Pers Ubiquit Comput (2014) 18:143–162
DOI 10.1007/s00779-012-0628-9

ORIGINAL ARTICLE

Efficient location aware intrusion detection to protect mobile devices

Sausan Yazji · Peter Scheuermann · Robert P. Dick · Goce Trajcevski · Ruoming Jin

Received: 27 February 2012 / Accepted: 18 October 2012 / Published online: 18 January 2013
© The Author(s) 2013. This article is published with open access at Springerlink.com

Abstract This paper addresses the problem of efficient intrusion detection for mobile devices via correlating the user's location and time data. We developed two statistical profiling approaches for modeling the normal spatio–temporal behavior of the users: one based on an empirical cumulative probability measure and the other based on the Markov properties of trajectories. An anomaly is detected when the probability of a particular (location, time) evolution matching the normal behavior of a given user becomes lower than a certain threshold, determined by controlling the recall rate of the model of the normal user's behavior. We used compression techniques to reduce processing overhead while maintaining high accuracy. Our evaluation based on the Reality Mining and Geolife data sets shows that the proposed system is capable of detecting a potential intrusion within 15 min and with 94 % accuracy.

S. Yazji (✉) · P. Scheuermann · G. Trajcevski
EECS Department, Northwestern University,
Evanston, IL 60208, USA
e-mail: s-yazji@northwestern.edu

P. Scheuermann
e-mail: peters@ece.northwestern.edu

G. Trajcevski
e-mail: g-trajcevski@northwestern.edu

R. P. Dick
EECS Department, University of Michigan,
Ann Arbor, MI 48109, USA
e-mail: dickrp@eecs.umich.edu

R. Jin
CS Department, Kent State University,
Kent, OH 44242, USA
e-mail: jin@cs.kent.edu

Keywords Mobile security · Trajectory analysis · Data reduction

1 Introduction

Recent technological advancements caused a huge increase in the use of mobile devices. Smart phones, notebooks, and iPads come with many capabilities including email, text messaging, gaming, web browsing, navigation, and recording pictures/videos. These devices store a lot of personal information and, if stolen, loss of control over the data may be more important than the loss of the smart mobile device.

Some prior works on mobile device security have focused on physical aspects and/or access control methods (e.g., strong passwords, voice recognition [26], or fingerprints [21]). However, such approaches do not protect the private data on stolen devices in the post-authentication state. Today's smart devices are already equipped with tools that allow us to obtain vast amount of data about user behavior, such as application usage logs. In addition, many mobile devices are equipped with location identification tools such as Global Positioning System (GPS) receivers, which can be used to track locations in case of theft. However, existing works using GPS-features to protect mobile devices (e.g., GadgetTrak [12] and RecoveryCop [25]) depend on the owner to report the theft, and it may take hours before the owner realizes it, at which point private data may have already been exploited. Even Laptop Cop [23] requires user intervention to remotely/manually delete the data on stolen devices.

Our main goal is to develop efficient techniques for protecting data saved on mobile devices by detecting anomalous spatio–temporal behavior as compared to the

regular motion patterns of the owners. A study performed by González et al. [14] on 100,000 trajectories of anonymized mobile phone users whose positions were tracked for a 6-month period has demonstrated that many individuals tend to have small sets of locations that they visit frequently (e.g. home, work, school) and tend to take the same path when moving between locations. Observations González et al. [14] imply that the user's presence at a certain time in a certain location is predictable—hence, we can utilize this to build a user profile which, in turn, can be used to perform anomaly detection.

In a previous study [34], we used network access patterns and file system activities on laptops to build a behavioral model based on K-means clustering that permitted attack detection with a latency of 5 min and an accuracy of 90 %. In a recent work [35], we used users' location information and trajectory data to build the profile of smart phone users, and we were able to detect attacks within 15 min with 81 % accuracy. This paper extends our results [35] as follows:

1. We present an enhanced user model based on the previously discussed spatio–temporal information and trajectory data approach where we assumed a normal distribution histogram for the user profile. We eliminated the low end of the distribution (lower than 10 % values) during the detection analysis in order to achieve 96 % detection accuracy.
2. We propose, implement, and compare two data reduction techniques that enable us to reduce the memory requirements by $\approx$90 % and consequently reduce the processing time. Those techniques are the *Row-Merge algorithm*, which combines adjacent rows in our data structures and the *MDLP* algorithm, which is an adaptation of an existing statistical technique [3] to our settings.
3. We evaluated our techniques on an additional spatio–temporal data set—Geolife [36–38].

In summary, this article makes the following main contributions.

- We develop two statistical profiling approaches and corresponding representations: one based on empirical cumulative probability measure and the other based on the Markov property, in order to model the normal behavior of a user in a fixed time-window. An anomaly is detected when the probability of a user window reflecting a normal behavior falls below a threshold that is determined by controlling the recall rate of the user's normal behavior.
- We present two techniques that reduce user profile memory requirements while still allowing accurate attack detection.
- We present a detailed experimental evaluation of the proposed methodologies over two data sets, quantifying the benefits of our approaches.

