electronics

MDPI

Article

Location-Visiting Characteristics Based Privacy Protection of Sensitive Relationships

Xiu-Feng Xia *, Miao Jiang *, Xiang-Yu Liu and Chuan-Yu Zong

School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China; liuxy@sau.edu.cn (X.-Y.L.); zongcy@sau.edu.cn (C.-Y.Z.)

* Correspondence: xiufengxia@mail.sau.edu.cn (X.-F.X.); jiangmiao1997@163.com (M.J.)

Abstract: In the era of Internet of Things (IoT), the problem of the privacy leakage of sensitive relationships is critical. This problem is caused by the spatial–temporal correlation between users in location-based social networks (LBSNs). To solve this problem, a sensitive relationship-protection algorithm based on location-visiting characteristics is proposed in this paper. Firstly, a new model based on location-visiting characteristics is proposed for calculating the similarity between users, which evaluates check-in features of users and locations. In order to avoid an adversary inferring sensitive relationship privacy and to ensure the utility of data, our proposed algorithm adopts a heuristic rule to evaluate the impact of deduction contributions and information loss caused by data modifications. In addition, location-search technology is proposed to improve the algorithm's execution efficiency. The experimental results show that our proposed algorithm can effectively protect the privacy of sensitive data.

Keywords: social networks; privacy protection; sensitive relationships; location-visiting characteristics; data utility

Citation: Xia, X.-F.; Jiang, M.; Liu, X.-Y.; Zong, C.Y. Location-Visiting Characteristics Based Privacy Protection of Sensitive Relationships. *Electronics* **2022**, *11*, 1214. https://doi.org/10.3390/electronics11081214

Academic Editor: Tiziana Margaria

Received: 11 February 2022
Accepted: 6 April 2022
Published: 12 April 2022

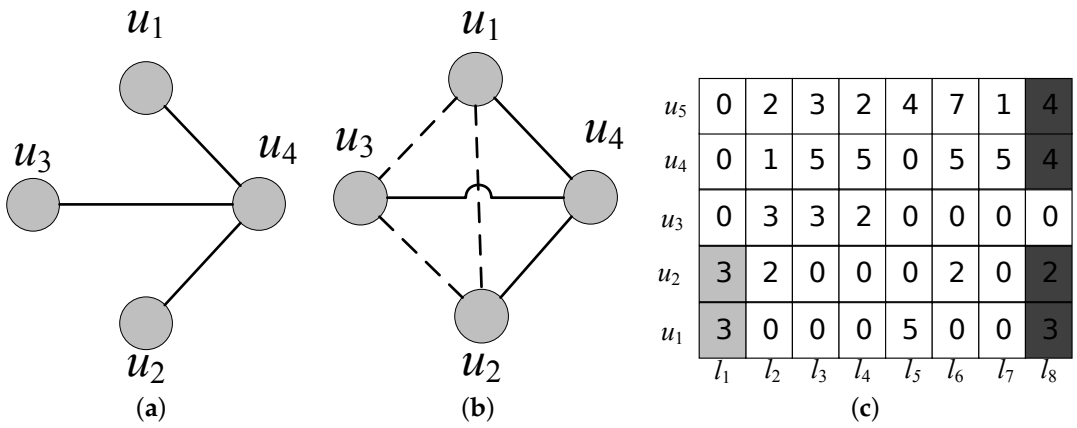
1. Introduction

With the development of critical technologies in the Internet of Things (IoT), IoT applications have become widely used throughout the world. As a result, human behavior has been extended by the development of technology [1]. Along with the rapid growth of IoT applications and devices, data security and privacy has become a critical issue due to the existence of many attackers. In addition, such attacks will become faster and more complicated, and consequently, potential threats will increase [2,3]. IoT exchanges, communicates and manages data from GPS, RFID and other information-sensing devices. Among these, GPS check-in data contain users' spatiotemporal information, leading to a high risk of privacy leakage.

IoT joins devices of varying strengths [4]. Environment sensors or smart hand-held devices are generating data at an unprecedented rate within the era of an Internet of Things (IoT)-driven world [5]. Many social networks (for example, Gowalla and Brightkite applications) support check-in services, and check-in data contain the spatiotemporal characteristics of the user [6]. A sensitive relationship mainly refers to a relationship between two users that is unwilling to be known by other people. Users with sensitive relationships usually have similar spatiotemporal behavior due to intimacy or similar interests, which makes two users show spatiotemporal correlation and similarity in check-in data. Therefore, attackers can deduce sensitive relationships based on background knowledge and related technologies [7]. For example, based on the background knowledge of the structure of the published social network graph and the check-in data of a user's corresponding node, attack technology with the co-check-in model [8] and EBM [9] models can deduce sensitive relationships with spatiotemporal correlation. It is insufficient to protect privacy information by simply removing the sensitive edges of two users in a social network before publishing data.

The motivation of this research is to offer a method for sensitive-relationship privacy protection by modifying operations on social network and spatiotemporal data with a novel protection model. In general methods, some researchers focus on the co-check-in model, which takes the proportion of co-check-ins in a check-in set it as the protection key and proposes a protection method via check-in suppression. However, this method does not fully consider the impact of the check-in location on sensitive relationships, which leads to high information loss and the risk of disclosing protection results.

Taking the example in Figure 1a, $\langle u_1, u_2\rangle$, $\langle u_1, u_3\rangle$ and $\langle u_2, u_3\rangle$ are regarded as sensitive relationships. Under the co-check-in based model, $\langle u_1, u_2\rangle$ and $\langle u_2, u_3\rangle$ may become inferred relationships that need protection because their corresponding check-ins have the same location. In addition, as shown in Figure 1b, $\langle u_1, u_3\rangle$ and $\langle u_2, u_3\rangle$ have the same number of check-ins. Therefore, they have the same probability of having an "existing sensitive relationship". However, according to Figure 1c, it is obvious that there are few visitors appearing at location l_5, which means l_5 has a high degree of privacy, while l_2 is a popular place, such as a subway station or a supermarket. This means users with check-ins at l_2 have a higher probability of having relationships with each other, but current methods do not fully consider this.



	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_8
u_5	0	2	3	2	4	7	1	4
u_4	0	1	5	5	0	5	5	4
u_3	0	3	3	2	0	0	0	0
u_2	3	2	0	0	0	2	0	2
u_1	3	0	0	0	5	0	0	3

Figure 1. Social network and user location check-in record graph: (**a**) G; (**b**) G'; (**c**) User location check-in distribution diagram.

For example, l_1 has a higher privacy degree than l_8, and check-ins of u_1 and u_2 at l_1 have a greater contribution than at other locations. Therefore, compared with removing u_2's check-in at l_8, removing u_2's check-in at l_1 is more reasonable. However, state-of-the-art efforts do not fully consider the impact of the degree of location privacy on the algorithm's performance, which means that check-ins at l_1 and l_8 have the same contribution to sensitive-relationship processing. In this way, if the check-ins at l_8 and l_1, which contribute little to sensitive relationships, are removed, more information may be lost. Secondly, omitting the check-in at l_1 leads to sensitive-relationship leakage. It can be seen that this protection model has defects in deducing the relationship between users.

