Recently, rates of vehicle ownership have risen globally, exacerbating problems including air pollution, lack of parking, and traffic congestion. While many solutions to these problems have been proposed, carpooling remains one of the most effective approaches. Recently, several carpooling platforms have been built on cloud computing systems, with originators posting online lists of departure/arrival points and schedules from which participants can search for rides that match their needs. However, it can be difficult to make matches quickly and the systems are subject to privacy concerns in that they may disclose private information such as names, registration data, and departure/arrival schedules. This paper proposes a dynamic matching method for car/taxi pools for use in mobile devices via ad hoc Wi-Fi networks. The proposed method also preserves user privacy including names and departure/arrival schedules. Moreover, the system does not require the user to register any personal data, so such data cannot be leaked. The system was implemented on the Android mobile platform, allowing users to immediately and securely access the system via their smartphones.

1. Introduction

In recent years, car ownership has exploded globally, exacerbating the negative side effects of driving, including depletion of fossil fuels and the production of carbon dioxide and other emissions; thus driving accelerates the development of air pollution, acid rain, and climate change. In addition, increased car ownership has increased traffic congestion [1–4], strained the availability of existing parking facilities [5–12], and reduced economic performance [13]. Scholars have proposed many solutions to these issues, including carpooling [14–20], which remains one of the most effective solutions.

In 1983 Fagin and Williams [14] first proposed a sharing mechanism, while other scholars [15–18] have also proposed various carpooling methods. In these methods, group members rotate serving as drivers. However, this approach is difficult to promote or apply widely because it only works if group members already know each other and agree ahead of time to organize a carpool.

In recent years, increases to network speeds have prompted scholars [19, 20] to propose the establishment of networked platforms [21–29] as a mechanism for organizing carpools. Di Martino et al. [19] proposed the Lift4U platform, a carpool matching algorithm based on cloud and positioning systems. The platform uses Bing Maps and GPS modules to provide access to car sharing, bike sharing, and public transport on mobile platforms. Lalos et al. [20] also describe a dynamic framework for car and taxi pool services through the use of positioning systems on mobile devices to determine the location of carpool members. These positions are then shown directly on the mobile device interface, helping carpool drivers to arrange pickups.

These systems use Internet and cloud computing technologies [30] to provide access to information required to participate in carpooling schemes; thus group members need not know each other or organize the carpool ahead of time. However, initiating a carpool or finding carpool originators entails a process of online registration, login, search, and comparison, thus producing a match between
rider demand and driver availability. Therefore, in addition to being ineffective in helping carpoolers to coordinate in real time, these methods require originators to expose their personal information including name, location, destination, driving time, and expose participants’ personal information including name and telephone number.

This paper proposes a carpooling matching method deployed on mobile devices, wherein users can instantly initiate or participate in carpools via smartphones or tablet PCs via ad hoc Wi-Fi networks. The proposed method also protects user privacy, using a private matching method [31, 32] to protect the privacy of participants on both sides of the transaction, including name, origin, destination, and driving time.

The proposed system was implemented on Android smartphones, allowing users to directly, immediately, and securely find carpoolers. The implementation can be used directly to find rides from private carpools or taxi pools. For example, imagine a group of strangers emerging from a subway station and arriving at the adjacent bus stop immediately after their intended bus had pulled away, leaving them to wait another 20 minutes for the next bus. These individuals can access the proposed system via their smartphones to arrange for a taxi pool, thus saving time and money, while protecting their personal information.

The remainder of this paper is organized as follows. Section 2 discusses security issues and solutions for the proposed method and describes the research methods. Section 3 describes the proposed method, while Section 4 presents the analysis methods used, including accuracy analysis, locational error, and security analysis, and provides a comparison with other related methods. Section 5 introduces the implementation of the proposed system on Android phones and the final section presents concluding remarks.

2. Related Work

Most research into carpooling focuses on scheduling algorithms and matching algorithms, which are briefly introduced here.

2.1. Carpool Scheduling Algorithms. Fagin and Williams [14] and others [15–18] have proposed fair carpool scheduling algorithms wherein the participants take turns to drive the rest of the group. Here, we introduce four approaches including simple rotation, simple tokens, subsets, and fair carpool. Among these, simple rotation has a fixed set of members, where person $i$ is responsible for driving on the $i$th day and every $N$ driving days. This approach is simple to describe and it is easy to determine who drives next. However, inconsistencies (e.g., if a driver cannot drive and has to swap days) can disrupt the schedule and make it difficult to determine who drives next.