In the rest of this paper, Sect. 2 places the work in the context of our system architecture and discusses the data and feature extraction methods. Section 3 presents the detail of the user profile representation and our anomaly-based detection schemes. Section 4 presents the methods used to reduce the size of the user profile data. Section 5 presents a comprehensive experimental evaluation of our methods. Section 6 describes related work and Sect. 7 concludes the paper and indicates directions for future work.

2 Preliminaries

We now give an overview of our system architecture, followed by discussion of the properties of the data and their use in feature extraction.

Our system for automatic generation of mobility models and detection of spatio–temporal behavioral anomalies is based on a client–server architecture utilizing cloud computing. Its main modules are (1) data collection, (2) feature extraction, (3) user profile/model building, (4) data reduction, and (5) anomaly detection. The detection accuracy will be determined by which anomalous behavior can be distinguished using such models and considering other users' models for anomaly detection; Fig. 1 illustrates the integration of these modules into our system architecture, which consists of the following sub-systems:

(**ICS**)—the *information capturing system*, which resides on the mobile device, contains an application to track the device location, register it periodically, and save it in a new log file every T minutes. It also contains the feature extraction module.

(**IMS**)—the *information management system*, which collects the log-files from the ICS and resides on a computer with higher performance and much looser power consumption constraints than the mobile device. It is responsible for building mobility models and performing anomaly detection. Upon building the user model, the IMS, possibly after the data reduction process, sends the user model to the mobile device, allowing local detection of attacks in the absence of wireless connection.

(**RMS**)—the *response management system*, which resides on both the mobile device and the remote server hosting the IMS. Upon receiving an alert, the RMS identifies the appropriate action to protect data on the mobile device, for example, notifying the device owner, locking the device, or automatically deleting private data.

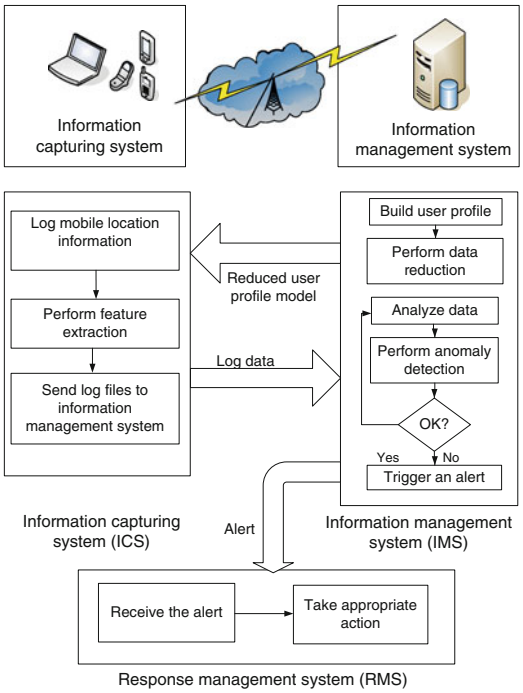

Fig. 1 System architecture

Our work focuses on the algorithms and implementations for the ICS and the IMS modules, since the RMS consists of user-dependent actions to be executed upon actual detection of an attack. Again, the rationale is to maximize the extent to which the mobile devices themselves can detect the anomalous spatio–temporal behavior. While the data structures representing the user motion are built at the server, in the case of transient network failure, classification can still be performed on the client using the most recent transmitted matrix. Clearly, this may affect the classification accuracy if the network connection is not available for a prolonged period of time.

2.1 Mobility profiles

We now present our setup for the data collection and the feature extraction modules.

2.1.1 Data collection

Motion traces are essential for model construction and anomaly detection. To obtain them, motion monitoring software needs to be developed to collect information about each user's motion patterns—that is, his spatio–temporal data (along with the other user activities such as file system access and network activities). These are saved as trace files, to be sent to the IMS system periodically at pre-determined intervals.

In our initial system implementation, we relied on the fact that a number of researchers gathered vast amount of motion traces and they are publicly available [11, 14, 27, 28, 38]. We note, however, that some of these traces were collected for reasons different from ours, with different experimental settings and requirements.

Our desiderata can be summarized by the following properties, abbreviated as (LCF):

- *longevity* **(L)**: collected for a long period of time, continuously;
- *consistency* **(C)**: collected at regular times (e.g., same times daily); and
- *high frequency* **(F)**: to support fast anomaly detection.

After analyzing the different available traces, two data sets—Reality Mining data set [11] and Geolife [38]—turned out to provide closest match for the (LCF) properties.

The Reality Mining data set contains traces collected over a 9-month period for over 100 users, consisting of phone calls logs, locations identified by tower IDs and area IDs, event logs, and device-specific data such as the device specs. The collection interval ranged from a few seconds to 15 min, with an average of 2.5 min (except when the mobile device was off) at regular time-instants daily.

The Geolife data set is a collection of GPS trajectories for 178 users in a period of over 4 years. The data was recorded with high frequency where 91 % of the trajectories are logged every 1–5 seconds or every 5–10 meters per point. With closer examination, we noticed that about 50 % of this data set is also compliant with the LCF properties.