To solve the above problems, this paper proposes a sensitive-relationship protection model based on location-visiting characteristics. The basic idea is to evaluate the privacy degree of each location l_i according to the number of users that visit l_i. In this way, our proposed model can obtain a vector of users' location-visiting characteristics and can combine the number of users' check-ins into an evaluation of the sensitive relationship. We further propose a sensitive-relationship privacy-protection method based on location-visiting characteristics. It removes and adds check-in data to prevent the disclosure of sensitive relationships based on our proposed model. It can evaluate which check-ins could be added (or removed) based on their importance. In this way, the problems of sensitive-relationship privacy leakage and low data availability can be solved. Above all, the contributions of this paper are as follows:

1. We define a sensitive-relationship privacy-protection model based on location-visiting characteristics. It considers the impact of privacy degree at different check-in locations on sensitive relationships;
2. We propose a sensitive-relationship protection algorithm based on users' visiting characteristics to protect sensitive relationships. It considers both information loss and privacy leakage via a heuristic rule;
3. We design a check-in location search technology to make our privacy-protection algorithm more efficient, meanwhile preserving both trajectory utility and its original shape.

The rest of the paper is organized as follows. Related works are outlined in Section 2. Preliminary studies and problem definition are proposed in Section 3. Section 4 discusses our algorithm in detail. The experimental results and performance of the proposed method are given in Section 5. Section 6 concludes the article with future directions of research.

Problem Statement:Given social network graph $G(V, E, C)$, sensitive edge set S, and user similarity threshold α, our target is to find an inference secure graph PG with sensitive-relationship privacy protection on the premise of minimal information loss.

2. Related Work

At present, research on the privacy protection of user information and user relationship information is divided into several aspects. The authors in [10,11] study the defense strategy of link inference attacks for graph-structured data. However, these studies ignore the privacy leakage of sensitive relationships caused by spatiotemporal information in location-based social networks. Some other studies use relationship information to protect users' location privacy in location-based social networks. Chen Weihe et al. [12] proposed the L-intimacy privacy protection model, which protects users' privacy according to their intimacy level with friends. Alrayes et al. [13] analyzed the possible privacy threats caused by users sharing location over LBSN and designed a feedback system based on privacy threat level. Ahuja et al. [14] focused on the common location privacy in social networks. They proposed a common location protection model to prevent attackers from users' common location based on social relationships and designed a privacy-protection mechanism. Shirani et al. [15] proposed an extensible framework named PLACE and proposed three novel privacy-protection bases, including location proximity, co-occurrence vector and following degree.

Pham et al. [9] proposed a Shannon-entropy-based location model (EBM) to measure the popularity of the location and then used this model to measure the influence of position entropy on relationship deduction. In [16], the authors proposed a time–place-tracking model (TLFM) based on visiting time delay and individual visiting coincidence. Furthermore, they proposed a method to evaluate the impact of individual location visiting on the privacy of other relationships in social networks. Backes et al. [17] assessed the effectiveness of some basic protective mechanisms in preventing friendship inference through co-existence. Camilli et al. [18] studied the potential security risks caused by the co-existence of multiple users in the same location when users post content on location-based social networks. Furthermore, they designed a privacy-protection strategy pertinently without affecting the normal operation of major services. Li et al. [8] established a sensitive-relationship inference model based on check-in data and adopted the method of deleting check-in data to reduce user similarity.

Feng et al. [19] proposed a service framework and personalized the dynamic privacy model to detect user-relational privacy risks. However, this cannot provide specific protection measures. Qian et al. [20] used admissible text to confuse content and automatically generate social behaviors to create indistinguishable edge connections. The goal of this is to prevent privacy reasoning attacks and to design a defense mechanism deployed on the user's local computer.

In order to enhance readability, the state-of-the-art schemes are summarized in Table 1. In conclusion, there are few protection schemes with a specific algorithm, and the existing specific methods do not fully consider the privacy degree of location nor the impact that

co-checking-in at different locations has on different deduction contributions to sensitive relationships. This problem may not only cause the omission of important check-ins, which leads to privacy leakage, but may also cause large information losses. This paper proposes a location-based sensitive-relationship protection algorithm for social networks.

Table 1. State-of-the-art schemes.

Scheme	Advantages	Disadvantages	References
Sensitive relation Deduce model	High accuracy of deduction, Short running time	Lack of protection methods	[9,16]
Graph structure-based protection method	Simple implementation, Short running time	Lack of protection against background knowledge attacks	[10,11]
Privacy leakage alert scheme	Low time cost, High applicability	Privacy protection degree uncertain, High risk.	[12,13,17,19]
Protection framework	High innovation, applicability and flexibility	Lack of specific implementation methods.	[14,15,18]
Protection Scheme with specific model	High degree of privacy and service quality	High precomputing overhead and time cost, Service requires specific optimization.	[8,20]

3. Preliminary Studies

In this paper, the social network is modeled as a graph $G(V, E, C)$, where V is the set of vertices, E is the set of edges and C is the set of check-in data. We define each vertex in set V as a user in a social network. The edge (u, v) in set E indicates that vertices u and v have a relationship. The check-in data in set C are represented as <u,t,l>, where u is the user ID, t is the check-in time, and l is the check-in location.

Definition 1. *Location Frequency–Inverse User Frequency (short for LF-IUF): this is defined as the product of location frequency (LF) and inverse user frequency (IUF), in which LF and IUF can be calculated by Equations (1) and (2), respectively. Given user u, the check-in data set C_u and the total check-in data set Cu of u at location l, the location frequency (LF) of user u at l is defined as Equation (1):*

$$LF_{u,l} = \frac{|C_{u,l}|}{|C_u|} \tag{1}$$

Given location l, user set U and a subset U_l, whose check-in points are located at l, the inverse user frequency (IUF) of location l is defined as Equation (2). Semantically, this Equation indicates the degree of location privacy. The lower the user check-in frequency at location l, the higher the privacy degree of l.

$$IUF_l = log(\frac{|U|}{|U_l|}) \tag{2}$$

The product of these two parts equal to u′s visiting characteristic at l is calculated as Equation (3):

$$LF - IUF_{u,l} = LF_{u,l} \cdot IUF_l = \frac{|C_{u,l}|}{|C_u|} \cdot log(\frac{|U|}{|U_l|}) \tag{3}$$

Definition 2. *Visiting Characteristics Vector: Given u's $LF - IUF_{u,l}$ at location set $L(L_1, L_2 \ldots, l_n)$, the visiting characteristics vectors of a user u are calculated using Equation (4).*

$$\overrightarrow{u} = (LF - IUF_{u,l_1}, LF - IUF_{u,l_2}, \ldots, LF - IUF_{u,l_n}) \tag{4}$$

Assuming the number of elements in set U is 100, the location-visiting characteristic of u_1 is calculated as $LF - IUF_{u1,l1} = 3/8 \cdot log(100/2) \approx 1.066$. Similarly, u_1 visiting charac-

teristics vector is obtained by calculating the $LF-IUF$ of u_1 at other locations, where $\overrightarrow{u_1} = [1.066, 0.0, 0.0, 0.0, 1.778, 0.0, 0.0, 0.877]$.