In the simple tokens approach, a participant $P$ pays a driver $D$ one “ride token” each time $P$ rides with $D$. The person with the fewest tokens would be the next driver. However, external conditions may render the designated driver unable to drive on a given day without warning.

In the subset approach, each of the $2^N - (N + 1)$ subsets of carpool members for $N$ persons can drive particular groups. Drivers get “subset ride tokens” from that subset to determine who drives next. However, this approach is very complex given the large number of subsets.

In fair carpool [14], each participant pays $U/k$ tokens to the driver in each carpool, where $U$ is a constant value and $k$ is the number of participants in each carpool time. However, this approach is unbalanced.

2.2. Carpooling Matching Algorithm. Many studies have proposed carpooling matching algorithms. Here, we review Di Martino et al.’s Lift4U platform [19], which uses Bing Maps and GPS modules to provide a range of carpooling applications (see Figure 1(a)). This method includes registration and data matching phases as follows.

(1) Registration Phase. In this stage, the user registers online. The user’s identity is confirmed, using information including the user’s name, gender, date of birth, and telephone number.

(2) Data, Matching Phase. This stage includes four steps through which the user is matched with a ride.

Step 1. The user accesses Bing Maps to establish points of departure and destination.

Step 2. The user sets a search radius around the departure point to search for other people seeking the same destination (see Figure 1(b)).

Step 3. The user finds other potential carpoolers within the search area (see Figure 1(c)), plots routes, and calculates the distances.

Step 4. The user determines the shortest route and invites potential carpoolers to join based on which pickups result in the shortest route (see Figure 1(d)).

3. Proposed Scheme

We use the error-tolerable private matching to achieve private carpool matching. This method not only allows the user to directly use mobile devices to find potential carpoolers with similar points of origin and destinations in real time but also allows the user to expand the search range using a quantifier, for which the quantification space can be autonomously selected (see Figure 2), thus finding more potential carpoolers.

This paper uses the symbols listed in Notations Section, where $Q(x, y) \in \{i \mid i = 0, 1, 2, \ldots, p = 2y\}$ s.t. $-y < x^{(q)} - x \leq y$. $Map()$ is the transformed longitude and latitude function. For example, $Map(x)$ represents the transformed longitude and latitude for map message $x$. Using Google Maps, the map message $Dst_A$ is thus transformed into the longitude and latitude message $LL_A$, and $LL_A = Map(Dst_A)$.

Figures 3 and 4 illustrate the operation process (using a taxi pool as an example). Figure 3 provides a detailed flow...
In this system, "originator" is the car owner inviting others to join a carpool or is an initial taxi passenger inviting others to share the ride, and "participant" is a person who wants to join a car or taxi pool. The process goes through the following steps.

**Step 1.** A user can either select "Listen" to determine whether someone has already initiated the carpool matching service (go to Step 2) or directly acts as the originator of a car or taxi pool (go to Step 3).

**Step 2.** After Listening, if the user receives a “Taxipool” or “Carpool” message, he/she can then elect to act as participant or originator. Otherwise if a “Taxipool” or “Carpool” message is not received and the participant option is not available, the user can still elect to act as originator, or return to Step 1.

**Step 3.** If the user elects to act as originator, an appropriate (i.e., “Carpool” or “Taxipool”) message is broadcast.

**Step 4.** If the user elects to act as participant, once he/she receives a “Taxipool” or “Carpool” message, he/she agrees to the match by sending a “GoMatch” message to the originator. Otherwise, go to Step 1.
Step 5. The originator receives the “GoMatch” message and inputs his/her own destination Dst_A and search range Rng_A. He/she computes $LL_A = \text{Map}(\text{Dst}_A)$, which uses Map() to transform Dst_A into the latitude and longitude message $LL_A$. He/she then uses the adjustment function to calculate $\text{Adj}_A = LL_A - LL_A^{(q)}$. Finally he/she transmits Adj_A and Rng_A to the participant.