2.1.2 Feature extraction

Reality Mining data set: The traces have over 55 data features capturing information about the users' mobility, activity, communication events, reporting time, and device-specific information such as the MAC address and the device maker. Since we focus on the spatio–temporal and trajectory features, properties like user activity, device-specific information, user communication style, and user affiliation information were not considered.

Reality Mining data provides three values to represent a location: cell tower ID, area ID, and area name. The cell tower ID gives information associated with user's location—therefore, it is a source of information for the user's movement over time. However, the tower ID information in the Reality Mining data set has no geographical coordinate information, and since each physical location could be associated with multiple tower IDs, we consider the tower ID as unreliable feature. Thus, we have selected the area ID to represent the location information in our study. Area ID represents the physical location (Library, Office, etc.) identified either by the information capturing system

(ICS) itself or by the user when reporting new locations such as home, office, restaurants, etc.

Our spatio–temporal analysis techniques depends on extracting the following features from the Reality Mining log:

1. (u_i)—User ID;
2. (l_j)—Location information, represented by the area ID in the traces; and
3. (t_k)—Timestamps of the data records in the trace.

Thus, our input data records are tuples of the form (u_i, l_j, t_k).

Geolife data set: These traces have a smaller number of features—only seven of them, including the longitude, latitude, and altitude information in addition to the date and time information, and the transportation mode (car, bus, walking, ...).

We note that this data set was collected in 30 different cities in China, United States, Korea, and Europe, and we focused on the trajectories that were collected in same cities.

Examining the row data directed us to think that most study users started at the location with 39.0°–41.0°, 115.5°–117.5°) coordinates and then moved to different areas by bus, train, plane, or boat. This location represents Beijing (China [31]). We focused on an area of (138 × 110) square miles [24].

To utilize this data set, the GPS location information needed to be mapped into an area ID, so that the structure is similar to the Reality Mining data set representation. The longitude and latitude information is provided by degrees, with precision up to (0.000001°). In the GPS system, at 39° latitude, any change on the longitude to the (±0.0001°) digit represents ≈8 m, while at the 116° longitude, the same change in the altitude represents ≈7 m. Therefore, in this data set, we rounded the coordinate numbers to the closest fourth decimal digit and have each coordinate pair represent an area ID, again having records/tuples of the form (u_i, l_j, t_k).

3 Data models and anomaly detection

We developed two statistical profiling approaches in order to model the normal behavior of a user in a fixed window. Model #1 is based on the empirical cumulative probability measure of location and time, while Model #2 is based on the Markov transition property. In this section, we describe each of them in detail.

3.1 Model #1: Location-in-time probability measure

In Model #1, for each user u_i, we extract the location l_j and timestamp t_k. For conciseness, we will sometimes neglect notation for user ID when it is clear from the context.

3.1.1 Building the user profile

Since our goal is to detect attacks by detecting deviation from the user's normal behavior, the first step is to develop a model of a user's normal behavior based on the set of locations that the user has visited during the data collection period. To build the respective user profiles for each user in the data set, we divided the data logs evenly into two consecutive data sets: *model_data* (used for model construction) and *test_data* (used for evaluation).

Utilizing the *model_data*, a user profile was constructed as follows:

1. For each user u_i, we extracted the distinct locations and kept track of them in a list (L_i).
2. We built a table of $|L_i|$ columns and NT rows to save the location probability values, where NT stands for the number of minutes in a day.
3. We calculated the probability $\text{Prob}_i(t_k, l_j)$ that represents the fraction of time in the *model_data* in which the user u_i was at location l_j at time t_k, where $1 \leq j \leq |L_i|$, and $1 \leq k \leq NT$. Recall that at any given time t_k, the user u_i should be at some unique location l_j from the location list L_i

$$\forall t_k \in NT,\ \exists\, l_j \in L_i \quad \text{where } \text{Prob}_i(t_k, l_j) > 0 \tag{1}$$

4. We extracted from the user distinct location list L_i the user's common locations list (UCL_i), which consists of locations that the user has visited more than 1 % of the time during the data collection period. All locations that have been visited less than 1 % of the time will be saved in the Infrequent list (IF_i) in order to be able to delete all related records from the model_data, and the respective columns from the user profile.

$$\forall l_j \in L_i :$$
$$\text{if } \sum_{k=1}^{NT} \text{Prob}_i(t_k, l_j) \geq 0.01 \text{ then } l_j \in UCL_i$$
$$\text{if } \sum_{k=1}^{NT} \text{Prob}_i(t_k, l_j) < 0.01 \text{ then } l_j \in IF_i$$

We selected the value of 1 % based on the study performed by Bayir et al. [4], which reported that individuals spend 79–85 % of their time in small number of locations (2–8), and less than 15 % of their time in large number of locations that they have visited only less than 1 % of the time. We observed that the fraction $|IF_i|/|L_i|$ can be significant, and hence, keeping track only of UCL_i is a first step toward reducing the storage costs.

	l_1	l_2	l_3	l_4	l_5
12:00 AM	0.04	0	0.011	0	0
12:01 AM	0.04	0	0.01	0	0
12:02 AM	0.03	0.01	0.02	0	0
12:03 AM	0.032	0	0.021	0	0
12:00 PM	0	0.01	0.01	0	0.03
11:58 PM	0.029	0	0.021	0	0
11:59 PM	0.04	0	0.019	0	0