Definition 3. *User Similarity: Given users' visiting characteristics vector of $\langle u, v \rangle$, the similarity of $\langle u, v \rangle$ is defined as the cosine similarity of u and v.*

In order to facilitate calculation, the vectors in this paper are all normalized, on which basis the cosine similarity of the two vectors can be calculated through the dot product, as shown in Equation (5).

$$Sim(u,v) = Cos(\overrightarrow{u}, \overrightarrow{v}) = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{||\overrightarrow{u}| \times |\overrightarrow{v}||} = \sum_{i=1}^{n} Nu(l_i) \cdot Nv(l_i) \tag{5}$$

Definition 4. *Inference Secure: Given social network graph $G(E, V, C)$, sensitive-relationship set S and similarity threshold α, if $\forall (u,v) \in S$, $(u,v) \notin E$ and $Sim(u,v) < \alpha$, then G satisfies the Inference Secure parameter.*

Definition 5. *Visiting Pattern: Given user's check-in number at location set $L(L_1, L_2 \ldots, l_n)$, the visiting pattern of user u is defined as Equation (6).*

$$P(u) = (|C_{u,l_1}|, (|C_{u,l_2}|, \ldots, (|C_{u,l_n}|) \tag{6}$$

To facilitate subsequent calculation, subsequent visiting patterns in this paper are calculated using Equation (7).

$$P_N(u) = (\frac{|C_{u,l_1}|}{|C_u|}, \frac{|C_{u,l_2}|}{|C_u|}, \ldots, \frac{|C_{u,l_i}|}{|C_u|}) \tag{7}$$

Definition 6. *Information Loss: Given the original social network graph G and inference-secure graph PG, the information loss between G and PG is defined as per Equation (8). Here, U refers to the user set based on G. Semantically, it measures the changes in users' visiting patterns before and after protection.*

$$Los(G, PG) = \sum_{i=1}^{|U|} \sqrt{\sum_{j=1}^{|L|} (\frac{|C_{ui,lj}|}{|C_{ui}|} - \frac{|C'_{ui,lj}|}{|C'_{ui}|})^2} \tag{8}$$

4. Proposed Scheme

This section proposes a sensitive-relationship privacy-protection algorithm (short for *LVCPP*) based on location-visiting characteristics to prevent sensitive-relationship inference attacks. Section 4.1 proposes the main algorithm *LVCPP*, Section 4.3 introduces heuristic rule, and the other Sections introduce sub-algorithms. In Appendix A, we explain the proposed algorithm's process with the example in Figure 1.

4.1. The LVCPP Algorithm

The main idea of the *LVCPP* algorithm is as follows: Firstly, a sensitive relationship is deleted from social network graph *G*, and then we use algorithm *DSD* (will be discussed later) to judge the sensitive relationship in set *S*. Next, we generate a sensitive-relationship set S_0, which needs to be protected. Then, the *DeleteCandidate* algorithm is invoked to suppress check-ins under the protected rules constraint. From then on, we invoke algorithm *AddCandidate*, which adds dummy check-ins according to the accessibility of spatiotemporal constraints. This can better protect sensitive relationship privacy and reduce the loss of information based on the heuristic rule without causing other sensitive relationships to leak. Finally, privacy protection of sensitive relationships is completed.

As shown in Algorithm 1, it first deletes the sensitive relationship from graph *G* and then finds the sensitive-relationship set S_0, which should be protected by algorithm *DSD* (line 2–3). Then, we construct a candidate check-ins list *Q* based on the check-in suppression

algorithm (line 4). During the construction, we also calculate the score of each check-in, which is calculated by a heuristic function. This measures the joint influence of deductive contribution and information loss of the suppression operation. The higher the *Score*, the higher the deductive contribution after suppression and the smaller the information loss. The suppression with the maximum *Score* value is executed repeatedly until the similarities of all sensitive relationships are below the threshold or the operation candidate list is empty (lines 5–9). After that, on the premise that protected sensitive relationships are not leaked, the *AddCandidate* algorithm will continue to be executed. Then, the list of candidate dummy check-in addition operations P is obtained through the *AddCandidate* algorithm (line 10). The addition with the largest *Score* value is executed repeatedly until S_0 is empty, which means all sensitive relationships are protected (lines 11–15). Finally, the algorithm returns the safety-deduction graph PG (line 16).

Algorithm 1: Location-visiting-characteristics-based privacy-protection algorithm

Input : Graph G,Sensitive-relationship set S, Similarity threshold α
Output: Inference secure graph PG

1 $C_{Del} = \emptyset, C_{Add} = \emptyset$;
2 $E(G) = E(G) \backslash S$;
3 $S_0 = DSD(G, S, \alpha)$;
4 $Q = DeleteCandidate(S_0, C_{Del})$;
5 **while** $S_0 \neq \emptyset$ *and* $Q \neq \emptyset$ **do**
6 Select op_{del} from Q,which its $Score(op)$ is max;
7 Operate op_{del} to Delete;
8 Update S_0 and C_{Del} and Q;
9 **end**
10 $P = AddCandidate(S_0, C_{Del})$;
11 **while** $S_0 \neq \emptyset$ *and* $P \neq \emptyset$ **do**
12 Select op_{add} from P,which its $Score(op)$ is max;
13 Operate op_{add} to Add;
14 Update S_0 and C_{Add} and P;
15 **end**
16 **return** PG

4.2. The Algorithm DSD

As shown in Algorithm 2, algorithm DSD takes social network graph G, sensitive-relationship set S and user similarity threshold α as input and outputs a sensitive-relationship set S_0 to be protected.

Algorithm 2: Determine Sensitive Data algorithm

Input : Graph G,Sensitive-relationship set S,Similarity threshold α
Output: Sensitive-relationship set S_0

1 $S_0 = \emptyset$;
2 **for** *each* (u, v) *in* S **do**
3 **for** *each* l_i *in* L **do**
4 Calculate $LF - IUF_{u,li}$ and $LF - IUF_{v,li}$;
5 **end**
6 **if** *Sim(u,v)* $>\alpha$ **then**
7 Insert (u, v) into S_0 ;
8 **end**
9 **end**
10 **return** S_0

Firstly, the sensitive-relationship set S is judged in the algorithm, and the users' visiting characteristics vectors corresponding to the user in each pair of relationships and their similarity are calculated (lines 2–5). If the similarity of the sensitive relationship is higher than the threshold α, the sensitive relationship will be added to set S_0 (lines 6–9). Finally, algorithm DSD returns a set of sensitive relationships S_0 (line 10).

4.3. Check-In Operation Metrics

In this section, heuristic rlue and metrics of check-in operation will be introduced. $Op =< u, L >$ is used to represent a check-in operation (suppression or addition). To measure the joint influence of check-in operation and the user visiting patterns on the similarity of sensitive relational users and to maximize the effect of the check-in operation, this paper designs heuristic rule, as shown in Equation (9), to measure the impact of a check-in operation. Semantically, it indicates ratio of deductive contribution to information loss. The higher the score is, the better the effect is.

$$Score(op) = \frac{Ic(op)}{Cost(op)} \tag{9}$$

The upper part represents the change in the similarity of the sensitive relationships after the check-in operation compared with before, which can be measured by the difference before and after an operation. The definition of deductive contribution $IC(OP)$ is shown in Equation (10). The lower part represents the information loss of the user's visiting pattern. The larger the value of $IC(op)$ is, the larger the similarity changes. The smaller the loss of information due to the check-in operation is, the better data utility is. The information loss is shown in:

$$Ic(op) = Cos(\overrightarrow{u'}, \overrightarrow{v'}) - Cos(\overrightarrow{u}, \overrightarrow{v}) \tag{10}$$

Equation (11) represents the change in the user with a sensitive relationship and the original visiting pattern after the check-in operation. This can be obtained by calculating the Euclidean distance between the original and existing visiting patterns of the two users.

$$Cost(op) = \sqrt{\sum_{i=1}^{|L|}(\frac{|C'_{u,li}|}{C'_u} - \frac{|C_{u,li}|}{C_u})^2} + \sqrt{\sum_{i=1}^{|L|}(\frac{|C'_{v,li}|}{C'_v} - \frac{|C_{v,li}|}{C_v})^2} \tag{11}$$

Take the example in Figure 1, if the check-in of suppression u_1 at location l_1 is C_1, $Sim(u_1, v_1) = 0.404$, then $Ic = 0.087, Cost = 0.985$ and $Score = 0.0883$.

