10-18-2020

ESPADE: An Efficient and Semantically Secure Shortest Path Discovery for Outsourced Location-Based Services

Bharath K. Samanthula

Divyadharshini Karthikeyan

Boxiang Dong

K. Anitha Kumari

cryptography

MDPI

Article

ESPADE: An Efficient and Semantically Secure Shortest Path Discovery for Outsourced Location-Based Services

Bharath K. Samanthula [1,*], Divya Karthikeyan [1], Boxiang Dong [1] and K. Anitha Kumari [2]

[1] Department of Computer Science, Montclair State University, 1 Normal Avenue, Montclair, NJ 07043, USA; karthikeyand1@montclair.edu (D.K.); dongb@montclair.edu (B.D.)

[2] Department of Information Technology, PSG College of Technology, Coimbatore, Tamil Nadu 641004, India; kak.it@psgtech.ac.in

* Correspondence: samanthulab@montclair.edu; Tel.: +1-973-655-5161

Received: 18 July 2020; Accepted: 2 October 2020; Published: 18 October 2020

Abstract: With the rapid growth of smart devices and technological advancements in tracking geospatial data, the demand for Location-Based Services (LBS) is facing a constant rise in several domains, including military, healthcare and transportation. It is a natural step to migrate LBS to a cloud environment to achieve on-demand scalability and increased resiliency. Nonetheless, outsourcing sensitive location data to a third-party cloud provider raises a host of privacy concerns as the data owners have reduced visibility and control over the outsourced data. In this paper, we consider outsourced LBS where users want to retrieve map directions without disclosing their location information. Specifically, our paper aims to address the following problem: Given a user's location s, a target destination t, and a graph G stored in a cloud, can users retrieve the shortest path route from s to t in a privacy-preserving manner? Although there exist a few solutions to this problem, they are either inefficient or insecure. For example, existing solutions either leak intermediate results to untrusted cloud providers or incur significant costs on the end-user. To address this gap, we propose an efficient and secure solution based on homomorphic encryption properties combined with a novel data aggregation technique. We formally show that our solution achieves semantic security guarantees under the semi-honest model. Additionally, we provide complexity analysis and experimental results to demonstrate that the proposed protocol is significantly more efficient than the current state-of-the-art techniques.

Keywords: shortest path; location-based service; cloud computing; semantic security; encryption

1. Introduction

Due to the advent of the Internet of Things (IoT), a wide range of smart devices (e.g., tablets and smartphones) are being used to boost businesses across several industries, including healthcare [1,2], manufacturing [3], and military [4]. The number of applications that provide services using geo-locations has also increased in recent years [5]. Specifically, the proliferation of smart devices is the driving force for the growth of Location-Based Services (LBS). LBS applications and platforms allow users to access relevant and updated information about their surroundings based on their real-time geo-location information. For example, navigation, gaming, advertising, and tracking are some domains that effectively leverage LBS [6].

Although the LBS feature renders dynamic user experience and enhances the way businesses can operate and interact with their customers, many such applications sacrifice user security for increased availability and performance [7]. For example, addressing the inherent privacy issues in LBS remains a critical challenge for many service providers [8,9]. In general, a user should

disclose his/her location data for the LBS provider to make accurate recommendations. This enables LBS providers to continuously track the user's personal information, such as their home address, travel plans, lifestyle, etc. Due to growing privacy concerns, users may not want to disclose their location information to LBS providers, but they may still want to get benefits from such applications. Additionally, the evolution of real-time LBS queries emphasizes the need for LBS providers to provide on-demand services. That is, LBS providers often require huge computational and storage resources to manage graph data and process location-aware queries. As a result, it is costly and challenging for LBS providers to manage on-premise infrastructure for delivering services. A natural solution to this problem is to adopt cloud computing technology [10] for efficient processing of LBS queries with on-demand scalability and increased resiliency [11]. The LBS provider can delegate their computational operations in addition to their data to the cloud. Nonetheless, the privacy issues mentioned above become even more challenging under an outsourced environment as the cloud service providers are remote and not trusted servers. A common approach to address the confidentiality of the location data is to encrypt it before being outsourced to a cloud. However, processing over encrypted data is not straightforward as the cloud cannot see the underlying data, thus affecting the quality of LBS. Therefore, we emphasize that there is a strong need to develop privacy-preserving technologies (e.g., [12,13]) that can equilibrate privacy and quality of LBS applications in a cloud environment.

In this paper, we focus on the Single-Source Single-Destination (SSSD) shortest path query, which is one of the commonly used LBS features, under an outsourced cloud environment. For example, consider finding the shortest path to the military base in a mission-critical search-and-rescue operation. To achieve data confidentiality, we assume that the geospatial data are represented as a graph and are encrypted before being outsourced to a cloud. Specifically, given an encrypted graph G stored in a cloud, the goal is to find the shortest path from the source to a destination in a privacy-preserving manner. In the literature, this problem is commonly referred to as Privacy-Preserving Shortest Path over Encrypted Graph (PSPEG) [14]. We emphasize that any solution to the PSPEG problem needs to address the following three privacy objectives:

1. **Privacy Objective 1 (PO1):** User's input information (i.e., source and destination locations) should not be revealed to the cloud service providers and any other users.
2. **Privacy Objective 2 (PO2):** The contents of graph G should never be revealed to the cloud service providers and unauthorized users.
3. **Privacy Objective 3 (PO3):** The shortest path information should be revealed only to the query issuer.

Existing PSPEG solutions (e.g., [14–16]) either do not meet all the above privacy objectives or are not very efficient. Therefore, the primary goal of this paper is to develop a solution that is both secure and efficient. Specifically, we address two research questions: (i) Investigate ways for the data owner to securely and efficiently outsource his/her graph data as well as the shortest path query processing task to a cloud. (ii) Study methods for the query issuer to efficiently retrieve the shortest path results from the cloud without compromising his/her location privacy. To address the above two questions, we adopt the Paillier cryptosystem [17] due to its efficiency and inherent additive homomorphic properties over encrypted data. The cloud server can directly operate on encrypted data to execute the steps involved in the shortest path-discovery process. Along this direction, we propose an Efficient and Semantically Secure Shortest Path Discovery over Encrypted Graph Data (ESPADE) protocol under a cloud environment. The proposed ESPADE protocol meets all of the three privacy objectives mentioned above and provides semantic security [18] under the semi-honest model of Secure Multi-party Computation (SMC) [19]. The main idea behind our protocol is to split the graph data into grids, apply homomorphic encryption operations using a novel data aggregation technique, and execute the shortest path discovery process in an iterative process. The underlying steps involved in ESPADE are based on the well-known Dijkstra's algorithm [20]; thus, ensuring the correctness. Our performance analysis shows that ESPADE is more secure and efficient compared to the existing PSPEG protocols. It is worth noting that our solution can be incorporated into critical outsourced

LBS applications (e.g., military rescue operations) where the secure computation of single-source single-destination shortest path queries is a fundamental task in the graph mining process.

Main Contributions

The existing solutions to the PSPEG problem are either insecure (e.g., leak intermediate results to the cloud providers) or not very efficient. Our proposed ESPADE protocol is both secure and efficient over existing solutions. The main contributions of this paper are summarized below:

1. **Semantic Security**: ESPADE meets all three privacy objectives mentioned earlier. That is, the contents of the outsourced graph G and the user's input query are never revealed to the cloud service providers and any unauthorized users. This is because our solution is designed to achieve semantic security under the semi-honest model of SMC. We refer the reader to Section 6.1 for more details.
2. **Efficiency**: Our protocol is significantly efficient (in terms of both computation and communication-wise) compared to existing solutions. It is worth noting that the majority of expensive computations in ESPADE are performed by cloud providers, thus minimizing the costs on the end-user.
3. **Correctness**: The steps involved in ESPADE are similar to the ones in the standard Dijkstra's algorithm. The only difference is that the underlying operations are performed either over encrypted or randomized data. For any given G and shortest path query Q, the shortest path returned by our protocol is the same as the one that would be returned by executing Dijkstra's algorithm on $\{G, Q\}$.
4. **Flexibility**: Upon outsourcing the graph data to the cloud, the data owner does not have to participate in any other operations. Specifically, the end-users can issue SSSD queries directly to the cloud and the majority of the query processing task is done by the cloud providers, which suffices in the main purpose of outsourcing in the first place.

The remainder of this paper is organized as follows: Section 2 discusses our system model. In Section 3, we touch upon more closely related work to ours. Section 4 presents the background information. Section 5 discusses the proposed ESPADE protocol in detail along with a running example. We present the security proofs and the comparative performance analysis of ESPADE with experimental results in Section 6, and conclude the paper with future work in Section 7.

2. Problem Model

In our problem setting, we consider three types of entities: (i) Alice (ii) Federated Cloud and (iii) Bob. The proposed system model along with the information flow among different entities is shown in Figure 1. Next, we discuss the role of each of these entities.

- **Alice**: We assume that Alice (data owner) holds sensitive geospatial data represented as a graph $G = \{V, E, W\}$, where $V = \{v_1, \ldots, v_n\}$ denotes the set of vertices, E denotes the edges connecting those vertices associated with weights W. In our model, we consider that G is an undirected weighted graph represented as an $\alpha \times \alpha$ grid matrix (refer to Section 5 for more details). For example, consider a graph used by Google Maps where G represents a road network. Here each vertex in V will correspond to a given point on the road network, and each edge E will correspond to the road segment that connects any two points on the road network. In this case, weight can be the distance between junctions or traffic flow.
- **Federated Cloud**: Without loss of generality, let Alice outsource G (in encrypted format) to a Federated Cloud (FC) environment consisting of two cloud services providers C_1 and C_2. We assume that C_1 and C_2 are semi-honest and do not collude. This is a realistic model as it is being used in several existing works (e.g., in [21–24]). This is because most of the cloud service providers are well-known IT organizations and it is highly unlikely that any two cloud service providers will collude as it may damage their reputation and can adversely affect their revenues.