Fig. 2 User profile for Model #1

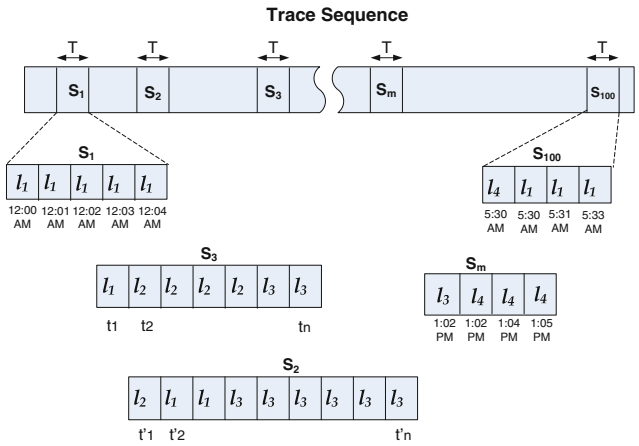

Fig. 3 Example of dividing the *test_data* set into 100 test samples

5. We eliminated the least visited locations from the profile table, and obtained a final user profile size of $|UCL_i| \leq |L_i|$ columns and NT rows.
6. We create a discrete probability distribution by counting visits to the common locations and then normalizing so they sum to one.

$$\sum_{k=1}^{NT} \sum_{j=1}^{|UCL_i|} LOC\text{-}IN\text{-}TIME_i(t_k, l_j) = 1 \quad (2)$$

Figure 2 shows an example user profile represented as a two-dimensional matrix with $(NT \times |UCL_i|)$ elements. Rows (t_k) correspond to the minutes of the day (12:00 AM, 12:01 AM, …, 11:59 PM) and columns (l_j) correspond to locations $(l_1, l_2, l_3, l_4, l_5)$. Each cell in the user profile represents the weighted probability $LOC\text{-}IN\text{-}TIME_i(t_k, l_j)$.

For example, this user profile shows that the user has never been in locations l_2, l_4, or l_5 at 12:00 AM during the data collection period, while 4 % of the data collection time he was at location l_1. Please note that while reading the timestamp in the mobility trace, we consider the hour and the minute values only; therefore, each row in the user profile represents the minute of the day plus 59 seconds.

3.1.2 Anomaly detection

The anomaly detection process is responsible for receiving streams of user mobility data, comparing them with the user profile, and identifying an anomaly (potential theft of the mobile device).

Our anomaly detection scheme falls into the class of statistical methods [7] which are based on the assumption that normal data instances occur in a high probability measure of the stochastic model while anomalies occur in the low probability region of the stochastic model. Our scheme is a nonparametric collective anomaly detection model, where the probability values are extracted from the traces and an anomaly represents an unusual sequence of data.

The first step of our collective anomaly detection scheme is to randomly select 100 samples $(S_1, S_2, S_3, \ldots, S_m, \ldots, S_{100})$ from the *test_data* set, for which the time span is T minutes as shown in Fig. 3.

A random sample S_m of time span T corresponds to a contiguous sequence of records: (u_i, l_j, t_k), (u_i, l_{j_1}, t_{k_1}), …, (u_i, l_{j_x}, t_{k_x}), …, (u_i, l_{j_n}, t_{k_n}) satisfying these three conditions:

- $t_k \leq t_{k_1}, \ldots, \leq t_{k_x}, \ldots, \leq t_{k_n}$
- $(t_{k_n} - t_k) \leq T$
- $(t_{k_{n+1}} - t_k) > T$

The number of records per sample varies among samples due to the variation in data collection interval.

For each sample S_m, we define the empirical cumulative probability P_{S_m} of the records in the sequence using the probability distribution table established during the profile building phase and based on the *model_data* representative of the user u_i as follows:

$$P_{S_m} = \sum_{(k,j) \in S_m} LOC\text{-}IN\text{-}TIME_i(t_k, l_j) \quad (3)$$

As an example, let us consider the sample S_1 as shown in Fig. 4. To calculate the P_{S_1} value, we check the user profile illustrated in Fig. 2 and we extract the values for each corresponding record from the representative user profile. Therefore,