4.4. The Algorithm DeleteCandidate

The *DeleteCandidate* algorithm takes social network graph G and sensitive relation set S as input and outputs a suppression operation list Q in descending order of *Score*. It comprehensively measures the influence of the check-in-suppression operation on deductive contribution and information loss through the heuristic rules. Since removing check-in will change part of the trajectory shape and increase the probability of leakage, in order to keep the trajectory in its original shape as much as possible and to reduce the risk of privacy leakage, this part will not remove the check-in with the greatest influence at the beginning and end of the trajectory.

As shown in Algorithm 3, it performs a heuristic search on the common check-in locations of sensitive relationships to ascertain whether check-ins at each location can be removed. If only one user's check-in at the given position can be suppressed in a sensitive relationship, the check-in-suppression operation op of this user at this position is generated and the *Score* value is calculated (lines 2–10). If all the users in the sensitive relationship can be suppressed at this location, the check-in-suppression operations op_u and op_v are generated. Then, their *Score* values are calculated, and the check-in operation with a larger *Score* value is regarded as op (line 12–15). Next, it inserts op and the corresponding *Score*

into list Q (line 16). Finally, the algorithm returns a candidate list of suppressed operations Q (line 19).

Algorithm 3: DeleteCandidate algorithm

Input : Graph G,Sensitive-relationship set S_0
Output: Check-in to remove candidate list Q

1 $Q = \emptyset$;
2 **for** *each (u,v) in* S_0 **do**
3 **for** *each co-check-in location* l_i *of u and v* **do**
4 **if** $\exists ckv$ *in* C_u *at* l_i *is not the first or the last check-in in its origin trajectory and* $\forall ckv$ *is,* **then**
5 Generate op = (u,l_i) ;
6 Calculate $< Score, op >$;
7 **end**
8 **if** $\exists ckv$ *in* C_v *at* l_i *is not the first or the last check-in in its origin trajectory and* $\forall cku$ *is,* **then**
9 Generate op = (v,l_i) ;
10 Calculate $< Score, op >$;
11 **end**
12 **if** $\exists ckv$ *and ckv at* l_i *are both not the first or the last check-in in its origin trajectory* **then**
13 Generate $op_u = (u, l_i)$ and $op_v = (v, l_i)$;
14 Calculate $< Score, op >$ = max(Score(op_u), Score(op_v));
15 **end**
16 Insert <op, $Score$> into Q;
17 **end**
18 **end**
19 **return** Q ;

4.5. The Algorithm AddCandidate

In this section, the check-in-addition algorithm is proposed to an addition operation list P in descending order of $Score$. The check-in addition process is constrained by two additional conditions: spatiotemporal accessibility and user visiting pattern. The user-visiting-pattern condition means that the dummy can only be added at locations the users have visited.

Definition 7. *Spatiotemporal accessibility: Between two locations, if the user is reachable in a certain period of time with the maximum average moving speed Vmax, spatiotemporal accessibility is satisfied, as shown in Equation (12):*

$$\begin{cases} \dfrac{dist(l_i, l_{i-1})}{(t_i - t_{i-1})} \leqslant V_{max}, \\ \dfrac{dist(l_{i+1}, l_i)}{(t_{i+1} - t_i)} \leqslant V_{max} \end{cases} \tag{12}$$

l_{i-1} and l_{i+1} represent two locations before and after the location candidate in the trajectory. $dist(L_{i-1}, l_i), dist(l_i, l_{i+1})$ represents the Euclidean distance between the candidate location. V_{max} represents the average maximum moving speed of the user. If this set of inequalities is true, then location l_i satisfies the spatiotemporal accessibility, and Equation (13) can be derived to represent the range of check-in times t_i at location l_i. t_i represents the range of check-in times of the candidate dummy check-in.

$$t_{i-1} + \frac{dist(l_i, l_{i-1})}{V_{max}} \leqslant t_i \leqslant t_{i+1} - \frac{dist(l_i, l_{i+1})}{V_{max}} \tag{13}$$

Figure 2 shows the structure of the candidate spatiotemporal index, which shows the process of addition candidate generation. Figure 2a shows the distance matrix obtained from pre-calculated data, and Figure 2b,c shows the index of distance and time of the dummy check-in location.

	l_1	l_2	l_3	l_4	l_5
l_1	0	0.1	0.2	0.3	0.4
l_2	0.1	0	0.22	0.31	0.12
l_3	0.2	0.22	0	0.8	0.5
l_4	0.3	0.31	0.8	0	0.3
l_5	0.7	0.72	0.5	0.3	0

(a)

l_1	l_2	l_3	l_4
	0.1	0.2	0.3
l_2	l_1		
	0.1		
l_3	l_1	l_2	
	0.2	0.22	
l_4	l_1	l_5	
	0.3	0.3	
l_5	l_4		
	0.3		

(b)

l_1	t_1	t_2	t_3
	$(13,u_1)$	$(5,u_3)$	$(2,u_4)$
l_2	t_6		
	$(2,u_2)$		
l_3	t_8	t_7	
	$(12,u_3)$	$(2,u_4)$	
l_4	t_9	t_{11}	
	$(18,u_1)$	$(9,u_2)$	
l_5	t_5		
	$(11,u_2)$		

(c)

Figure 2. Distance matrix and location D-index and T-index: (**a**) the distance matrix between locations, with the distance unit being *km*; (**b**) the index of distance between candidate location and others; (**c**) the index of the optional adding time for each candidate dummy check-in location.

Definition 8. *Candidate spatiotemporal index: Given a candidate location l, the distance index of this location from other locations is defined as a list, represented by $d.D$. The stored elements in list $d.D$ are the distances between location l and other locations. All elements in $d.D$ are arranged in order of ascending distance. The elements in the time index $t.D$ are the candidates of the dummy check-in time. It stores the length of the trajectory and the user ID. All elements in $t.D$ are arranged in descending order according to the length of trajectory.*

Algorithm 4 takes graph G, sensitive-relationship set S_0, deleted check-in set C_{del} and added check-in set C_{add} as input. It outputs a candidate list of check-in-addition operations P. Through Algorithm 4, information loss can be effectively reduced, more sensitive relationships can be protected and the corresponding similarity of sensitive relationships can be reduced.

Firstly, Algorithm 4 obtains the distance index list $t.D$ of each point within the range of spatiotemporal accessibility via searching (lines 1–5). Then, it iterates through each trajectory to judge whether the location can be a candidate. If two consecutive locations are spatiotemporally accessible to location l, l can be a candidate for the dummy check-in. Then, according to the equation for calculating waiting time, the candidate location and trajectory length is added into $t.D$ (line 6–7). For a trajectory, the information loss caused by the addition operation is measured according to the length of the candidate addition trajectory. The longer the trajectory, the smaller the information loss within a single check-in operation. Therefore, the check-in for each location in $t.D$ with the maximum trajectory length will be the most suitable candidate. Then, operation op is generated and its *Score* value is calculated (line 8–10). The algorithm inserts op and the corresponding *Score* into list P (line 11). Finally, it returns P (line 12).