Step 6. The participant inputs his/her own destination Dst_B and calculates $LL_B = \text{Map}(\text{Dst}_B)$. After receiving Adj_A and Rng_A from the originator, the participant uses the adjustment
function to calculate $LL'_B = LL_B - Adj_A$ and then computes $LL''_B = Q(LL'_B, Rng_A)$. Finally, the participant calculates $h_B = H(LL''_B)$ and transmits $h_B$ back to the originator.

**Step 7.** After the originator receives $h_B$, he/she calculates $H(LL''_A)$ and determines whether it matches $h_B$. If it matches, he/she transmits an “OK” message to the participant.

**Step 8.** Finally, when the participant receives the “OK” message, he/she begins to negotiate the taxi pool or carpool arrangement with the originator.

### 4. System Analysis and Comparison

#### 4.1. Match Correctness

In this system, matching will only succeed if the distance between $Dst_A$ and $Dst_B$ falls within $Rng_A$. That is to say, matching is only successful when

$$-Rng_A < LL_A - LL_B \leq Rng_A.$$  \(1\)

**Theorem 1.** In this system, if $-Rng_A < LL_A - LL_B \leq Rng_A$, matching will be successful.

**Proof.** Since $Adj_A = LL_A - LL'_A$ and $LL'_B = LL_B - Adj_A$, we have

$$LL_A - LL_B = LL'_A - LL'_B.$$  \(2\)

From the assumption $-Rng_A < LL_A - LL_B \leq Rng_A$, we get

$$-Rng_A < LL'_A - LL'_B \leq Rng_A.$$  \(3\)

Also from the definition of $Q(x, y)$, we know that $-Rng_A < LL''_A - LL''_B \leq Rng_A$ since $LL''_B = Q(LL'_B, Rng_A)$. We can get

$$-Rng_A \leq LL'_A - LL''_B < Rng_A.$$  \(4\)

From (3) and (4), we get $-2Rng_A < LL''_A - LL''_B < 2Rng_A$. Because both $LL''_A$ and $LL''_B$ belong to $\{ip | i = 0, 1, 2, \ldots, p = 2Rng_A\}$, we can conclude that $LL''_A = LL''_B$.

Conversely, if (1) does not hold, matching fails. That is, if $LL_A - LL_B > Rng_A$ or $LL_A - LL_B \leq -Rng_A$, matching fails.

**Theorem 2.** In this system, if $-Rng_A < LL_A - LL_B \leq Rng_A$ does not hold, matching fails.

**Proof.** (1) If $LL_A - LL_B > Rng_A$, from (2), we get

$$LL''_A - LL''_B > Rng_A.$$  \(5\)

From (4), we know that

$$LL'_A - LL''_B \geq -Rng_A.$$  \(6\)

From (5) and (6), we conclude that $LL''_A - LL''_B > 0$. Therefore, $LL''_A \neq LL''_B$.

(2) If $LL_A - LL_B \leq -Rng_A$, from (2), we get

$$LL'_A - LL'_B \leq -Rng_A.$$  \(7\)

From (4), we know that

$$LL''_B - LL''_A < Rng_A.$$  \(8\)
From (7) and (8), we conclude that \( L_L^{(q)} A - L_L^{(q)} B < 0 \). Therefore, \( L_L^{(q)} A \neq L_L^{(q)} B \).

4.2. Obtaining Latitude/Longitude Values and Quantified Scopes and Regulating Pairing Errors. Google Maps features different coordinate formats and GPS modes, requiring conversion for interoperability. The conversion formula is as follows:

\[
X \text{ degrees } Y \text{ minutes } Z \text{ seconds (GPS mode)} = X + \frac{Y}{60} + \frac{Z}{3600} \text{ (Google Maps decimal degrees)}.
\]

From (7) and (8), we conclude that \( L_L^{(q)} A - L_L^{(q)} B < 0 \). Therefore, \( L_L^{(q)} A \neq L_L^{(q)} B \).

4.2. Obtaining Latitude/Longitude Values and Quantified Scopes and Regulating Pairing Errors. Google Maps features different coordinate formats and GPS modes, requiring conversion for interoperability. The conversion formula is as follows:

\[
X \text{ degrees } Y \text{ minutes } Z \text{ seconds (GPS mode)} = X + \frac{Y}{60} + \frac{Z}{3600} \text{ (Google Maps decimal degrees)}.
\]

From (7) and (8), we conclude that \( L_L^{(q)} A - L_L^{(q)} B < 0 \). Therefore, \( L_L^{(q)} A \neq L_L^{(q)} B \).