- **Bob**: The end-user Bob issues SSSD shortest path queries to FC. C_1 and C_2 will collaboratively compute the shortest path from source s to t, and return the path to Bob.

Under the above problem setting, we formally define the proposed ESPADE protocol as follows:

$$\text{ESPADE}(G, \langle s, t \rangle) \rightarrow \text{SP}(s, t) \tag{1}$$

where G is known only to Alice and is outsourced (after proper randomization) to FC. $\langle s, t \rangle$ denote the source and destination locations, respectively, known only to Bob. At the end of the ESPADE protocol, the shortest path route, denoted by $\text{SP}(s, t)$, should be revealed only to Bob.

Briefly, our model involves the following main steps: (1) C_2 initially generates a public-private key pair (pk, sk) based on the Paillier cryptosystem [17] and distributes the public key pk to Alice, C_1 and Bob, (2) Alice randomly splits her graph data G in a systematic way and outsources them to C_1 and C_2, (3) Bob randomly splits his shortest path query information and forwards them to FC, (4) C_1 and C_2 involve secure computations to collaboratively extract the shortest path information, and (5) FC sends the randomized sub-graph data to Bob, who then aggregates and extracts the shortest path information based on Dijkstra's algorithm.

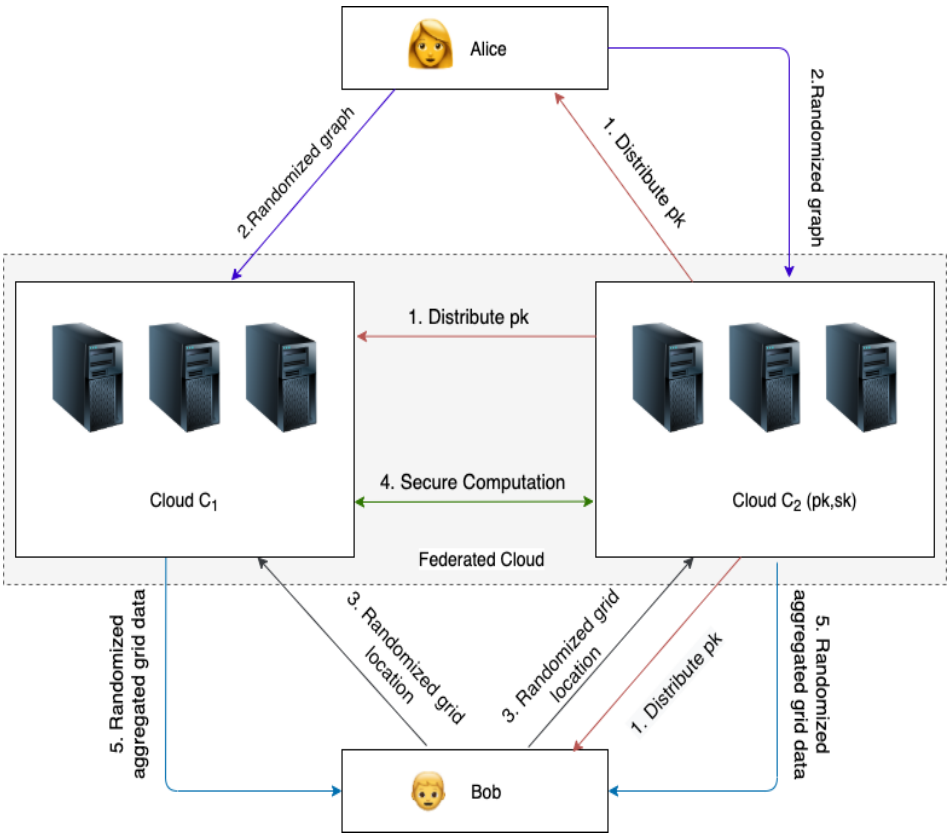


Figure 1. The Proposed ESPADE Model.

3. Related Work

In this section, we discuss the existing methods for privacy-preserving shortest path (PPSP) computation. We also demonstrate the limitations of the existing solutions.

3.1. Privacy-Preserving Shortest Path over Plaintext Data

In this sub-section, we briefly touch upon existing privacy-preserving shortest path solutions in which the graph data stored on the server-side are in a non-encrypted format.

3.1.1. Obfuscation Methods

In obfuscation methods [25], Bob does not send his query directly to the LBS provider. For example, in [26], Bob computes a set of dummy locations and obfuscates his source location s with the fake locations. Similarly, he obfuscates the destination location t with many fake locations. Suppose S and T denote the obfuscated source and destination locations, respectively. Bob forwards (S, T) to the LBS provider. Upon receiving, the LBS provider computes the shortest path from every location in S to every location in T, based on G resulting in $|S| * |T|$ paths which are forwarded to Bob. Finally, Bob retains the shortest path that corresponds to the original source-destination location pair. It is worth noting that any PPSP solution based on the obfuscation method is not secure as it will reveal substantial information about the query to the LBS provider. For example, LBS knows that the original source (or destination) location is one among the many locations in the set S (or T). Additionally, obfuscation-based methods assume that outsourced data (i.e., G in our case) are known to the LBS provider, whereas in our problem setting the contents of G are hidden from both cloud service providers.

3.1.2. Private Information Retrieval (PIR) Methods

In PIR [27], the server is assumed to hold a database of items and the end-user wants to retrieve the *ith* item from the database in an oblivious manner. Computational PIR [28] is one kind of PIR which can work with a single server and utilizes cryptographic techniques to ensure the privacy of the query from a computationally bounded server. Existing PIR-based solutions to PPSP assume that the outsourced graph database is not privacy-sensitive and is known to the LBS provider. For example, Mouratidis and Yiu [29] proposed a hardware-aided PIR-based solution to PPSP that relies on a tamper-resistant secure co-processor installed at the server-side. However, such schemes require decrypting the data in a secure area at the cloud and perform the computation on decrypted data; thus, they do not protect data access patterns from the server [30].

In this paper, we assume that the graph data are sensitive and thus the above schemes are not applicable to our problem domain, especially for outsourced environments where the outsourced data are in an encrypted format and has to be protected even from the cloud service providers.

3.2. PPSP over Encrypted Graph Data

In the past decade, various techniques for location-based query processing over outsourced encrypted data have been proposed [14–16,31,32]. However, there exist only a few solutions to the PPSP problem over encrypted graph data. Blanton et al. [15] proposed a data-oblivious algorithm for the SSSD shortest path problem on protected information. First, their algorithm is well-suited only for dense graphs. Second, their framework is based on the (k, n) threshold linear secret sharing scheme that requires at least three parties whereas our framework utilizes a more practical two-cloud model.

Zhang et al. [16] proposed a shortest path computing framework based on homomorphic encryption and secure multi-party computation. However, their protocols assume that only the edge-weight information was encrypted; and thus, their solution reveals the vertex information to the cloud server. Additionally, they utilized 1-out-of-n oblivious transfer cryptographic primitive, which can be prohibitively expensive and their solution requires the participation of the data owner during the shortest path computation, whereas in our protocol the data owner does not participate in any operations after the data outsourcing step.

Samanthula et al. [14] proposed two PPSP solutions over encrypted graph data, referred as PSPEG_1 and PSPEG_2, under different cloud settings. First, PSPEG_1 utilizes a single cloud service provider whereas PSPEG_2 uses a two-cloud model similar to ours. However, both of these protocols incur significant computation and communication costs on Alice and Bob. Furthermore, there is no provable way to quantify the security guarantees of these two protocols as they did not provide any security proofs. In Section 6, we formally show that our proposed ESPADE protocol is more efficient

and secure compared to both PSPEG_1 and PSPEG_2. Specifically, the proposed ESPADE protocol utilizes a data aggregation technique under encryption to boost the performance of each iteration in the shortest path discovery process whereas PSPEG_1 and PSPEG_2 incur significant costs (computation, communication, and round) on the cloud providers and the end-user. We refer the reader to Section 6 for a detailed analysis on the secure guarantees of ESPADE and its comparison with existing work.

4. Preliminaries

In this section, we discuss basic concepts and core algorithms that are utilized in ESPADE as background knowledge. First, we present the main steps involved in the Dijkstra's algorithm. Then, we discuss the importance of homomorphic encryption in outsourced environments along with the properties of Paillier Cryptosystem, which are crucial to perform certain operations over encrypted data in our proposed protocol. Finally, we discuss the two-party secure multiplication (SM) protocol over encrypted data, which is used as a building block in ESPADE. Some common notations utilized in this paper are shown in Table 1.

Table 1. Common Notations.

Notation	Description
SSSD-SP	Single-source single-destination shortest path
ESPADE	Efficient and semantically secure shortest path discovery over encrypted graph data
FC	A federated cloud environment consisting of two cloud service providers C_1 and C_2
G	A weighted graph consisting of $V = \{v_1 \ldots, v_n\}$ vertices with E edges and W weights
n	The total number of grids G is divided into
$v_{i,j}$	jth vertex in grid location i
$v_{i,j}^k$	kth neighbor of vertex $v_{i,j}$
$w_{i,j}^k$	Weight between the two vertices: $v_{i,j}$ and $v_{i,j}^k$
(s, t)	Source and destination locations
$SP(s, t)$	Shortest path from s to t based on G
(pk, sk)	A pair of public-private key pair generated based on Paillier cryptosystem
E_{pk}	Paillier's Encryption function with public key pk
D_{sk}	Paillier's Decryption function with private key sk
$r \in_R \mathbb{Z}_N$	A random number chosen uniformly in the group $\mathbb{Z}_N$

4.1. The Dijkstra's Algorithm

There exist several algorithms to find the single-source shortest path for a weighted graph $G = \{V, E, W\}$, such as Dijkstra's and Bellman–Ford algorithms. In this paper, we focus on Dijkstra's algorithm as it is one of the most commonly used algorithms for non-negative weighted graphs. In general, most of the location-based services, such as map directions, deal with non-negative weighted graphs. For brevity, in this paper, we restrict our discussion to non-negative weighted graphs. Given a weighted graph G, Dijkstra's algorithm can be used to compute the shortest path from a given source (s) to the destination (t), denoted by $\text{SP}(s, t)$. Let us consider a graph G with the following elements:

- Vertices denoted by u or v;
- Each edge that connects two vertices (u, v) has weights associated with it, denoted by $w_{u,v}$.