Algorithm 4: AddCandidate algorithm

Input : Graph G,Sensitive-relationship set S_0,Check-in deleted Set C_{Del},Check-in added Set C_{Add},max moving distance d_{max}

Output: Add Check-in candidate list P

$P = \emptyset$;
$t.D = \emptyset, l.D = \emptyset$;
for *each (u,v) in* S_0 **do**
  for *each location* l_i *in trajectory of u and v* **do**
    Perform a traversal search at location l_i until the distance is longer than d_{max}, generate $l.D$.;
  end
  Calculate time of each dummy candidate in $l.D$, generate $t.D$.;
end
for *candidate location* $l \in t.D$ **do**
  Generate $op = (u, l, t)$;
  Calculate $Score = Score(op)$;
  Insert $(op, Score)$ into P ;
end
return P ;

4.6. Complexity Analysis

Definition 9. *Co-location Check-in set: Given two users u and v at the location l, where they both checked-in. The joint check-in set of u and u at l is defined as the co-location check-in set* Co_l. *Co represents the collection of* Co_l *at each co-check-in location.*

Assuming the maximum set of co-check-in locations of user pairs is L, Tr represents the maximum trajectory set of a user, and C represents the maximum check-in set of a user. Co is defined as the maximum co-location check-in set.

In the complexity analysis, each sub-algorithm is analyzed first, and then the complexity of scheme is analyzed.

In algorithm Dsd, the calculation of $LF - IUF$ needs $|S||L|$ times; the complexity of this part is $O(|S||L)$. In the *DeleteCandidate* algorithm, in the worst case, complexity of one suppression operation is $O(|S||L|)$. It can remove $|S||Co|$ check-in in most cases, which means the suppression operation can be performed $|S||Co|$ times. The complexity of this part is $O(|S|^2|L||Co|)$. In the *AddCandidate* algorithm, the complexity of selecting a candidate is $O(|S||L|)$. One user can add $|C| - |Tr|$ check-ins at most. The largest number of check-ins that can be added in total is $|S||C|$, so the complexity of this part is $O(|S|^2|L||C|)$. As a result, the overall complexity is $O(|S|^2|L||Co| + |S|^2|L||C|)$. In addition, $|Co| < 2|C|$, so the overall complexity in the worst case is $O(3|S|^2|L||C|)$.

5. Experimental Evaluation

5.1. Experiment Settings

This section analyzes and evaluates the performance of the proposed sensitive-relationship privacy-protection algorithm. To verify the effectiveness of the algorithm, two real datasets, Brightkite and Gowalla, were used in the experiment, which were published by the Stanford Network Analysis Platform (SNAP). The Brightkite dataset contains 58,228 users' check-ins from April 2008 to October 2010, with 4,490,000 check-ins and 214,078 friend relationships. In the Gowalla dataset, 196,591 users checked in from February 2009 to October 2010, with 6.44 million check-ins and 950,327 friend relationships. Table 2 shows the relevant statistics for the two datasets.

Table 2. Specific metrics for Brightkite and Gowalla datasets.

Metrics	Brightkite	Gowalla
Users	58,228	196,591
Check-ins	4,491,143	6,442,892
Relationships	214,078	950,327
Max Moving Speed (km/min)	1.23	1.13

In this paper, two versions of the LVCPP algorithm are implemented: R-LVCPP and H-LVCPP. In the process of protection, R-LVCPP and H-LVCPP select deleted objects by randomization and heuristic rules, respectively. The comparison algorithm is the Li algorithm [18], which adopts heuristic rules to protect sensitive relationships and reduces information loss by deleting check-ins and edges. This paper implements and modifies the Li algorithm and redefines the success rate of protection and information loss.

This paper tests the execution time, information loss and protection success rate of the algorithms. In the test, the similarity threshold ranges from 0.1 to 0.5. Different numbers of sensitive relationships were randomly selected in the experiment for protection, and the number of protected sensitive edges k was 50, 100, 150, 200 and 250. The software and hardware environment of this experiment is as follows:

(1) Computer hardware configuration: Intel Core I5 2.5 ghz, 8 GB DRAM;
(2) Operating system: Windows 10;
(3) Programming environment: Python language, PyCharm development platform.

5.2. The Performance Evaluation

The running time evaluation. We first evaluate changes in the algorithm's running time with the similarity threshold α and the number of sensitive edges k. The experimental results are shown in Figures 3 and 4.

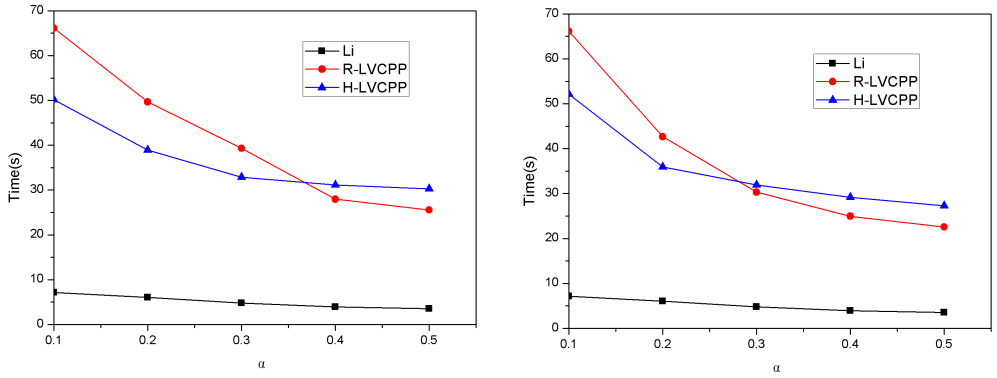


Figure 3. Running time evaluation under different α: (**a**) Brightkite; (**b**) Gowalla.

Figure 3 shows the variation in running time of the three algorithms under different α, where $k = 50$. We find that the running time gradually decreases with the decrease in α. When threshold α increases, the number of sensitive edges that should be protected decreases, plus the relationship can be protected with reduced check-in operations, so the runtime decreases. Li has the lowest runtime because the similarity change caused by the check-in operation is greater. As the similarity calculation method is different, Li can delete less check-ins to bring the similarity under the threshold. H-LVCPP runtime is less than that of R-LVCPP because more check-ins need to be deleted when the threshold is small in the random selecting rule. The more check-ins that need to be deleted, the more efficient H-LVCPP will be because the heuristic strategy adopted by H-LVCPP can accurately select operation candidates with the most contributions.

Figure 4 shows the running time of these three algorithms under different k. The parameter α is set to 0.5. We find that, with the increase in k, the number of sensitive edges that should be protected increases. We also find that the running time cost of H-LVCPP is higher than the Li and R-LVCPP algorithms. The Li algorithm has the best performance with the lowest time cost, and the runtimes of R/H-LVCPP are higher than that of Li. The reason is the similarity change caused by the check-in operation is greater in Li, and the complexity of Li is much lower than LVCPP. When $k < 50$, the runtime of H-LVCPP is lower than R-LVCPP because the heuristic strategy adopted by H-LVCPP considers the global impact of each operation, which will take time to more accurately select the most-contributing candidate. The more sensitive edges need to protected, the more efficient H-LVCPP will be. Thus, H-LVCPP is more efficient than R-LVCPP when the α is small.

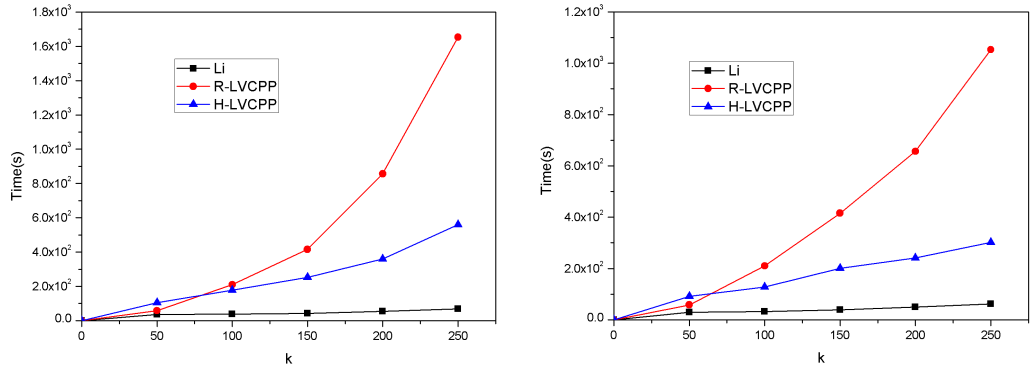


Figure 4. Runtime varies with parameter k: (**a**) Brightkite; (**b**) Gowalla.