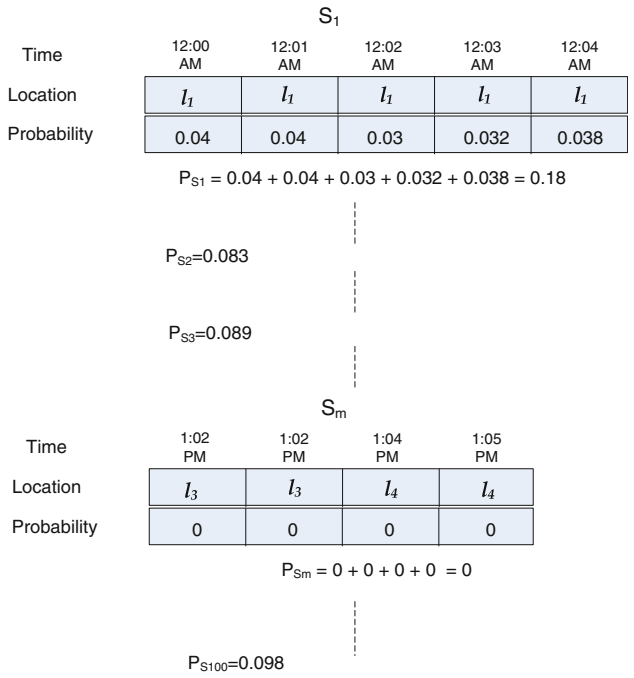

Fig. 4 Example of calculating P_{S_m} value

$$
\begin{aligned}
P_{S_1} = {} & \mathit{LOC\text{-}IN\text{-}TIME}_i(12:00AM, l_1) \\
& + \mathit{LOC\text{-}IN\text{-}TIME}_i(12:01AM, l_1) \\
& + \mathit{LOC\text{-}IN\text{-}TIME}_i(12:02AM, l_1) \\
& + \mathit{LOC\text{-}IN\text{-}TIME}_i(12:03AM, l_1) \\
& + \mathit{LOC\text{-}IN\text{-}TIME}_i(12:04AM, l_1)
\end{aligned}
$$

$$P_{S_1} = 0.04 + 0.04 + 0.03 + 0.032 + 0.038 = 0.18$$

We calculate the P_{S_m} values, essential for defining the user trust value, for all 100 samples as illustrated (cf. Fig. 4) similarly.

We are now ready to relate the time span of a sample derived from the model data to the detection delay:

Definition 1 Detection delay (T) is the shortest length (measured in time) of the trace generated by the mobile device that would allow the system to distinguish among users with an acceptable accuracy rate.

The detection delay T equals the time span of the user samples discussed above and the incoming data stream windows need to cover also the same time span T.

Definition 2 Trust value (P_{trust}) for Model #1 is the empirical cumulative probability of samples of span T that represents a confidence interval of 90 % based on the user profile. All data stream windows with cumulative probability less than P_{trust} are considered attacks.

Attacks are detected via mismatches between the data stream windows and the samples conforming to the normal user behavior, yielding an attack detection delay T. When the empirical cumulative probability of a specific data stream window drops below the *Trust value* (P_{trust}), our system concludes that the mobile device is used by someone other than its owner, or what we call in this paper, the device is under attack.

Definition 3 False acceptance rate (FAR) is the percentage of the attack data stream windows that are accepted by the system as normal user behavior.

Definition 4 False rejection rate (FRR) is the percentage of the user's normal data stream windows that are identified by the system as an attack.

We focus on the FAR and FRR values—an ideal system should have FAR = FRR = 0. Yet errors are possible since human mobility traces can deviate from the calculated profile from time to time. Therefore, our goal is to associate with every user a P_{trust} value that strikes a good balance between the FAR and FRR values.

In the example illustrated in Fig. 4, the smallest P_{S_m} value equals zero; therefore, setting P_{trust} also equals to the smallest P_{S_m} value, implies that the system will accept every incoming stream window and treat as acceptable user behavior. Subsequently, in this case, we obtain FAR = 100 % and FRR = 0 %.

Figure 5 shows the relationship between precision and recall as examined in our data set, where $(x =)recall = 1 - \text{FRR}$ and $(y =)precision = 1 - \text{FAR}$. We observe that the high recall, say 0.9, implies a small FRR (0.1) and large FAR (0.8) values. As we decrease the recall, the FRR values increase, for example, for recall = 0.7, we get FRR = 0.3 and correspondingly FAR = 0.6.

Our heuristic starts with the observation that the histogram of the cumulative probabilities for the 100 traces of each user is close to the histogram for user u_{92} as shown in Fig 6. Next, we choose a P_{trust} value for each user that guarantees FRR $\leq$ 10 % which corresponds to accepting 90 % of the user's normal behavior based on the trace samples. If we consider P_{S_m} to be a random window cumulative probability, then the range from the determined P_{trust} value to one forms a 90 % confidence interval for