For a given (s, t) pair, the algorithm will traverse neighboring vertices in G, finding the shortest path by adding edge weights and replacing the current shortest path if a lower total distance is found. The main steps involved in Dijkstra's algorithm are discussed below.

A. Initialization

- The current vertex cv is marked as source s;
- Each vertex in G is initially marked as unvisited;
- All vertices are assigned with ∞ as the distance from s, while for s itself the distance is assigned as 0. That is, $dist(v) = \infty, \forall\, v \in V \wedge v \neq s$;
- $\mathrm{SP}(s,t) = \{\}$.

B. Iterative Process

- Step 1: For the current vertex cv, consider all unvisited vertices that are directly connected to cv. Let us denote this set by L.
- Step 2: For each vertex $u \in L$, calculate its new distance from the source as $dist'(u) = dist(cv) + w_{u,cv}$. If $dist'(u) \leq dist(u)$, then update vertex u's distance as $dist'(u)$. Otherwise, it is unchanged.
- Step 3: Mark cv as visited and update $\mathrm{SP}(s,t)$ as $\mathrm{SP}(s,t)\|cv$. If $cv = t$, then terminate and return $\mathrm{SP}(s,t)$ as the output. Otherwise, select the vertex $m \in L$, which has the smallest distance, set cv to m, and proceed to Step 1.

4.2. Homomorphic Encryption and Paillier Cryptosystem

Homomorphic Encryption (HE) provides a way to process the information in an encrypted format and produce the same results as if the operations were performed over plaintext data [33]. In outsourced environments, HE enables untrusted servers to directly perform operations over encrypted data while maintaining the confidentiality of the data. Existing HE schemes are broadly categorized into Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE) schemes. On one hand, FHE supports multiple types of operations (i.e., addition and multiplication) over encrypted data, but the existing FHE schemes are very expensive and not practical. On the other hand, PHE schemes support only one type of operation, either addition or multiplication, over encrypted data.

The Paillier cryptosystem [17] is an asymmetric PHE scheme that can support an arbitrary number of addition operations over encrypted data and whose security relies on the hardness of integer factorization. Let E_{pk} denote the Paillier's encryption function, where the public key pk consists of (N, g). Here N denotes the RSA modulurats and g denotes the geneor from group $\mathbb{Z}_N^*$. The encryption of a message m under Paillier's scheme is given by, $E_{pk}(m) = (g^m * r^N) \bmod N^2$, where $r \in_R \mathbb{Z}_N$. For any two given messages, $m_1, m_2 \in \mathbb{Z}_N$, the following properties of Paillier's encryption function always hold:

- **Additive Homomorphism**: The output of multiplying the ciphertexts of m_1 and m_2 is equivalent to the encryption of $m_1 + m_2 \bmod N$. That is,

$$E_{pk}(m_1) * E_{pk}(m_2) = E_{pk}(m_1 + m_2 \bmod N)$$

- **Partial Multiplication**: Given a constant $b \in \mathbb{Z}_N$, rising the ciphertext of m_1 to the power of b is equivalent to the encryption of $b * m_1$. That is,

$$E_{pk}(m_1)^b = E_{pk}(b * m_1)$$

- **Semantic Security**: Paillier's encryption function is a probabilistic scheme meaning that encryptions of the same message will result in different ciphertexts. Therefore, given a set of ciphertexts, an adversary cannot deduce any information about the underlying plaintexts. That is, ciphertexts are indistinguishable from one another; thus, the scheme exhibits the semantic security property [18].

In this paper, we adopt Paillier's scheme as the underlying encryption scheme due to its security guarantees, homomorphic properties, and possible optimizations (refer to Section 6.2.2 for more details).

4.3. Secure Multiplication (SM)

As mentioned above, Paillier's encryption scheme allows parties to locally perform addition operations over encrypted data. However, multiplication over encrypted data cannot be done locally and it requires the help of the party holding the corresponding private-key. Without loss of generality, suppose C_1 holds private data $\langle E_{pk}(a), E_{pk}(b)\rangle$ and C_2 holds the corresponding Paillier's private-key sk. We assume that a and b are not known to C_1 and C_2. Under this setting, the goal of Secure Multiplication (SM) [34] is to enable C_1 and C_2 to jointly compute $E_{pk}(a * b)$. The output of the SM protocol—i.e., $E_{pk}(a * b)$, should be known only to C_1. During the execution of SM, the contents of a, b, and $(a * b)$ should not be revealed to C_1 and C_2.

The main steps involved in the SM protocol [34] are highlighted in Algorithm 1. Initially, C_1 generates two random numbers $r_a, r_b \in_R \mathbb{Z}_N$, and uses them to randomize the encrypted inputs. That is, C_1 computes $A = E_{pk}(a + r_a)$ and $B = E_{pk}(b + r_b)$ using the additive homomorphic properties of Paillier's scheme and forwards these values to C_2. Upon receiving, C_2 decrypts (A, B) to get (a', b'), respectively. Then, it multiplies a' and b' whose equation can be expanded as follows:

$$a' * b' = (a + r_a) * (b + r_b) = a * b + a * r_b + b * r_a + r_a * r_b$$

At this point, C_2 knows that $p = a' * b'$, which it encrypts under pk and sends the resulting encrypted value P to C_1. Finally, C_1 performs homomorphic operations on P locally and removes the randomized factors (i.e., $a * r_b, b * r_a$ and $r_a * r_b$) under encryption to get $E_{pk}(a * b)$ as the final output.

Algorithm 1 SM$(E_{pk}(a), E_{pk}(b)) \rightarrow E_{pk}(a * b)$

Require: C_1 holds $\langle E_{pk}(a), E_{pk}(b)\rangle$ and C_2 holds sk. a and b are not known to C_1 and C_2

1: C_1:

(a). Pick two random numbers $r_a, r_b \in \mathbb{Z}_N$
(b). $A \leftarrow E_{pk}(a) * E_{pk}(r_a)$
(c). $B \leftarrow E_{pk}(b) * E_{pk}(r_b)$
(d). Send A, B to C_2

2: C_2:

(a). Receive A and B from C_1
(b). $a' \leftarrow D_{sk}(A)$; $b' \leftarrow D_{sk}(B)$
(c). $p \leftarrow a' * b' \bmod N$
(d). $P \leftarrow E_{pk}(p)$
(e). Send P to C_1

3: C_1:

(a). Receive P from C_2
(b). $y \leftarrow P * E_{pk}(a)^{N-r_b}$
(c). $y' \leftarrow y * E_{pk}(b)^{N-r_a}$
(d). $E_{pk}(a * b) \leftarrow y' * E_{pk}(r_a * r_b)^{N-1}$

Example 1. *Suppose* $a = 59$ *and* $b = 58$. *For simplicity, let* $r_a = 1$ *and* $r_b = 3$. *Assume that* C_1 *holds* $\langle E_{pk}(59), E_{pk}(58)\rangle$. *Various intermediate results computed during the execution of SM protocol are as follows. Initially,* C_1 *computes* $A = E_{pk}(a) * E_{pk}(r_a) = E_{pk}(60)$, $B = E_{pk}(b) * E_{pk}(r_b) = E_{pk}(61)$ *and sends them to* C_2. *Then,* C_2 *decrypts and multiplies them to get* $p = 3660$. *After this,* C_2 *encrypts* p *to get* $P = E_{pk}(3660)$ *and sends it to* C_1. *Upon receiving* P, C_1 *computes* $y = E_{pk}(3660 - a * r_b) = E_{pk}(3483)$, *and* $y' = y * E_{pk}(-b * r_a) = E_{pk}(3425)$. *Finally,* P_1 *computes* $E_{pk}(a * b) = y' * E_{pk}(-r_a * r_b) = E_{pk}(3422)$.

5. The Proposed ESPADE Protocol

In this section, we present our ESPADE protocol in detail. As mentioned in Section 2, our proposed protocol consists of three parties: Alice, Federated Cloud, and Bob. ESPADE is constructed based on the Paillier cryptosystem and by utilizing the SM primitive as a building block. For the rest of this paper, we explicitly make the following assumptions:

1. Alice's geospatial data are represented as a weighted graph G with V vertices and W weights. The contents of G are sensitive and thus need to be kept confidential from cloud service providers and unauthorized parties. We assume that each vertex in G is associated with a unique identification number—for example, denoting the combination of latitude and longitude information.
2. C_2 generates the public-private key pair (pk, sk) based on the Paillier's scheme and securely distributes pk to Alice, C_1, and Bob. We assume that there exist secure communication channels (e.g., SSL) between each pair of parties participating in our protocol.
3. Similar to existing work [14–16], we assume that all the participating parties in our protocol are semi-honest [19]. The semi-honest model is a practical security model, due to the following reasons. First, building protocols under the semi-honest model is an important first step for constructing protocols under stronger security models (e.g., against covert and malicious adversaries). Second, protocols under the semi-honest model are typically considered to be quite efficient, which may not be the case for other adversarial models. Third, protocols that are proven to be secure under the semi-honest model can prevent inadvertent leakage of information among participating parties. Finally, it is highly unlikely that the well-established cloud service providers (e.g., Amazon and Microsoft) would deviate from the prescribed protocol and collude, as this would damage their reputation and consumer trust. Therefore, we believe that the semi-honest model is a practical security model for our problem domain.