The information loss evaluation. Next, we evaluate the information loss of algorithms under different α and K. The information loss indicators adopted in this paper include (1) the number of deleted check-ins and (2) the average visiting pattern loss. The experimental results are shown in Figures 5–7.

The check-in loss evaluation. We evaluate the check-in loss under different α when $k = 150$. As depicted in Figure 5, when α decreases, the number of sensitive relationships that need protection decreases, which causes the number of deleted check-ins to gradually decrease. Li has the best result, and H-LVCPP is better than R-LVCPP. The change in similarity caused by the check-in operation is greater in Li, as it can delete less check-ins for protection. R-LVCPP adopts a randomization method, so the contribution of deleted check-ins is uncertain, which may cause it to delete more check-ins than is ideal. By adopting the heuristic rule, H-LVCPP chooses the candidate with the most contributions and deletes fewer check-ins than R-LVCPP.

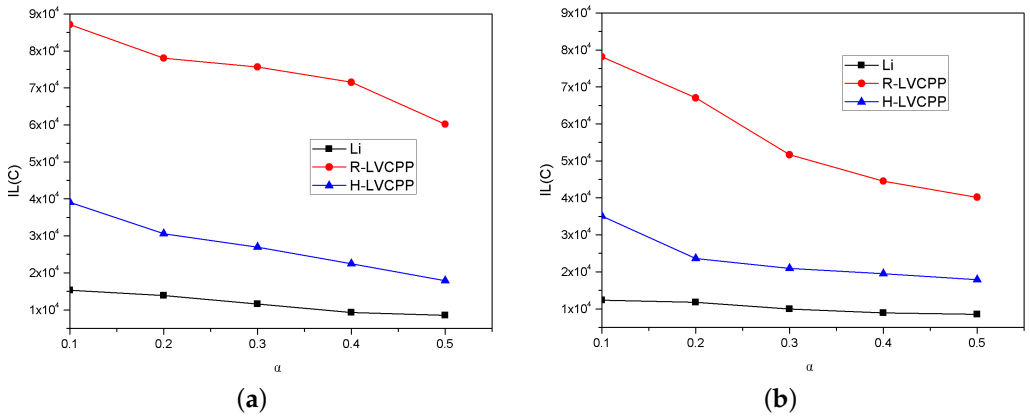


Figure 5. Check-in loss evaluation under different α: (**a**) Brightkite; (**b**) Gowalla.

As shown in Figure 6, the total number of deleted check-ins increases with K, Li has has the best performance, R-LVCPP is the worst and H-LVCPP is better than R-LVCPP and not far from Li. It is obvious that the number of lost check-ins with the H-LVCPP and Li algorithms using heuristic algorithms is much lower than that of R-LVCPP. The results in Figures 5 and 6 have some similarities, and the number of sensitive relationships that need protection increases, which increases the number of check-ins that need to be deleted. The efficiency of H-LVCPP is better than that of R-LVCPP but slightly lower than Li.

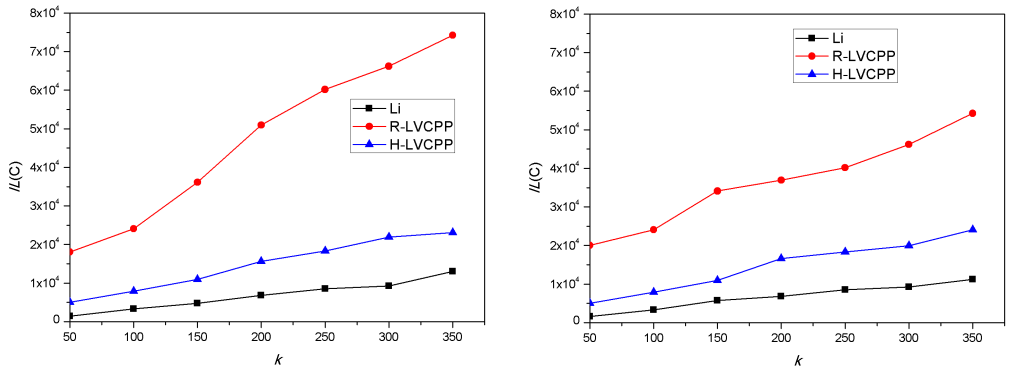


Figure 6. Check-in loss evaluation under different k: (**a**) Brightkite; (**b**) Gowalla.

Pattern loss evaluation. The pattern loss of the user visiting mode is shown in Figure 7. R-LVCPP with randomization method causes the largest pattern loss, while in H-LVCPP, it is smaller. From the previous experimental results, we can see that the number of check-ins deleted by Li is far less than LVCPP, so Li causes the lowest pattern loss. The heuristic rules adopted by H-LVCPP will select candidates with small pattern loss, so the result is more acceptable than that with R-LVCPP.

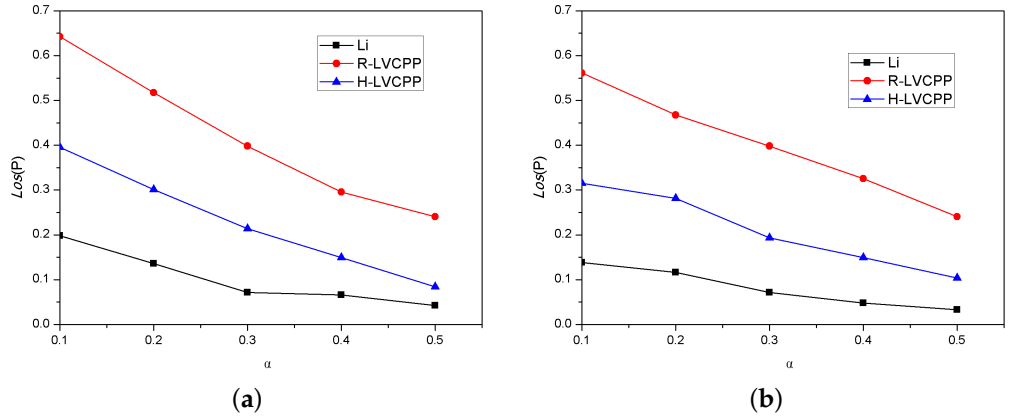


Figure 7. Pattern loss evaluation under different α: (**a**) Brightkite; (**b**) Gowalla.

The protection success rate evaluation. In this section, we define the success rate R in Equation (14), where S is the sensitive-relationship dataset and Sd is a subset after the protection of S, which contains protection-failure relationships under the location-visiting-characteristics' protection model.

$$R = \frac{|S| - |Sd|}{|S|} \tag{14}$$

The experimental results are shown in Figures 8 and 9.