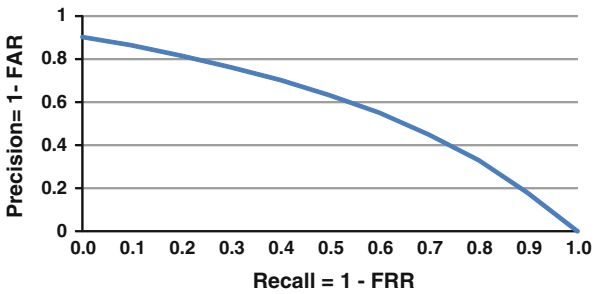

Fig. 5 The relationship between precision and recall

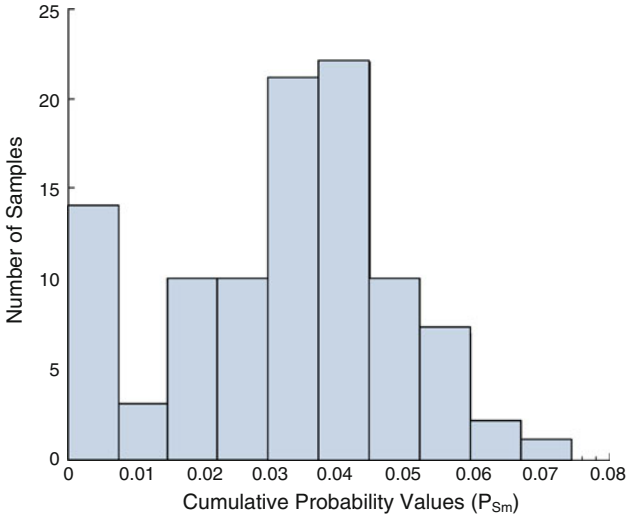

Fig. 6 The histogram of cumulative probability for user u_{92}

P_{S_m}. Intuitively, for FRR $\leq$ 10 %, the P_{trust} score can maximally discover behavior anomalies (corresponding the true attack) with a very small false alarm rate.

After calculating the P_{trust_i} for each user, the anomaly detection process can start, formally described by Algorithm 1.

Algorithm 1 : Detect Mobile Theft Based on Location-in-Time Probability Measure

```
INPUT: LOC-IN-TIME_i
INPUT: User data streams
OUTPUT: Alarm in case of attack
Let EOS =end of stream
while EOS = false do
    Receive data stream window SW_m of size T
    Initialize the Trajectory Probability value P_SWm
    P_SWm = 0
    Size_SW = number of records in the obtained user data stream
    for n = 1 to Size_SW do
        Read the user trajectory record (t_kn)
        if l_n ∈ UCL_i > 0 then
            Read the probability value LOC-IN-TIME_i(t_kn, l_jn) from the user profile
            Calculate the cumulative probability value for the trace P_SWm = P_SWm + LOC-IN-TIME_i(t_kn, l_jn)
        end if
    end for

    if P_SWm < P_trust_i then
        Trigger an alarm
    else
        Continue
    end if
end while
```

3.2 Model #2: Markov-based transition probability matrix

Model #2 is a collective anomaly detection scheme that makes use of them Markov chain stationary property. In our model, states correspond to tuples of the form (u_i, t_k, l_j). The Markov property in our context means that the probability of a user u_i moving to location $l_{j'}$ at time t_k does depend on previous location l_j visited by u_i in the *model_data*.

Conceptually, the user's location–duration trace is divided into sequences, that is, trajectories. Each trajectory consists of a start point (*SSP*), a number of intermediate points, and an end point *SEP*, and may differ semantically due to the notion of stopping time T_{STP} which is defined as the time interval during which the user is stationary. Based on observations from other researchers [32], we use $T_{STP} = 30$ min.

3.2.1 Building user profile

For Model #2, the user profile is a three-dimensional table *LOC-TIME-MOVE*$_i$. Each entry in this table, *LOC-TIME-MOVE*$_i(t_k, l_{j'}, l_j)$, represents the weighted probability of the user u_i moving from location l_j to location $l_{j'}$ at time t_k.

Figure 7 represents the state diagram corresponding to a trace of user sequences starting at location l_1 at time t_1. The nodes stand for the identified (locations,time) tuples while the edges represent moves from a location to another weighted by the probability of the transition. Note that for conciseness, we did not store the time information in the nodes of the state graph, but are showing it in the rightmost column.

This state graph illustrates that if the user u_i was at location l_1 at time t_1, there is 50 % probability that the user will stay at the same location l_1 at time t_2, 20 % probability to go to location l_2, 25 % probability to go to location l_3, and 5 % probability to go to location l_4, while the user has never travelled to locations l_5 or l_6 from location l_1 at time t_2 during the data collection period. At time t_3, the user who ended up at location l_1 at time t_2 has 70 % probability to

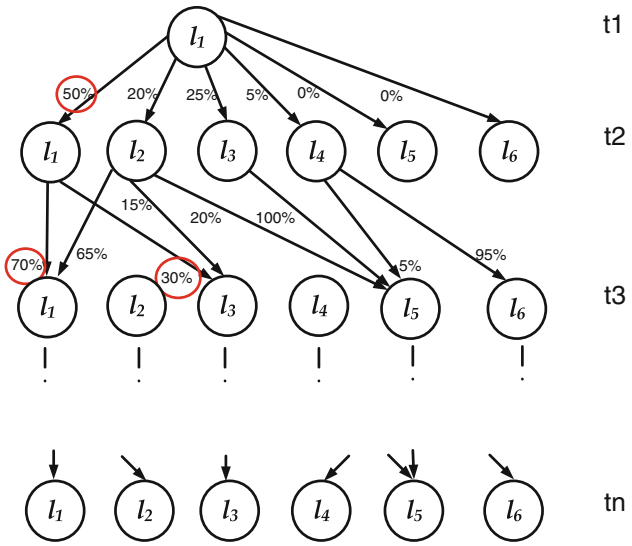

Fig. 7 State graph representing the user sequences when the user starts at location l_1 at time t_1

stay at location l_1, and 30 % probability to go to location l_3, and so on.