The goal of ESPADE is to securely outsource G to FC and execute Bob's shortest path query (s, t) in a privacy-preserving manner. The proposed ESPADE protocol consists of the following two stages:

- **Stage 1—Secure Outsourcing of Graph G (SOG)**: In this stage, Alice transforms her graph data G into a proper $\alpha \times \alpha$ grid matrix. During this process, Alice relies on our data aggregation technique to intelligently capture the information in each grid. After this transformation, Alice outsources the aggregated graph information to the federated cloud environment using the randomization approach. At the end of this stage, only C_1 knows the encrypted graph data.
- **Stage 2—Secure Retrieval of Shortest Path (SRSP)**: In this stage, Bob securely sends his shortest path query $Q = \langle s, t \rangle$ to FC. Then, C_1 and C_2 jointly involve secure computations to retrieve the shortest path in an iterative process, based on Dijkstra's algorithm. At the end of this stage, only Bob knows the shortest path from s to t.

The main steps involved in the proposed ESPADE protocol are given in Algorithm 2. Next, we explain each of the two stages of ESPADE in detail.

Algorithm 2 ESPADE($G, \langle s, t \rangle$) $\rightarrow$ SP(s, t)

Require: Alice holds private graph G; C_2 holds private key sk; Bob holds the shortest path query $\langle s, t \rangle$
(Note: parameters pk, ℓ, m, and n are public)

{**Stage1—Secure Outsourcing of Graph**}

1: Alice:

(a). **for** $1 \leq i \leq n$ **do**:

- Compute M_i from G, m, and ℓ
- Compute aggregated grid information T_i from M_i
- Split $T_{i,j}$: $T^1_{i,j} = T_{i,j} + r_{i,j} \bmod N$ and $T^2_{i,j} = N - r_{i,j}$, for $1 \leq j \leq 3m + 1$

(b). Send T^1 to C_1 and T^2 to C_2

2: C_2:

(a). Receive T^2 from Alice
(b). $F_{i,j} \leftarrow E_{pk}(T^2_{i,j})$, for $1 \leq i \leq n$ and $1 \leq j \leq 3m + 1$
(c). Send F to C_1

3: C_1:

(a). Receive T^1 from Alice and F from C_2
(b). $G'_{i,j} \leftarrow E_{pk}(T^1_{i,j}) * F_{i,j} \bmod N^2$, $1 \leq i \leq n$ and $1 \leq j \leq 3m + 1$

{**Stage2—Secure Retrieval of Shortest Path**}

4: SRSP($G', \langle s, t \rangle$)

5.1. Secure Outsourcing of Graph G (SOG)

During Stage 1, Alice first divides her graph G into $\alpha \times \alpha$ square grids (for example, 1 mile by 1 mile). Let n denote the total number of grids in G and g_v denote the grid ID in which a vertex v resides. Each grid is represented by the set of vertices that reside inside the grid, their neighbors, and the associated weights. Alice represents each piece of grid information as a matrix where each row corresponds to a vertex in that grid. Suppose ℓ denotes the maximum number of unique vertices in each grid and m denotes the maximum of 1-hop neighbors, a vertex in G can have. Upon dividing the graph G into n grids, Alice constructs the matrix M_i for each grid, for $1 \leq i \leq n$. Each row in M_i corresponds to a particular vertex in grid i and its m neighboring vertex information. Specifically, for grid i, the number of unique vertices are denoted by $v_{i,1}, \ldots, v_{i,\ell}$. For each vertex $v_{i,j}$ in grid i, Alice stores $3m$ entries, such that each entry consists of the neighboring vertex, its associated edge weight, and the grid ID of the neighboring vertex, where $1 \leq j \leq m$. As an example, for vertex $v_{i,j}$, Alice stores $\langle v^1_{i,j}, w^1_{i,j}, g_{v^1_{i,j}} \rangle, \ldots, \langle v^m_{i,j}, w^m_{i,j}, g_{v^m_{i,j}} \rangle$. In this case, $v^1_{i,j}, \ldots, v^m_{i,j}$ and $w^1_{i,j}, \ldots, w^m_{i,j}$ denote the neighboring vertices of $v_{i,j}$ and the corresponding edge weights, respectively, for $1 \leq j \leq m$. Additionally, $g_{v^1_{i,j}}$ denotes the Grid ID of vertex $v^1_{i,j}$. A sample snapshot of the grid information captured in M_i is shown below.

$$M_i = \begin{bmatrix} v_{i,1} & \langle v^1_{i,1}, w^1_{i,1}, g_{v^1_{i,1}} \rangle & \ldots & \langle v^m_{i,1}, w^m_{i,1}, g_{v^m_{i,1}} \rangle \\ v_{i,2} & \langle v^1_{i,2}, w^1_{i,2}, g_{v^1_{i,2}} \rangle & \ldots & \langle v^m_{i,2}, w^m_{i,2}, g_{v^m_{i,2}} \rangle \\ \vdots & \vdots & \vdots & \vdots \\ v_{i,\ell} & \langle v^1_{i,\ell}, w^1_{i,\ell}, g_{v^1_{i,\ell}} \rangle & \ldots & \langle v^m_{i,\ell}, w^m_{i,\ell}, g_{v^m_{i,\ell}} \rangle \end{bmatrix}$$

For security reasons, we assume that all the matrices constructed from G are of the same size; that is, $\ell \times m$. However, if a grid has less than ℓ vertices, Alice can add dummy entries. Similarly, if a vertex has less than m neighbors, she can insert dummy values.

Upon creating all the matrices, we propose that Alice adopts the following data aggregation technique to reduce its outsourcing costs, as well as later query processing costs for the end-users. Alice transforms matrix H_i into a vector T_i of size $3m + 1$ by concatenating the column-wise entries in M_i. That is, the first entry of T_i is computed as $\langle v_{i,1} \| v_{i,2}, \ldots, \| v_{i,\ell} \rangle$, the second entry as $\langle v^1_{i,1} \| v^1_{i,2}, \ldots, \| v^1_{i,\ell} \rangle$, and so on. After transforming all the matrices into vectors, Alice needs to outsource T_i's to FC in an encrypted format. However, for large values of n, m and ℓ, it would be expensive for Alice to encrypt T_i, for $1 \leq i \leq n$. Therefore, we propose the following approach for Alice to outsource $T_{i,j}$. Alice selects a random number $r_{i,j}$ and splits $T_{i,j}$ into two random shares: $T^1_{i,j} = T_{i,j} + r_{i,j} \bmod N$ and $T^2_{i,j} = N - r_{i,j}$. It is worth noting that $T_{i,j} = T^1_{i,j} + T^2_{i,j} \bmod N$ always holds, for $1 \leq i \leq n$ and $1 \leq j \leq 3m + 1$. Then, Alice outsources $T^1_{i,j}$ and $T^2_{i,j}$ to C_1 and C_2, respectively. Upon receiving $T^2_{i,j}$, C_2 computes $E_{pk}(T^2_{i,j})$ and forwards it to C_1. Finally, C_1 computes $E_{pk}(T^1_{i,j}) * E_{pk}(T^2_{i,j}) \bmod N^2$ which is equivalent to $E_{pk}(T^1_{i,j} + T^2_{i,j}) = E_{pk}(T_{i,j})$. We denote the final encrypted dataset by G', which is known only to C_1.

5.2. Secure Retrieval of Shortest Path (SRSP)

Following from Stage 1, C_1 has encrypted graph dataset G'. During Stage 2, Bob with private input $\langle s, t \rangle$, C_1 with private input G' and C_2 with private key sk wants to jointly find the shortest path from s to t in a privacy-preserving manner. At the end of Stage 2, SP(s, t) should be known only to Bob.

The main steps involved in Stage 2 of ESPADE are shown in Algorithm 3. Next, we discuss the steps of Stage 2 in detail below:

- To start with, Bob creates a graph G_s that initially contains no values except his starting point s. The goal of Bob is to expand G_s by retrieving graph data from FC in an iterative manner until he has sufficient graph data to construct SP(s, t). First, he computes the grid location (e.g., using his GPS) in which his source location s resides, denoted by cg. He also sets the current vertex cv to s. Now, Bob wants to request cg's grid data from FC so that he can expand G_s without revealing any information about cg to C_1 and C_2. A trivial approach here is for Bob to encrypt cg and forward it to C_1, but this would require exponentiation module N^2 operations, which are expensive. To avoid this, Bob splits cg into two random shares cg_1 and cg_2, such that $cg_1 = cg + r \bmod N$ and $cg_2 = N - r$, where $r \in_R \mathbb{Z}_N$. Bob sends cg_1 and cg_2 to C_1 and C_2, respectively.
- Upon receiving cg_2, C_2 encrypts it under pk and forwards $E_{pk}(cg_2)$ to C_1.
- After receiving cg_1 from Alice and $E_{pk}(cg_2)$ from C_2, C_1 computes $E_{pk}(cg)$ by performing homomorphic operations, as $E_{pk}(cg_1) * E_{pk}(cg_2) \bmod N^2$. Then, it obliviously checks which grid's information Bob is requesting. To achieve this, C_1 computes $\Delta_i = E_{pk}(cg) * E_{pk}(i)^{N-1} \bmod N^2$. The idea behind this operation is to subtract Bob's requesting grid ID cg from all grid IDs under encryption. The observation here is that exactly one of the values of Δ is an encryption of 0. C_1 randomizes Δ by computing $X_i = \Delta_i^{r_i} \bmod N^2$, where $r_i \in_R \mathbb{Z}_N$. Then, C_1 randomly permutes $Y \leftarrow \pi(X)$ and sends Y to C_2. Here π is a random permutation function known only to C_1.
- Upon receiving Y, C_2 decrypts it component-wise using the private key sk resulting in a new vector Z. It is worth noting that only one of the entries in Z is 0. Now, C_2 generates a new encrypted vector P based on whether the value of Z_i is 0 or not. Specifically, it generates P as follows and sends it to C_1:

$$P_i = \begin{cases} E_{pk}(1) & \text{if } Z_i = 0 \\ E_{pk}(0) & \text{otherwise} \end{cases}$$