4.2. Obtaining Latitude/Longitude Values and Quantified Scopes and Regulating Pairing Errors. Google Maps features different coordinate formats and GPS modes, requiring conversion for interoperability. The conversion formula is as follows:

\[
X \text{ degrees } Y \text{ minutes } Z \text{ seconds (GPS mode)} = X + \frac{Y}{60} + \frac{Z}{3600} \text{ (Google Maps decimal degrees)}.
\]

Longitude and latitude have different relationships to distance. In latitude, one degree is equivalent to 111.19 km, while one degree of longitude is equivalent to 101.35 km. Therefore, an input range with an error value of 1 km can be used to obtain a quantized interval of 2 km, which is converted to approximately 2/111.19 degrees latitude and 2/101.35 degrees longitude. This produces a set of quantized values.

If the quantized values for latitude and longitude are obtained, a restriction exists that the matching range can produce a square-shaped "square pairing" area, two km on each corner of the square pairing area exceeds Rng_A, but matching is successful (see red stripped ball in Figure 5(a)), while the maximal error distance is + ( 1 - \( \frac{\sqrt{2}}{2} \) ) Rng_A (or the maximal distance error rate is +41.4%), and the average false acceptance rate (FAR) is 21.46% and false rejection rate (FRR) is zero. However, one can also adjust the determination method for control of maximal error distance in \( - ( 1 - \frac{\sqrt{2}}{2} ) Rng_A \) (approximately \( -29.29 \) % with FAR being zero and FRR being 36.34%) (see Figure 5(b)) or \( \pm(3 - 2 \sqrt{2}) Rng_A \) (approximately \( \pm17.16 \) % with FAR being 4.57% and FRR being 16.62%) (see Figure 5(c)). Herein, Figure 5(c) is a relatively acceptable option. That is, for the user's input error range of 1 km (1000 m), the worst-case scenario would be (1) successful matching at a distance of 1171.5 meters (see point A in Figure 5(c)) or (2) unsuccessful matching at a distance of 828.5 meters (see point B in Figure 5(c)).

4.3. Security Analysis. When using the program, users may be concerned that others may be able to access their departure or destination point information. The proposed protocol ensures user privacy, allowing users to engage with the system without fear that their personal information may be leaked to unauthorized parties. In the proposed scheme, attackers are unable to obtain a plain text message through analyzing the transmitted information. Moreover, after finishing this protocol, neither the originator nor participant is aware of each other's information unless they share a common destination. We analyze possible means of attack as follows.

(1) Participant Attempts to Eavesdrop on the Privacy of Originator. In the proposed protocol, the participant is only able to receive Adj_A and Rng_A messages and is unable to access private information related to the originator such as intended destination.

(2) Originator Attempts to Eavesdrop on the Privacy of Participant. In the proposed protocol, the originator is only able to receive the participant's h_B message. If matching fails, the originator is unable to access private information related to the participant such as intended destination.

(3) Third Party Attackers Attempt to Eavesdrop on the Privacy of Originator or Participant. In the proposed protocol, although an attacker can eavesdrop on the Adj_A, Rng_A, and h_B messages, he/she is unable to access private information, such as the intended destination of either the originator or participant.

Thus, if the originator and participant have different intended destinations, neither is aware of the other. In addition, if an attacker eavesdrops on these messages, the messages cannot be compromised thus preserving the privacy of both parties.

In addition, if the negotiated content is subject to eavesdropping concerns, one of the following two advanced steps can be taken.

(1) The originator (in Step 7) calculates \( SK_{AB} = H(L_L^{(q)} A + 1) \) and the participant (in Step 8) calculates...
Figure 6: Implementation.
4.4. Comparison. Table 1 presents a comparison of our proposed system with scheduling algorithms (e.g., Fagin and Williams [14]) and carpooling matching algorithms (including Lift4U [19] and Lalos et al’s algorithm [20]). All the algorithms can provide carpool functions, while [19, 20] require GPS for the comparison. In terms of scheduling algorithms, because the system requires prior communication to coordinate matching, the users must know each other prior to using the system. In addition, because scheduling algorithms are planned ahead of time and Lift4U relies on a networked platform for matching, therefore it is unable to provide real-time, spur-of-the-moment carpooling. Most methods require prior registration or login and use networked platforms to run their matching algorithms, thus requiring Internet access. Also, most methods fail to consider privacy concerns. However, the proposed method provides all of these functions, while providing privacy protection.