To build the user profile utilizing the three-dimensional data structure, we perform the following tasks:

1. Read the *model_data* set.
2. Build a list of the user's distinct locations (L_i).
3. Build and initialize a three-dimensional matrix of size ($NT \times |Li| \times |Li|$) where each 2-D plane represents a different starting location for a trajectory.
4. Identify the first record in the data set and keep track of its timestamp t_1 and location l_1. This location becomes the Starting Point, *SSP*, for this trajectory. Increase the frequency value and calculate the probability value $\text{Prob}_i(t_1, l_1, l_1)$.
5. Read the time stamp and the location of the next record t_2,
   - If $t_2 - t_1 \geq T_{STP}$, then this record will be considered a new *SSP* for a new trajectory, and the previous point is an *SEP* for the previous trajectory. Go to Step 5.
   - If $t_2 - t_1 < T_{STP}$, we increase the frequency value by one and calculate the probability value $\text{Prob}_i(t_2, l_2, l_1)$.
6. We repeat the task #4 until we reach the end of the data set.
7. Create the UCL_i list by eliminating all locations that the user visited less than 1 % of the time, (same reason as indicated in Sect. 3.1).
8. Create the new user profile with the *NT* rows, $|UCL_i|$ columns, and $|UCL_i|$ depth.
9. Replace the probability value, $\text{Prob}_i(t_k, l_{j'}, l_j)$, in each cell in the new user profile with the weighted probability value, $LOC\text{-}TIME\text{-}MOVE_i(t_k, l_{j'}, l_j)$.
10. The final user profile will have the sum of each row in each starting location matrix equals to one.

$$\forall l_{j'} \in UCL_i \text{ and } \forall k \in NT$$
$$\sum_{j=1}^{UCL_i} LOC\text{-}TIME\text{-}MOVE_i(t_k, l_{j'}, l_j) = 1 \quad (4)$$

Upon completing this process for all the records in the *model_data* set, the user profile will be produced as a three-dimensional matrix as shown in Fig. 8. The rows represent the minutes of the day where each minute is a row, the columns represent the locations from the UCL_i list, and the depth represents the starting locations from the UCL_i.

For example, if we consider the user trajectory represented in the Fig. 7, the associated user profile would be the one shown in Fig. 8. In this example, the user starts at location l_1 at time t_1. The cell (t_2, l_1, l_1) indicates the weighted probability value for the user to be at location l_1 at time t_2, when he was at location l_1 in the previous record, $LOC\text{-}TIME\text{-}MOVE_i(t_2, l_1, l_1) = 50$ %, and the cell (t_2, l_2, l_1) indicates the probability value for the user goes to location l_2 at time t_2, when he was at location l_1 in the previous record, $LOC\text{-}TIME\text{-}MOVE_i(t_2, l_2, l_1) = 20$ %, and so on.

3.2.2 Anomaly detection

The anomaly detection process for Model #2 follows the same principle as in Model #1 anomaly detection process,

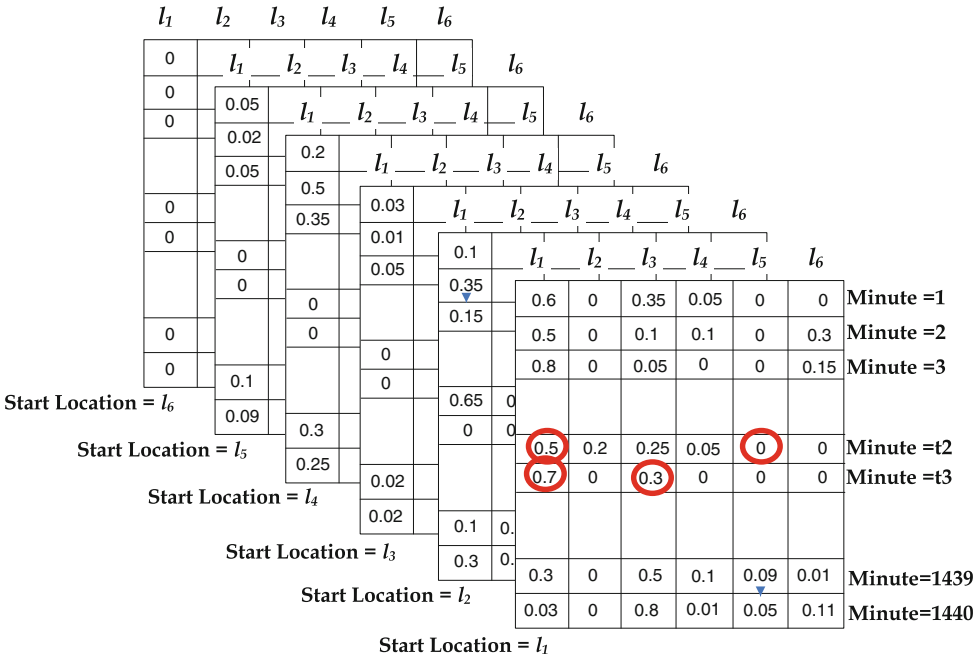

Fig. 8 Mobility model for user u_i (Model #2)

where we focus on the evaluation of P'_{trust}, FAR′, and FRR′ to identify the system ability to detect attack. The calculation for FAR′ and FRR′ is the same as explained in Sect. 3.1.2, but the computation of trust values P'_{trust} for each user is different.