Algorithm 3 SRSP(G', $\langle s, t \rangle$)

Require: Bob holds SSSD query $\langle s, t \rangle$; C_1 holds G' and C_2 holds the private key sk

1: Bob:

(a). $G_s \leftarrow \langle s \rangle$ and $cv \leftarrow s$
(b). Compute the current grid ID cg of cv
(c). Compute two random shares of cg as $cg_1 \leftarrow cg + r \bmod N$ and $cg_2 \leftarrow N - r$, where $r \in_R Z_N$
(d). Send cg_1 to C_1 and cg_2 to C_2

2: C_2:

(a). Receive cg_2 from Bob
(b). Compute and send $E_{pk}(cg_2)$ to C_1

3: C_1:

(a). Receive cg_1 from Bob and $E_{pk}(cg_2)$ from C_2
(b). $E_{pk}(cg) \leftarrow E_{pk}(cg_1) * E_{pk}(cg_2) \bmod N^2$
(c). **for** $1 \leq i \leq n$ **do**:

- $\Delta_i \leftarrow E_{pk}(cg) * E_{pk}(i)^{N-1} \bmod N^2$
- $X_i = \Delta_i^{r_i} \bmod N^2$, where $r_i \in_R \mathbb{Z}_N$

(d). $Y \leftarrow \pi(X)$; Send Y to C_2

4: C_2:

(a). Receive Y from C_1
(b). **for** $1 \leq i \leq n$ **do**:

- $Z_i = D_{sk}(Y)$
- **if** $Z_i = 0$ **then**
  $P_i = E_{pk}(1)$
  else
  $P_i = E_{pk}(0)$

(c). Send P to C_1

5: C_1 :

(a). $Q \leftarrow \pi^{-1}(P)$
(b). **for** $1 \leq i \leq n$ **do**:

- $\Phi_{i,j} \leftarrow \mathrm{SM}(Q_i, G'_{i,j})$, for $1 \leq j \leq 3m + 1$ {SM requires participation of both C_1 and C_2}

(c). **for** $1 \leq j \leq 3m + 1$ **do**

- $\Lambda_j \leftarrow \prod_{i=1}^{n} \Phi_{i,j} \bmod N^2$
- Compute $\Lambda'_j \leftarrow \Lambda_j * E_{pk}(r_j) \bmod N^2$, where $r \in_R Z_N$.
- $\lambda_{1,j} \leftarrow N - r_j$

(d). Send λ_1 to Bob and Λ' to C_2

6: C_2:

(a). $\lambda_{2,j} \leftarrow D_{sk}(\Lambda'_j)$ for $1 \leq j \leq 3m + 1$
(b). Send λ_2 to Bob

7: Bob:

(a). Receive λ_1 from C_1 and λ_2 from C_2
(b). $\lambda_j \leftarrow \lambda_{1,j} + \lambda_{2,j} \bmod N$, for $1 \leq j \leq 3m + 1$
(c). Update G_s based on λ_j and execute Dijkstra's algorithm
(d). **if** t is marked as visited **then**
return SP(s, t)
else
Identify the new neighboring vertex cv and proceed to step 1(b)

- C_1 performs inverse permutation on P to get $Q = \pi^{-1}(P)$. It is worth noting that Q_i equals $E_{pk}(1)$ if $i = cg$, and $Q_i = E_{pk}(0)$ otherwise. After this, C_1 with private input $\langle Q, G' \rangle$ and C_2 with private key sk are involved in a set of secure multiplication operations. Specifically, C_1 with input $\langle Q_i, G'_{i,j} \rangle$ and C_2 jointly execute $\text{SM}(Q_i, G'_{i,j})$, for $1 \leq i \leq n$ and $1 \leq j \leq 3m+1$. Suppose Φ denotes the output of SM. Since Q consists of $E_{pk}(1)$ only for Bob's current grid cg, secure multiplication will result in Φ to store the aggregated grid information of cg. For $i \neq cg$, every other entry in G'_i is multiplied by $E_{pk}(0)$; thus, the result will be encryptions of 0's for all other grids. Note that the output of SM—that is $\Phi_{i,j}$—is known only to C_1, for $1 \leq i \leq n$ and $1 \leq j \leq 3m+1$. Next, C_1 aggregates all the SM results column-wise locally. That is, C_1 computes $\Lambda_j \leftarrow \prod_{i=1}^{n} \Phi_{i,j} \bmod N^2$. The important observation here is that Λ contains the entire grid data in which cg resides. At this point, C_1 needs to somehow send the current grid data to Bob. In order to alleviate the overload on Bob, C_1 utilizes the randomization approach. That is, C_1 selects random numbers $r_j \in_R \mathbb{Z}_N$ and adds it to Λ using additive homomorphic properties by computing $\Lambda'_j \leftarrow \Lambda_j * E_{pk}(r_j) \bmod N^2$. Additionally, C_1 computes $\lambda_{1,j} = N - r_j$. Now, C_1 sends $\lambda_{1,j}$ to Bob and Λ'_j to C_2, for $1 \leq j \leq 3m+1$.
- After receiving the encrypted randomized vector Λ', C_2 decrypts it component-wise using sk to get $\lambda_{2,j}$, for $1 \leq j \leq 3m+1$. Then, C_2 sends λ_2 to Bob. Due to the randomization by C_1, it is worth noting that the decrypted values in this step do not reveal any information to C_2.
- Finally, upon receiving λ_1 and λ_2 from C_1 and C_2, Bob adds them component-wise to get $\lambda_j \leftarrow \lambda_{1,j} + \lambda_{2,j} \bmod N$, for $1 \leq j \leq 3m+1$ which consists of cg's grid data. Bob will then update G_s based on this new grid information and executes the Dijkstra's algorithm to determine the shortest path marking each vertex with minimum distance as visited. If Bob's destination t is in this subgraph and marked as visited, Bob can calculate the shortest distance from s to t locally, and thus terminates the protocol by returning $\text{SP}(s,t)$. Otherwise, Bob identifies the new vertex nv for which the grid information is missing and sets it as the new current vertex cv. Then, the algorithm is repeated (i.e., go to step 1(b) of Algorithm 3) with an updated cv as input to the next iteration.

Example 2. *In this example, we consider a sample weighted graph G (refer to Figure 2) and illustrate various intermediate steps during the execution of ESPADE. Here G consists of 16 vertices, denoted from A to P, and it is split into four evenly distributed square grids. The four grid cells are denoted by $M_1, \ldots, M_4$. Without the loss of generality, suppose Bob wants to retrieve the shortest path from A to P using ESPADE. Various intermediate results along with the shortest path discovery process in each iteration are shown in Figure 3. For brevity, we only show the steps involved during Stage 2 of ESPADE.*

- ***Iteration 1**: Initially, Bob sets his current vertex cv to A. In this case, the current grid ID of cv is 1 since cv resides in M_1. That is, Bob sets $cg = 1$. Bob randomly splits his cg information and sends them to C_1 and C_2, separately. At the end of the first iteration, Bob receives all the vertex and associated edge weight information of M_1. He updates his sub-graph G_s (refer to Figure 3a) and executes Dijkstra's algorithm. After marking C as visited, Bob makes I the current vertex.*
- ***Iteration 2**: Bob updates the current grid ID cg value to 3, as vertex I resides in M_3. cg is passed as input to the second iteration. At the end of the second iteration, Bob expands G_s as shown in Figure 3b.*
  - *Similarly, Bob retrieves M_2 and M_4 information in iterations 3 and 4, respectively. Refer to Figure 3c,d. At the end, Bob finds out that $SP(A,P) = \{A, C, D, G, M, P\}$.*

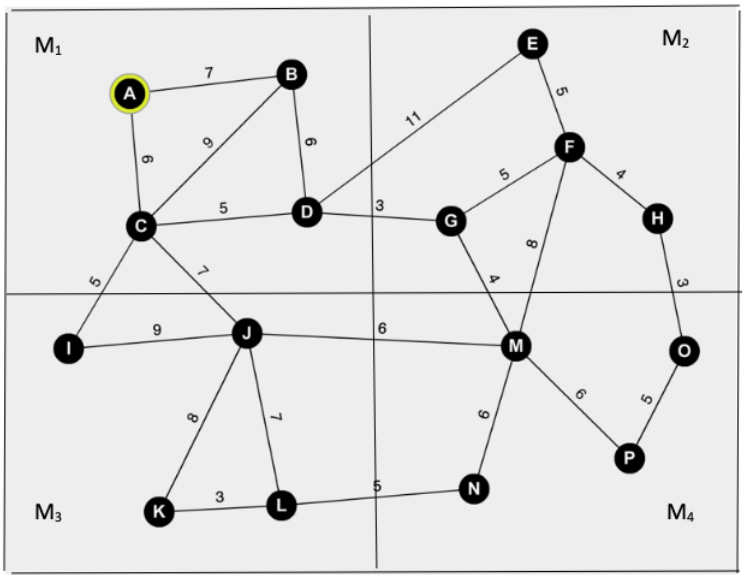
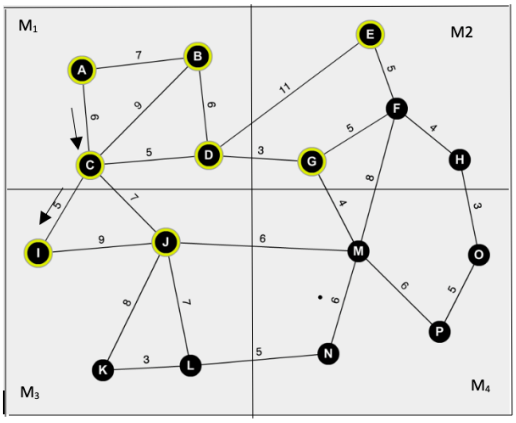
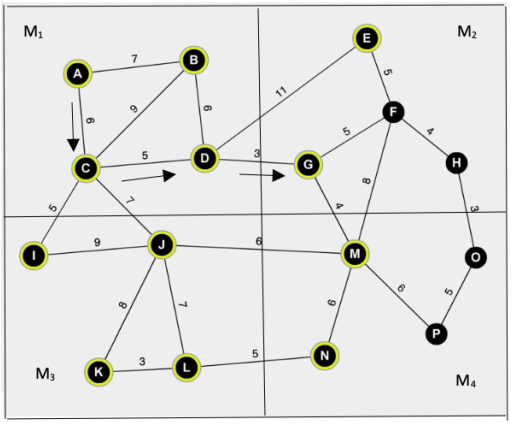