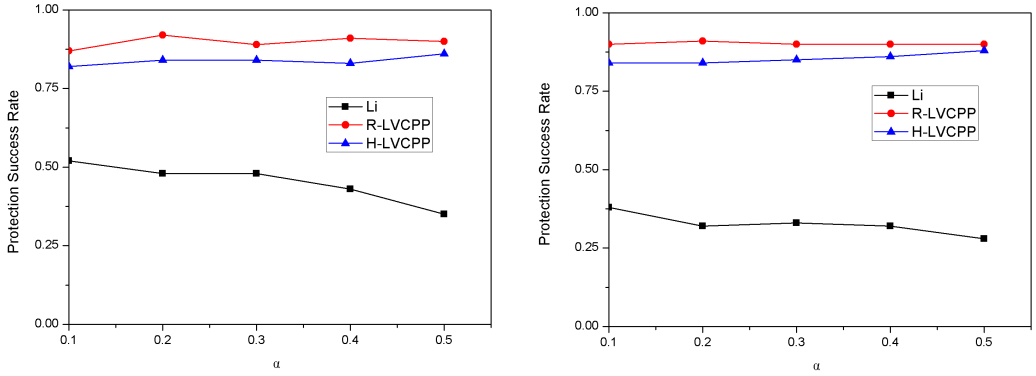


Figure 8. Protection success rate evaluation under different α: (**a**) Brightkite; (**b**) Gowalla.

The protection success rate varies with α, as shown in Figure 8 when k = 150. The success rate changes little, R-LVCPP has the highest protection success rate, H-LVCPP is slightly lower and Li is the worst. The reason is that both H-LVCPP and R-LVCPP adopt the location-visiting-characteristics' protection model, while Li does not. The conditions of R/H-LVCPP for the protection completed are more stringent than Li, so they delete more check-ins and the protection effect is better. Meanwhile, in order to retain the original trajectory shape, H-LVCPP has more constraints, which will reduce the number of candidates, so the success rate is slightly lower. With the increase in α, there are fewer check-ins deleted by Li, and the remaining check-ins cause the decrease in the protection success rate.

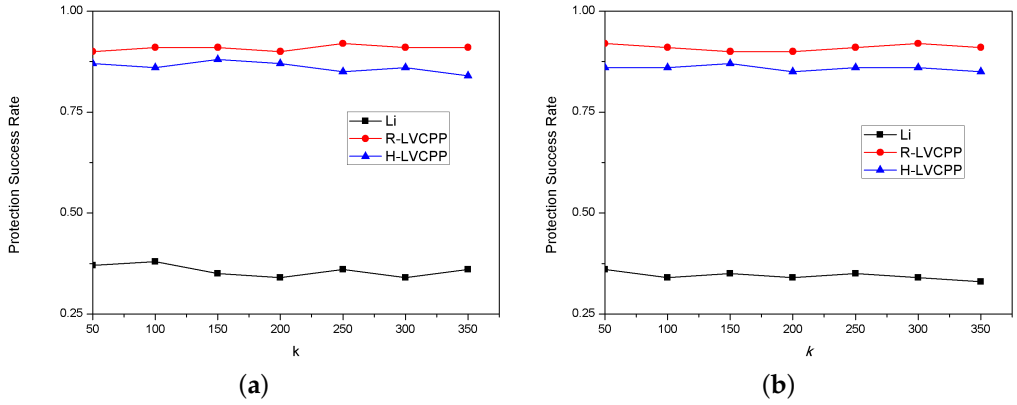


Figure 9. Protection success rate evaluation under different k: (**a**) Brightkite; (**b**) Gowalla.

The protection success rate varies with k, as shown in Figure 9 when α = 0.5. With decreasing k, the protection success rate of these three algorithms has little change. The result and reason for this is similar to Figure 8, in that Li is the worst, and due to the reason that H-LVCPP is subject to more constraints, the success rate of its protection is slightly lower than the best effect of R-LVCPP. Meanwhile, parameter k has little effect on the success rate, so the result line is more smooth. It is obvious that the Li algorithm, which does not consider the degree of location privacy, has a smaller running time and information loss in protecting sensitive relationships. However, compared with the other two algorithms, its success rate is the lowest, resulting in a greater risk of privacy leakage. In the tested algorithms, the comprehensive performance of H-LVCPP is the best.

In the following parts, the comparison of the proposed method with state-of-the-art methods will be fully discussed in several aspects. The result is shown in Table 3, and the conclusions of the comparison are shown in Table 4.

In Tables 3 and 4, we compare and conclude the performance of the proposed method LVCPP with four other state-of-the-art schemes, including Li [8], EBM [9], Liu [10] and

Chen [12]. These schemes represent different ideas of current state-of-the-art protection methods. EBM represents schemes focused on detecting sensitive relationships, Liu represents Graph-structure-based protection methods, Chen represents the Leakage alert method and Li represents other spatiotemporal-feature-based methods that protect privacy by check-in operation. We have implemented these four algorithms and made corresponding modifications for comparison. In this part, parameter k is 150 and parameter α is 0.5. In the following tables, "-" means the effect is uncertain, "×" means there is no corresponding ability, "✓" means corresponding effect is well, and "✓ ✓" means the effect is outstanding.

Table 3. Comparison of the proposed method with state-of-the-art methods.

	Pattern Loss	Protection Success Rate	Time
EBM	-	-	19.61 s
Liu	1	0.12	14.34 s
Li	0.21	0.39	34.87 s
Chen	-	-	18.33 s
LVCPP	0.33	0.88	214.21 s

Table 4. Conclusion of Comparison.

	Representative Scheme	Specific Protection Method	High Privacy Security	Low Information Loss	High Time Efficiency
Sensitive relation Deduce model	EBM	×	×	✓	✓
Graph-structure-based protection method	Liu	✓	×	×	✓
Leakage alert method	Chen	×	-	✓	✓
Other spatiotemporal-feature-based method	Li	✓	×	✓ ✓	✓
Proposed method	LVCPP	✓	✓ ✓	✓	×

As shown in Table 3, in the information loss aspect, the pattern information loss of EBM and Chen are uncertain because no specific protection method is proposed. Liu does not consider the spatiotemporal features of sensitive relationships, and the node operation will delete all user check-ins, so its pattern loss is the worst. Li and LVCPP adopt heuristic rules, which will obviously reduce the number of check-ins deleted; thus, their pattern losses are small. In the success rate aspect, Liu has lowest rate because it only deletes the nodes and relationships in a common neighbor model, which will lead to few relationships being protected. Although its heuristic rules delete many check-ins, it does not fully consider spatiotemporal characteristics, so its success rate is not high. In the protection model of Li, the similarity of many leaked privacy relationships in the LVCPP model is below the threshold, so there are few protected sensitive relationships. LVCPP has the highest score because it considers the location-visiting characteristics; thus, many sensitive relationships that cannot be deduced by other protection models are protected under this model. In terms of running time, LVCPP is the worst because it sacrifices high complexity for a high success rate and low information loss. Chen and EBM both have a low running time because they focus on relationship deduction and alert without specific protection operations. The Liu method has the second best result because it only protects privacy under the graph structure. The result of Li is slightly higher because it adopts a heuristic algorithm.

In conclusion, as shown in Table 4, many state-of-the-art methods are focused on deducing the leakage of sensitive-relationship privacy and creating an alert in such a case.

Some schemes propose specific methods, but few schemes are based on spatiotemporal operation. As shown in the results, though the state-of-the-art schemes based on spatiotemporal features perform well in terms of efficiency and information loss, they have a high risk of privacy leakage. Our proposed method, LVCPP, has the best performance because of its high privacy security and its loss of information being within the acceptable range, though it takes much more time to run.

6. Conclusions

This paper proposes a sensitive-relationship-protection algorithm to prevent sensitive-relationship leakage. According to heuristic rules, the algorithm deletes check-ins and adds fake check-ins to protect sensitive relationships. The results based on real social network data show that the proposed algorithm (*LVCPP*) can effectively protect the privacy of sensitive relationships.

In the future, we plan to reinforce our approach with a dynamic social network that is caused by changes during communication. In another aspect, forms of data in IoT applications are gradually increasing, and we will conduct further research to improve the similarity measure.

