, "A novel privacy-preserving location-based services search scheme in outsourced cloud," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 457–469, 2023. [Online]. Available: https://doi.org/10.1109/TCC.2021.3098420

[22] L. Gonzalez, M. Zurbarán, P. Wightman, D. Jabba, M. Jimeno, and E. Zurek, "Sensitivity analysis and countermeasures for transformation-based location obfuscation," in *2014 International Carnahan Conference on Security Technology (ICCST)*, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CCST.2014.6987048

[23] P. Wightman, W. Coronell, D. Jabba, M. Jimeno, and M. Labrador, "Evaluation of location obfuscation techniques for privacy in location based information systems," in *2011 IEEE Third Latin-American Conference on Communications*, 2011, pp. 1–6. [Online]. Available: https://doi.org/10.1109/LatinCOM.2011.6107399

Pedro Wightman received his B.Sc. in Systems Engineering from the Universidad del Norte. He is a Doctor in Computer Science and Engineering from the University of South Florida. Currently, Pedro is an Associate professor at the School of Engineering, Science and Technology of the Universidad del Rosario. He is a senior member of IEEE and a senior researcher of the Ministry of Science. His research lines are location-based services, especially focused on location data privacy, AI in education, blockchain and medical information systems, communication infrastructure for the internet of things, and industry 4.0. He is the author of 3 technical books and several publications in indexed journals and international events.

Nicolás Avilán holds a B.Sc. in Physics from Universidad Nacional de Colombia, and a M.Sc., and a Ph.D. in Physics from Universidad de los Andes. He currently serves as a Principal Professor at the School of Engineering, Science and Technology at Universidad del Rosario. His primary research focuses on mathematical modeling across various systems, ranging from physiological studies to the simulation of light transmission through matter, the exploration of cosmological models, and the analysis of ground states for quantum field theories on curved backgrounds. Nicolás is the author of 14 papers published in indexed journals.

Augusto Salazar received his B.S. degree in Systems Engineering from Universidad del Norte, Barranquilla, Colombia (2003), and his M.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan (2012). He worked in the area of embedded systems for nine years at companies like Ericsson LMF (FI), Hitron Technologies (TW), and Proscend Communications (TW). Since 2012, he has been an Assistant Professor with the Department of Systems Engineering, Universidad del Norte. His research interests include embedded systems, mobile application development, and game analytics.