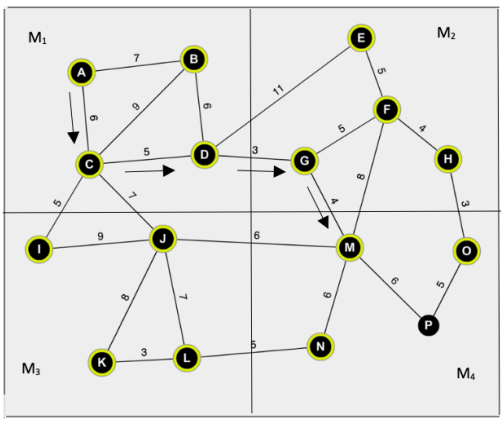
Figure 2. A snapshot of geospatial network G with 16 vertices which are divided into four square grids.



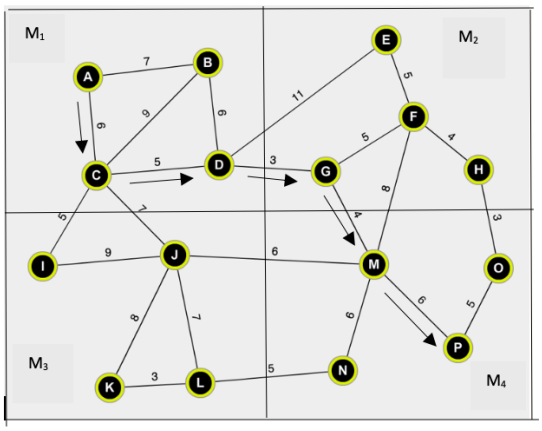
(**a**) Iteration 1 with $cv = A$ and $cg = 1$



(**b**) Iteration 2 with $cv = I$ and $cg = 3$



(**c**) Iteration 3 with $cv = G$ and $cg = 2$



(**d**) Iteration 4 with $cv = M$ and $cg = 4$

Figure 3. Shortest path exploration and intermediate results during the execution of proposed ESPADE protocol.

6. Performance Analysis of ESPADE

In this section, we present the security and performance analysis of the ESPADE protocol. First, we formally show that ESPADE is secure under the semi-honest model of SMC. Then, we discuss the complexity costs of our protocol and provide a comparative analysis with two existing solutions including experimental results.

6.1. Security Analysis under the Semi-Honest Model

As mentioned earlier, we assume that all the parties participating in our protocol are semi-honest as it is the commonly used adversarial model. Under the semi-honest model, parties follow the prescribed steps of the protocol, but they are free to deduce any meaningful information about the inputs based on the messages they see during the execution of the protocol. In this paper, we adopt the following security definition that is commonly used for the semi-honest model [19,35]:

Definition 1. *Suppose P be a party participating in a protocol π with input a and output b. Assume that $\Pi_{P,real}(\pi)$ denotes the P's execution image of π. In general, an execution image of P consists of all the inputs, outputs, and the intermediate messages it sees during the execution of the protocol. Let $\Pi_{P,sim}(\pi)$ denote the P's simulated image based on π and $\langle a, b\rangle$. Then, π is secure if the (real) execution image of P is computationally indistinguishable from its simulated image.*

6.1.1. Proof of Security for Stage 1

During Stage 1, the execution image of C_2 is given by $\Pi_{C_2,real}(\text{ESPADE}) = \{\langle T^2_{i,j}, F_{i,j}\rangle, \text{for } 1 \leq i \leq n \wedge 1 \leq j \leq 3m+1\}$. Here $T^2_{i,j}$ is the encrypted value received from Alice at step 2(a) of Algorithm 2. Additionally, $F_{i,j}$ is a random number derived upon decrypting $T^2_{i,j}$. Without loss of generality, let the simulated image of C_2 be given as $\Pi_{C_2,sim}(\text{ESPADE}) = \{\langle a_{i,j}, a'_{i,j}\rangle, \text{for } 1 \leq i \leq n \wedge 1 \leq j \leq 3m+1\}$, where $a_{i,j}$ and $a'_{i,j}$ are randomly chosen from $\mathbb{Z}_{N^2}$ and $\mathbb{Z}_N$, respectively. Since the encryption function of Paillier's scheme is semantically secure, it is guaranteed that $T^2_{i,j}$ will be a random number in $\mathbb{Z}_{N^2}$. Thus, $T^2_{i,j}$ is indistinguishable from $a_{i,j}$. Similarly, $F_{i,j}$ is distinguishable from $a'_{i,j}$. As a result, $\Pi_{C_2,sim}(\text{ESPADE})$ is computationally indistinguishable from $\Pi_{C_2,real}(\text{ESPADE})$, which implies that C_2 does not learn anything during the execution of Stage 1.

On the other hand, the execution image of C_1 in Stage 1 is given by $\Pi_{C_1,real}(\text{ESPADE}) = \{T^2\}$, where T^2 is an encrypted matrix sent by C_2 (at step 2(c) of Algorithm 2). Let the simulated image of C_1 be $\Pi_{C_1,sim}(\text{ESPADE}) = \{B\}$, where the values of $B_{i,j}$ are randomly selected from $\mathbb{Z}_{N^2}$. Since E_{pk} is a semantically secure encryption function, it guarantees that ciphertexts $T^2_{i,j}$ are randomly distributed in $\mathbb{Z}_{N^2}$. This shows that $T_{i,j}$ is computationally indistinguishable from $B_{i,j}$; thus, $\Pi_{C_1,sim}(\text{ESPADE})$ is computationally indistinguishable from $\Pi_{C_1,real}(\text{ESPADE})$. Therefore, C_1 cannot deduce any information about G in Stage 1.

6.1.2. Proof of Security for Stage 2

In this sub-section, we formally prove that Stage 2 of ESPADE is secure as per Definition 1. Without the loss of generality, we consider the messages exchanged between C_1 and C_2 in a single iteration. We emphasize that similar analyses can be carried out for other iterations.

During Stage 2, the execution image of C_2 is given by,

$$\Pi_{C_2,real}(\text{ESPADE}) = \{cg2, \langle Y, Z\rangle, \langle \Lambda', \gamma_2\rangle\}$$

where cg_2 is a randomized value sent by Bob (at step 1(d) of Algorithm 3). Additionally, Y is an encrypted vector sent by C_1 at step 3(d) and Z is the resulting decrypted vector, such that exactly one of the entries is 0 and all others values are random numbers in $\mathbb{Z}_N$. Similarly, Λ' is the encrypted randomized vector sent by C_1 at step 5(d) and γ_2 is the resulting decrypted vector. Without loss of generality, let the simulated image of C_2 be given by,

$$\Pi_{C_2,sim}(\text{ESPADE}) = \{h, \langle I, J\rangle, \langle K, L\rangle\}$$

Here h denotes a random number generated from $\mathbb{Z}_N$. I denotes a vector of size $3m+1$ whose elements are randomly selected from $\mathbb{Z}_{N^2}$ whereas vector J is randomly generated such that only one of the entries is 0 and the remaining entries are random numbers in $\mathbb{Z}_N$. Similarly, K and L denote vectors of size $3m+1$ whose elements are randomly selected from $\mathbb{Z}_{N^2}$ and $\mathbb{Z}_N$, respectively. Since cg_2 and h are both random numbers from $\mathbb{Z}_{N^2}$, it is evident that they are computationally indistinguishable. Since E_{pk} generates ciphertexts that are uniformly random in $\mathbb{Z}_{N^2}$, we conclude that I and K are computationally indistinguishable from Y and Λ', respectively. Plus, J is computationally indistinguishable from Z as they both have exactly one of the entries as 0 and remain as random numbers in $\mathbb{Z}_N$. Furthermore, L and γ_2 are vectors consisting of random numbers chosen from $\mathbb{Z}_N$; thus, they are computationally indistinguishable. Based on the above results, it is implied that C_2 cannot deduce any information about G and Bob's query during Stage 2.

Similarly, according to Algorithm 3, we can show that C_1's execution image can be simulated from random numbers. The important observation here is that all the messages received by C_1 (from Bob and C_2) are either in an encrypted format or are randomized numbers distributed in $\mathbb{Z}_N$. Therefore, no information is revealed to C_1.

Following from Algorithm 2, we emphasize that ESPADE is constructed by sequentially combining the two stages. As shown above, Stages 1 and 2 are secure under the semi-honest model. Additionally, it is worth noting that the output of Stage 1 (which is in encrypted format) is passed as an input to Stage 2. Therefore, by Composition Theorem [35], we can conclude that ESPADE is secure under the semi-honest adversary model. That is, ESPADE ensures that neither the contents of G nor the shortest path query is revealed to C_1 and C_2. Therefore, our proposed protocol meets all the three privacy objectives (i.e., PO1, PO2, and PO3), described in Section 1.