Author Contributions: Conceptualization, M.J.; methodology, X.-Y.L.; software, M.J.; validation, X.-F.X., M.J. and X.-Y.L.;formal analysis, X.-F.X.; investigation, M.J.; resources, X.-F.X., C.-Y.Z.; data curation, M.J.; writing-original draft preparation, X.-F.X., M.J.; writing-review and editing, X.-F.X., M.J.; visualization, M.J.; supervision, X.-F.X.; project administration, X.-F.X., C.-Y.Z.; funding acquisition, C.-Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No.61802268).

Data Availability Statement: Two publicly available datasets were used in this study. Gowalla is available at [http://snap.stanford.edu/data/loc-Gowalla.html] (accessed on 10 February 2022), Brightkite can be downloaded from [http://snap.stanford.edu/data/loc-Brightkite.html] (accessed on 10 February 2022).

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

Appendix A

In this section, we will use the example in Figure 1 to explain the proposed algorithm. Assuming the number of users set in the example is 100, the threshold is 0.40 and the sensitivity relationship is $\langle u_1, u_2 \rangle$, the situation of $\langle u_1, u_2 \rangle$ before protection is shown in Table A1:

Table A1. Protection Example.

User	Check-In Vector	LF-IUF Vector	Visiting Pattern Vector
u_1	$\overrightarrow{u_1} = [3, 0, 0, 0, 5, 0, 0, 3]$	[1.066, 0.0, 0.0, 0.0, 1.778, 0.0, 0.0, 0.877]	[0.272, 0.0, 0.0, 0.0, 0.45, 0.0, 0.0, 0.272]
u_2	$\overrightarrow{u_2} = [3, 2, 0, 0, 0, 2, 0, 2]$	[1.304, 0.71, 0.0, 0.0, 0.0, 0.779, 0.0, 0.715]	[0.333, 0.222, 0.0, 0.0, 0.0, 0.222, 0.0, 0.222]

In addition, $\overrightarrow{U_l} = [2, 4, 3, 3, 2, 3, 2, 4]$, and U_l represent the number of users who have check-in at location l.

Determine Sensitive Data algorithm: Firstly, $\langle u_1, u_2 \rangle$ is deleted in G, and the DSD algorithm judges whether the relationship needs protection. According to Equation (13), user similarity of $\langle u_1, u_2 \rangle$ is $0.49 > 0.40$. Then, $\langle u_1, u_2 \rangle$ is added S_0.

DeleteCandidate algorithm: Then, the DeleteCandidate algorithm searches for qualified candidates, generates the suppression operation op and calculates the Score value. Assuming each location is qualified for candidate, the scores of candidates are shown in Table A2:

Table A2. Score in DeleteCandidate.

User	Score Vector
u_1	[0.088,0,0,0,−0.066,0,0,0.0310]
u_1	[0.044,−0.031,0,0,0,−0.037,0.050]

The algorithm chooses the candidate with the highest score, which is the check-in of u_1 at l_1. It operates op_{max} to delete the check-in and updates S_0, C_{Del} and Q. After that, the similarity of $\langle u_1, u_2 \rangle$ is 0.4045.

AddCandidate algorithm: After, the AddCandidate algorithm searches for qualified candidates that satisfy spatiotemporal accessibility, generates check-in-addition operation op and calculates Score value. Assuming each location is qualified for a candidate (except u_1 at l_1), the Score vector of candidates is shown in Table A3:

Table A3. Score in AddCandidate.

User	Score Vector
u_1	[x,−0.050,0.004,0.004,0.051,−0.059,0.005,−0.026]
u_1	[−0.005,0.034,0.008,0.008,−0.184,0.039,0.011,−0.040]

Next, the algorithm adds a dummy check-in of u_1 at l_5 and updates S_0, C_{Del} and List P. Then, the similarity is 0.352 below the threshold α. $\langle u_1, u_2 \rangle$ is removed, and S_0 is $\emptyset$.

Finally, the LVCPP algorithm ends and returns the inference secure graph PG.

References

1. Namasudra, S. An improved attribute-based encryption technique towards the data security in cloud computing. *Concurr. Comput. Pract. Exerc.* **2019**, *31*, e4364 . [CrossRef]
2. Namasudra, S. Fast and secure data accessing by using DNA computing for the cloud environment. *IEEE Trans. Serv. Comput.* **2020**, *1*, 99. [CrossRef]
3. Hooshmand, M.K.; Gad, I. Feature selection approach using ensemble learning for network anomaly detection. *CAAI Trans. Intell. Technol.* **2020**, *5*, 283–293.
4. Wani, A.; Khaliq, R. SDN-based intrusion detection system for IoT using deep learning classifier (IDSIoT-SDL). *CAAI Trans. Intell. Technol.* **2021**, *6*, 281–290. [CrossRef]
5. Mohindru, G.; Mondal, K. Different hybrid machine intelligence techniques for handling IoT-based imbalanced data. *CAAI Trans. Intell. Technol.* **2021**, *6*, 405–416. [CrossRef]
6. Wang, L.; Meng, X.F.T. Location privacy preservation in big data era: A survey. *J. Softw.* **2014**, *25*, 693–712.
7. Cranshaw, J.; Toch, E.; Hong, J.; Kittur, A.; Sadeh, N. Bridging the gap between physical location and online social networks. In Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Copenhagen, Denmark, 26–29 September 2010.
8. Liu, X.; Li, M.; Xia, X. Spatio-Temporal Features Based Sensitive Relationship Protection in Social Networks. In Proceedings of the International Conference on Web Information Systems and Applications, Taiyuan, China, 14–15 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 330–343.
9. Pham, H.; Shahabi, C.; Liu, Y. Inferring social strength from spatiotemporal data. *ACM Trans. Database Syst.* **2016**, *41*, 1–47. [CrossRef]
10. Liu, X.; Yang, X. Protecting sensitive relationships against inference attacks in social networks. In Proceedings of the International Conference on Database Systems for Advanced Applications, Busan, Korea, 15–19 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 335–350.
11. Huang, H.; Zhang, D.Z.T. Weighted Large-Scale Social Network Data Privacy Protection Method. *J. Comput. Res. Dev.* **2020**, *57*, 363–377.
12. Chen, W.H.; Li, W.J. A model for protecting location privacy against attacks from friends in SNS. *Comput. Eng. Sci.* **2015**, *37*, 692–698.
13. Alrayes, F.; Abdelmoty, A. Towards Location Privacy Awareness on Geo-Social Networks. In Proceedings of the International Conference on Next Generation Mobile Applications, Cardiff, UK, 24–26 August 2016.
14. Ahuja, R. Protecting against Inference Attacks on Co-Location Data. In Proceedings of the 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019.
15. Shahabi, C.; Fan, L. Privacy-preserving inference of social relationships from location data. In Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 3–6 November 2015.

16. Pham, H.; Shahabi, C. Spatial influence-measuring followship in the real world. In Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016.
17. Backes, M.; Humbert, M.; Pang, J. walk2friends: Inferringsocial links from mobility profiles. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017.
18. CaMilli, M. Preserving Co-Location Privacy in Geo-Social Networks. *arXiv* **2012**, arXiv:1203.3946.
19. Feng, Z.; Tan, H.; Shen, H. Relationship Privacy Protection for Mobile Social Network. In Proceedings of the International Conference on Advanced Cloud & Big Data, Chengdu, China, 13–16 August 2017.
20. Qian, J.; Li, X.; Yu, W. Social Network De-anonymization: More Adversarial Knowledge, More Users Re-Identified? *ACM Trans. Internet Technol.* **2017**, *19*, 1–22. [CrossRef]