5. Implementation

To validate the proposed method, we used the Android development platform to implement the matching program on a NEXUS S smartphone and used the SHA-1 hash algorithm as the hash function for the matching process. Using Wi-Fi as the transmission medium, we conducted actual tests of the privacy-enabled carpool matching system on a smartphone running Android.

As the flow chart shows in Figure 3, after the user initiates the program (see Figure 6(a)), the program first engages the “Listen” (i.e., “search”) mode (see Figure 6(b)). If it fails to find a ride, it organizes a new carpool (see Figure 6(c)). We assume that Alice initiates a new carpool. She then inputs her approximate intended destination (e.g., city) and uses Wi-Fi to broadcast a carpool or taxi pool message to find potential carpoolers. When multiple users (i.e., participants) receive the carpool or taxi pool message through the “Listen” mode, (see Figure 6(d)), willing parties which match the originator’s query specifications (in this case “Bob”) transmit back a “GoMatch” (i.e., “Join”) message. Alice receives the “GoMatch” message and then decides whether to approve the match (see Figure 6(e)).

Next Alice can opt to use a Diffie-Hellman key exchange (i.e., the “DH receiver”) (see Figure 6(f)). If so, both parties will use the Diffie-Hellman key exchange to establish a session key (see Figures 6(g) and 6(h)). Next, Alice inputs her own destination DstA and error range RngA (see Figure 6(i)), using map to transmit the converted coordinates, longitude, and latitude (see Figure 6(j)) and generate the resulting adjustments AdjA with RngA for transmission to Bob. Once Bob receives AdjA and RngA, he inputs his own destination DstB (see Figure 6(k)), using map to transmit the converted coordinates (see Figure 6(l)). The calculated hB is then transmitted to Alice who then compares whether H(LLB1) and hB are identical (see Figure 6(m)). If the match is successful, Alice sends the “OK” message to Bob. When Bob receives the successful match message (see Figure 6(n)), he and Alice begin to proceed to the carpool negotiation (see Figures 6(o) and 6(p)).

6. Conclusion

In this paper, we propose a protocol by which mobile devices can be used to provide instant matching for carpools and taxi pools in a way that protects the information privacy of all parties (e.g., names, telephone numbers, travel times, origins, and destinations). We review previous similar schemes and compare their various features and properties, for example, whether they require the use of GPS, whether the users need to know each other in advance, whether they are required to register, whether they can protect the user’s personal information, and whether they can be used to arrange spontaneous carpools instantly. We also perform a security analysis to illustrate the privacy controls of the proposed scheme. We discuss how to obtain the quantification space and quantized values from the longitudes and latitudes, explain methods for arranging matches within a given error range, and make recommendations for actual system implementation. We also show that matching will succeed as long as the distance between the two parties falls within a set error distance range and will fail if this distance is exceeded. Finally, we describe the implementation of the proposed scheme on an Android smartphone to demonstrate feasibility. Future work will

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Carpool</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Does not require GPS for matching</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Users do not need to know each other</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Spur-of-the-moment matching</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Does not require registration</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Does not require Internet access</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
<tr>
<td>Privacy protection</td>
<td>v</td>
<td>v</td>
<td>v</td>
<td>v</td>
</tr>
</tbody>
</table>
focus on developing a more accurate method for producing “circular pairing” rather than “square pairing” to reduce error values or error ratios arising from the determination of coordinates.

Notations

- $LL_A$: Latitude and longitude of $A$
- $LL_A(i)$: Quantified latitude and longitude of $LL_A$
- $Adj_A$: Adjustment value of $A$
- $Dst_A$: Destination input for $A$
- $Rng_A$: Destination range for $A$
- $SK_{AB}$: Session key for $A$ and $B$
- Carpooling: Carpool match initiating message
- GoMatch: Match agreement message
- $H(\cdot)$: One-way hash function
- $Q(x, y)$: $x^{(q)} \in \{i p | i = 0, 1, 2, \ldots, p = 2y\}$ s.t. $- y < x^{(q)} - x < y$
- $Map(\cdot)$: Latitude and longitude conversion function.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the NSC under Grant NSC 102-2221-E-182-038. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References