6.2. Complexity Analysis

In this sub-section, we analyze the computation, communication and round complexities of the proposed ESPADE protocol. Table 2 shows the complexity costs incurred for various participating parties during the execution of our protocol.

6.2.1. Computation Costs

In our problem setting, we explicitly assume that the value of generator g is set to $N+1$ which helps us to optimize the Paillier encryption function without affecting the underlying security guarantees. Specifically, when $g = N+1$, Paillier's encryption function can be reduced to $E_{pk}(m) = (1 + m * N) * r^N \bmod N^2$, for any message $m \in \mathbb{Z}_N$. Additionally, it is worth noting that the computation of $r^N \bmod N^2$ can be done offline, since it is independent of the message to be encrypted. As a result, the actual online computation cost of Paillier encryption is two multiplications (under modulo N^2). We refer the reader to [36] for detailed security analysis on this setting. Following from the above optimizations, we only consider the online computation costs during our complexity analyses of ESPADE.

Table 2. Complexity Results: Computation, Communication and Round Costs for different parties in ESPADE.

	Online Computation	**Communication (in bits)**	**Round**
Alice (one-time)	$(6m+2) * n$ additions	$2n * (3m+1) * \log N$	-
Bob	$O(mn)$ additions	$O(n \log N)$	$O(n)$
Federated Cloud	$O(mn^2 \log N)$ multiplications	$O(mn^2 \log N)$	$O(n)$

First, during step1(a) of Stage 1, the computation cost of Alice is bounded by $(6m+2) * n$ addition operations. Note that Alice does not participate in any other operations after outsourcing the data

to FC. It is evident that Alice's computation costs are negligible as it is a one-time cost. Similarly, the computation cost of Bob mainly depends on steps 1 and 7 in Algorithm 3. In each iteration, Bob needs to perform $O(m)$ additions. Since the number of iterations is bounded by $O(n)$, the total computation cost of Bob in ESPADE is bounded by $O(nm)$ additions (which is low compared to the overall computation cost of the protocol as shown below).

Next, we discuss the computation costs incurred on the federated cloud. During Stage 1, at step 2(b) of Algorithm 2, C_2 is involved in $n * (3m + 1)$ encryption operations, which is equivalent to $2n * (3m + 1)$ multiplication operations. Here n denotes the total number grids and m denotes the maximum number of 1-hop neighbors, a vertex can have in G. Additionally, at step 3(b), C_1 performs $n * (3m + 1)$ multiplications. Combining these results, the total computation cost of FC in Stage 1 is bounded by $O(mn^2)$ multiplications.

During Stage 2, the computation cost of FC mainly depends on steps 3, 4, and 5 of Algorithm 3. For steps 3 and 5, C_1 needs to perform $(n \log N)$ and $(mn \log N)$ multiplications, respectively. For step 4, C_2 is involved in $\log N$ multiplications. Therefore, the total computation cost of FC in stage 2 is bounded by $O(mn \log N)$ multiplications. Putting everything together, the total computation cost of FC (i.e., the combined computation costs of C_1 and C_2) in ESPADE is bounded by $O(mn^2 \log N)$ multiplications.

6.2.2. Communication and Round Complexity

One the one hand, for Alice, the communication cost depends on step 1(c) of Algorithm 2, where it sends out two matrices T^1 and T^2 to FC. The size of these matrices is $n \times (3m + 1)$. Since each entry in these matrices is a random number chosen from $\mathbb{Z}_N$, the total size of each matrix is $n * (3m + 1) * \log N$ bits. Therefore, the total communication cost of Alice is $(2n * (3m + 1) * \log N)$ bits. On the other hand, in each iteration of Stage 2, Bob splits the cg value into two random shares and forwards them to FC. This results in $2 \log N$ bits of communication. Since the number of total iterations is bounded by $O(n)$, the total communication cost of Bob is bounded by $O(n \log N)$ bits. On the other hand, the total communication cost of FC is bounded by $O(mn^2 \log N)$ bits.

The number of communication rounds between Bob and FC is bounded by $O(n)$. Furthermore, the number of communication rounds between C_1 and C_2 is bounded by $O(n)$. Therefore, the round complexity of ESPADE is bounded by $O(n)$.

6.3. Performance Comparison with Existing Work

In this sub-section, we compare the performance of ESPADE with two closely related works, namely PSPEG_1 and PSPEG_2 [14]. The performance comparison results are shown in Table 3.

Table 3. Performance comparison of ESPADE with PSPEG_1 and PSPEG_2.

Features	**PSPEG_1**	**PSPEG_2**	**ESPADE**
Cloud Model	Single-Cloud	Two-Cloud	Two-Cloud
Shortest-Path Accuracy	✓	✓	✓
Proof of Security	✗	✗	✓
Data Outsourcing Cost	$O(\ell mn \log N)$ mul.	$O(m\|V\| \log N)$ mul.	$\boldsymbol{O(mn)}$ add.
Bob's Computation Cost	$O(n^2 \log N')$ mul.	$O(\|V\| \log \|V\|)$ add.	$\boldsymbol{O(mn)}$ add.
Total Computation Cost	$O(mn^2 \log N')$ mul.	$O(m\|V\|^2 \log N)$ mul.	$\boldsymbol{O(mn^2 \log N)}$ mul.
Total Communication Cost	$O(mn^2 \log N')$ bits	$O(m\|V\|^2 \log N)$ bits	$\boldsymbol{O(mn^2 \log N)}$ bits
Total Round Complexity	$\boldsymbol{O(n)}$	$O(\|V\|)$	$\boldsymbol{O(n)}$

On the one hand, PSPEG_1 is based on a single-cloud architecture whereas PSPEG_2 and ESPADE adopt a two-cloud federated model. Additionally, we observe that PSPEG_1, PSPEG_2 and ESPADE

always produce correct results as the underlying operations in all the three protocols are constructed based on Dijkstra's algorithm. On the other hand, the security guarantees of PSPEG_1 and PSPEG_2 are unclear as no formal proofs were provided to demonstrate the confidentiality of the outsourced data, privacy of the user's shortest path query and the protection of access patterns. In our case, as we formally showed in Section 6.1, ESPADE is semantically secure under the semi-honest model; thus, it meets all the three privacy criteria. Additionally, it is worth noting that ESPADE hides data access patterns due to the underlying random permutation operations.

We observe that PSPEG_1 is very inefficient in terms of computation and communication complexities. Specifically, since PSEPG_1 utilizes a single-cloud model, it incurs significant costs on the data owner Alice and the end-user Bob. For the data outsourcing step, the computation costs of Alice in PSPEG_1 and PSPEG_2 are bounded by $O(\ell mn \log N)$ and $O(m|V| \log N)$ multiplications, respectively, where n denotes the number of grids and $|V|$ denotes the number of vertices in G. Unlike PSEPG_1 and PSEPG_2, our proposed ESPADE protocol utilizes a random splitting approach during the data outsourcing step, which incurs low costs on Alice. In particular, the computation cost of Alice is bounded by $O(mn)$ additions. It is clear that the computation cost of Alice is significantly less in ESPADE in comparison with PSEPG_1 and PSEPG_2.

For PSEPG_1, the computation cost of Bob is bounded by $O(n^2 \log N')$ multiplications, where N' denotes the RSA moduli, such that $N^2 < N'$. In PSPEG_2, this computation burden is alleviated to a certain extent by pushing some expensive computations from Bob's side to the second cloud. Nonetheless, the computation cost incurred on bob is bounded by $O(|V| \log |V|)$ which is still high even for moderate-sized graphs ($|V| \approx 10,000$). In ESPADE, the computation cost of Bob is bounded by $O(mn)$ additions. It is worth noting that, with the effective combination of homomorphic encryption properties and our secure data aggregation technique, ESPADE significantly improves the computation cost of Bob. As shown in Table 3, ESPADE incurs overall less computation and communication costs compared to PSEPG_1 and PSEPG_2, especially for $n < |V|$. Furthermore, the round complexities of PSEPG_1 and ESPADE are bounded by $O(n)$, whereas for PSEPG_2 it is bounded by $O(|V|)$.

6.4. Experimental Results

In this sub-section, we demonstrate the superiority of ESPADE over PSPEG_1 and PSPEG_2 through empirical analysis. All three protocols were implemented in Java using the BigInteger Class to handle arbitrary-precision arithmetic operations, and experiments were conducted on a Intel® Core™ i7 3.1 GHz PC running macOS 10.13.6 High Sierra with 16GB memory. The Paillier encryption key size is set to 1024 bits (i.e., the size of N in bits is 1024) and all the results presented are average values over five executions.

In our experiments, randomly generated datasets were used. For $\ell = 20, m = 20, n = 100, |V| = 5000$, the running time for Alice in PSPEG_1 and PSPEG_2 are 4.11 and 6.81 min, respectively, whereas the running time for Alice in ESPADE is 21 milliseconds. Note that Alice needs to perform expensive exponentiation operations to encrypt the matrix data in PSPEG_1 and PSPEG_2, whereas in ESPADE Alice simply involves in modulo addition operations. Additionally, the running time for Bob in PSPEG_1 and PSPEG_2 are 1.02 min and 121 milliseconds, respectively. The running time for Bob in ESPADE is 7 milliseconds. The above results clearly justify our performance analysis in Section 6.3 and show that ESPADE is significantly more efficient, by several orders of magnitude, than PSPEG_1 and PSPEG_2.

In summary, by using the proposed data aggregation approach in ESPADE, Alice can securely outsource her graph data and effectively delegate the shortest path query processing task to Federated Cloud (FC). Furthermore, the costs incurred on Bob during Stage 2 of ESPADE are minimal. Based on the above discussions, we conclude that ESPADE significantly offers improved performance in computation and communication load over PSPEG_1 and PSPEG_2, and at the same time, offering a higher level of security protection.

7. Conclusions

Existing research shows that location-based services violate user's privacy and the issue becomes even more challenging when such applications are pushed to remote and non-trusted cloud servers. In this paper, we addressed the single-source single-destination shortest path query processing problem in outsourced LBS. Specifically, we proposed an efficient and semantically secure shortest path discovery protocol for encrypted graph data outsourced to a federated cloud environment. At the core of our proposed ESPADE protocol, we utilized homomorphic encryption combined with a novel data aggregation technique to enable the cloud service providers to operate over encrypted aggregated data in a privacy-preserving manner. We formally showed that our protocol is secure under the semi-honest model and also hides access patterns. Additionally, we discussed the complexity analysis of ESPADE and demonstrated that it is more efficient and secure compared to PSPEG_1 and PSPEG_2. Our experimental results show that ESPADE is significantly faster than the existing solutions.

Improving the performance of ESPADE further largely depends on minimizing the amount of data sent from the federated cloud to Bob. Additionally, improving the performance of the secure multiplication protocol is another important step to improve the overall efficiency of ESPADE. For future work, we will investigate better data aggregation and pruning techniques to enhance the secure retrieval of the shortest path (Stage 2) process in ESPADE. We will also extend our research to other graph mining tasks, such as minimum spanning tree and breadth-first search, over encrypted graph data. Another direction for future work is to extend the ESPADE protocol into a secure protocol under other adversarial models (e.g., covert and malicious models).

Author Contributions: Conceptualization, B.K.S. and D.K.; methodology, B.K.S. and D.K.; software, D.K. and B.K.S.; validation, B.K.S., D.K., B.D. and A.K.K.; formal analysis, B.K.S.; investigation, B.K.S. and D.K.; resources, B.K.S. and D.K.; data curation, B.K.S., and D.K.; writing—original draft preparation, B.K.S. and D.K.; writing—review and editing, B.K.S., D.K., B.D. and A.K.K.; visualization, B.K.S. and D.K.; supervision, B.K.S.; project administration, B.K.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bennett, J.; Rokas, O.; Chen, L. Healthcare in the Smart Home: A Study of Past, Present and Future. *Sustainability* **2017**, *9*, 840. [CrossRef]
2. Islam, S.M.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access* **2015**, *3*, 678–708. [CrossRef]
3. Jeong, S.; Kim, W.J.; Cho, S. Internet of Things for Smart Manufacturing System: Trust Issues in Resource Allocation. *IEEE Internet Things J.* **2018**, *5*, 4418–4427. [CrossRef]
4. Fraga-Lamas, P.; Fernández-Caramés, T.M.; Suárez-Albela, M.; Castedo, L.; González-López, M. A Review on Internet of Things for Defense and Public Safety. *Sensors* **2016**, *16*, 1644. [CrossRef] [PubMed]
5. Huang, H.; Gartner, G.; Krisp, J.M.; Raubal, M.; Weghe, N.V. Location based services: Ongoing evolution and research agenda. *J. Locat. Based Serv.* **2018**, *12*, 63–93.
6. Junglas, I.A.; Watson, R.T. Location-based services. *Commun. ACM* **2008**, *51*, 65–69. [CrossRef]
7. Perusco, L.; Michael, K. Control, trust, privacy, and security: Evaluating location-based services. *IEEE Technol. Soc. Mag.* **2007**, *26*, 4–16. [CrossRef]
8. Asuquo, P.; Cruickshank, H.; Morley, J.; Ogah, C.; Lei, A.; Hathal, W.; Bao, S.; Sun, Z. Security and Privacy in Location-Based Services for Vehicular and Mobile Communications: An Overview, Challenges, and Countermeasures. *IEEE Internet Things J.* **2018**, *5*, 4778–4802. [CrossRef]
9. Rathod A.; Jariwala, V. Investigation of Privacy Issues in Location-Based Services. In *Recent Findings in Intelligent Computing Techniques*; Sa, P., Bakshi, S., Hatzilygeroudis, I., Sahoo, M., Eds.; Springer: Singapore, 2019; Volume 707, pp. 55–65.
10. Bokhari, M.U.; Makki, Q.; Tamandani, Y.K. A Survey on Cloud Computing. In *Big Data Analytics*; Aggarwal, V., Bhatnagar, V., Mishra, D., Eds.; Springer: Singapore, 2018; Volume 654, pp. 149–164.

11. Studedi, P.; Mohomed, I.; Terry, D. WhereStore: Location-based data storage for mobile devices interacting with the cloud. In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, San Francisco, CA, USA, 15–18 June 2010; ACM: New York, NY, USA, 2010; pp. 1–8.
12. Ghinita, G.; Kalnis, P.; Khoshgozaran, A.; Shahabi, C.; Tan, K.-L. Private queries in location based services: Anonymizers are not necessary. In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, Vancouver, BC, Canada, 10–12 June 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 121–132.
13. Yi, X.; Paulet, R.; Bertino, E.; Varadharajan, V. Practical k nearest neighbor queries with location privacy. In Proceedings of the IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 31 March–4 April 2014; pp. 640–651.
14. Samanthula, B.K.; Rao, F.; Bertino, E.; Yi, X. Privacy-Preserving Protocols for Shortest Path Discovery over Outsourced Encrypted Graph Data. In Proceedings of the IEEE International Conference on Information Reuse and Integration, San Francisco, CA, USA, 13–15 August 2015; pp. 427–434.
15. Blanton, M.; Steele, A.; Alisagari, M. Data-oblivious graph algorithms for secure computation and outsourcing. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, Hangzhou, China, 7–10 May 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 207–218.
16. Zhang, L.; Li, J.; Yang, S.; Wang, B. Privacy Preserving in Cloud Environment for Obstructed Shortest Path Query. *Wirel. Pers. Commun.* **2020**, *96*, 2305–2322. [CrossRef]
17. Paillier, P. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology—EUROCRYPT*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1592, pp. 223–238.
18. Goldwasser, S.; Micali, S.; Rackoff, C. The knowledge complexity of interactive proof-systems. In Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing, Providence, RI, USA, 6–8 May 1985; Association for Computing Machinery: New York, NY, USA, 1985; pp. 291–304.
19. Goldreich, O. General Cryptographic Protocols. In *Foundations of Cryptography*; Cambridge University Press: Cambridge, UK, 2004; Volume 2, pp. 599–746.
20. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*; MIT Press: Cambridge, MA, USA. 2009; pp. 658–663.
21. Bugiel, S.; Nürnberger, S.; Sadeghi, A.R.; Schneider, T. Twin Clouds: Secure Cloud Computing with Low Latency. In *Communications and Multimedia Security*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7025, pp. 32–44.
22. Wang, B.; Li, M.; Chow, S.M.; Li, H. Computing encrypted cloud data efficiently under multiple keys. In Proceedings of the 2013 IEEE Conference on Communications and Network Security, National Harbor, MD, USA, 21–23 October 2013; pp. 504–513.
23. Samanthula, B.K.; Elmehdwi, Y.; Jiang, W. k-Nearest Neighbor Classification over Semantically Secure Encrypted Relational Data. *IEEE Trans. Knowl. Data Eng.* **2015**, 27, 1261–1273. [CrossRef]
24. Samanthula, B.K.; Albehairi, S.; Dong, B. A Privacy-Preserving Framework for Collaborative Association Rule Mining in Cloud. In Proceedings of the IEEE Cloud Summit, Washington, DC, USA, 8–10 August 2019; pp. 116–121.
25. Barak, B.; Goldreich, O.; Impagliazzo, R.; Rudich, S.; Sahai, A.; Vadhan, S.P.; Yang, K. On the (im)possibility of obfuscating programs. *J. ACM* **2012**, *59*, 6. [CrossRef]
26. Lee, K.C.K.; Lee, W.-C.; Leong, H.; Zheng, B. Navigational path privacy protection: Navigational path privacy protection. In Proceedings of the 18th ACM Conference on Information and Knowledge Management, Hong Kong, China, 2–6 November 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 691–700.
27. Ostrovsky, R.; Skeith, W.E. A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In *Public Key Cryptography*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4450, pp. 393–411.
28. Kushilevitz, E.; Ostrovsky, R. Replication is not needed: Single database, computationally-private information retrieval. In Proceedings of the 38th Annual Symposium on Foundations of Computer Science, Miami Beach, FL, USA, 19–22 October 1997; pp. 364–373.

29. Mouratidis, K.; Yiu, M.L. Shortest path computation with no information leakage. *Proc. VLDB Endow.* **2012**, *5*, 692–703. [CrossRef]
30. Mehrotra, S.; Sharma, S.; Ullman, J.D.; Ghosh, D.; Gupta, P.; Mishra, A. Panda: Partitioned Data Security on Outsourced Sensitive and Non-sensitive Data. *arXiv* **2020**, arXiv:2005.06154.
31. Li, L.; Lu, R.; Huang, C. EPLQ: Efficient Privacy-Preserving Location-Based Query over Outsourced Encrypted Data. *IEEE Internet Things J.* **2015**, *3*, 206–218. [CrossRef]
32. Zhu, X.; Ayday, E.; Vitenberg, R. A privacy-preserving framework for outsourcing location-based services to the cloud. *IEEE Trans. Dependable Secur. Comput.* **2019**. [CrossRef]
33. Acar, A.; Aksu, H.; Uluagac, A.S.; Conti, M. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* **2018**, *51*, 35p. [CrossRef]
34. Elmehdwi, Y.; Samanthula B.K.; Jiang, W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In Proceedings of the IEEE 30th International Conference on Data Engineering, Chicago, IL, USA, 31 March–4 April 2014; pp. 664–675.
35. Goldreich, O. Encryption Schemes. In *Foundations of Cryptography*; Cambridge University Press: Cambridge, UK, 2004; Volume 2, pp. 373–470.
36. Damgård, I.; Jurik, M. A Generalisation, a Simpli.cation and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography* Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 1992, pp. 119–136.