Definition 5 The trust value (P'_{trust}) for Model #2 is the trace joint probability value that represents a confidence interval of 90 % based on the user profile. All traces with probability value less than P'_{trust} are considered attacks.

For Model #2 and since we capture the user movement, where the probability of the user to be in any location at any time is highly dependent on the previous location at the previous point of time, we calculate the Markov sequence probability value for each trace sample (S_m) rather than the cumulative probability value as follows:

$$P'_{S_m} = \prod_{(k,j,j')\in S_m} LOC\text{-}TIME\text{-}MOVE_i(t_k, l_{j'}, l_j). \quad (5)$$

The joint probability value is the product of the probabilities of all records in the trace sample S_m, as indicated in the *LOC-TIME-MOVE* table. Equation 5 shows that if any record in the sequence has a probability of zero, which corresponds to the fact that the user has never moved between these two locations at this time before, the whole trace will be considered an attack because the $P'_{S_m} = 0$. To reduce the penalty for deviation from the normal path, we introduce the concept of Trace Threat Level (TL), which represents the percentage of the records in the trace that has no representation in the user profile. Thus, if $LOC\text{-}TIME\text{-}MOVE_i(t_k, l_{j'}, l_j) = 0$, we eliminate this value from the calculation of the trace joint probability value and increase the Threat Level value by one. We use a threat level threshold of $TL_{trust} = 10$ % of the total records in the trace, based on empirical analysis.

As an example, Fig. 9 shows two paths; the solid curve represents the normal path in the user's profile and the dashed curve represents the currently detected trajectory. In this example, the user profile indicates that when the starting point at time t_1 is location l_2, the normal path of duration T is $l_2 \rightarrow l_3 \rightarrow l_4 \rightarrow l_5 \rightarrow l_6 \rightarrow l_7$. In contrast, the captured user trajectory that starts at location l_2 at time t_1 consists of the sequence $l_2 \rightarrow l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \rightarrow l_5 \rightarrow l_6$. To determine whether this is an expected or anomalous user behavior, we compare the joint probability of this path with the profile of the particular user. The calculated value should be equal to or greater than the trust value for that user.

To calculate the captured trajectory joint probability P'_{SW_m} as indicated in Eq. 5, we first identify the starting point $SSP = l_j$ and the time t_k. (Please note that we use P'_{SW_m} to indicate *trajectory probability* which is used to perform the anomaly detection process, and P'_{S_m} to refer to *sample probability* which is used for the calculation of P'_{trust} value). Then, we check whether $l_j \in UCL_i$ or not. If not, we increase the threat level TL value by one. Otherwise, we identify this location, l_j as the starting location, and check the value $LOC\text{-}TIME\text{-}MOVE_i(t_k, l_j, l_j)$ to calculate the trace joint probability value P'_{SW_m}. Next step is to identify the next record, and read the data $l_{j'}$ and $t_{k'}$ from that record. If $l_{j'} \in UCL_i$, we obtain the weighted probability value $LOC\text{-}TIME\text{-}MOVE_i(t_k, l_{j'}, l_j)$. If not, we increase the TL value again. This process is repeated for the entire user trace and, upon completion, we check whether $TL \geq TL_{trust}$ or not. If it is greater than TL_{trust}, this trajectory is judged to have been generated by someone other than the user, that is, an attacker. If not, we subsequently check the P'_{SW_m} value. If $P'_{SW_m} \geq P'_{trust_i}$, the trajectory is judged to belong to the user; otherwise, it is treated as a trajectory generated by an attacker.

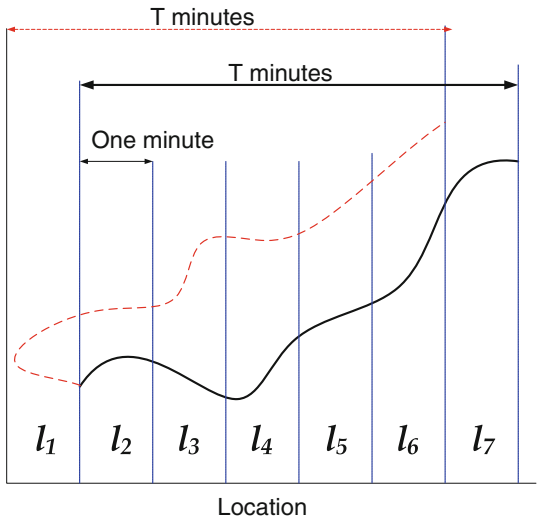

Fig. 9 User path analysis

4 Data reduction

We now aim to further improve the efficiency by reducing the size of the user model, with low impact on the accuracy of the attack detection. The reduction benefits are twofold:

1. reduce the amount of memory occupied by the user model, and
2. reduce the CPU time required to perform the detection process and therefore enhance the system performance.

The Reality Mining data set consists of 93 users with total number of distinct locations is in the range 3–100, and average of 28 locations. While the Geolife data set has 65 users with total number of distinct locations 134–1,200.

For these data sets, the user profile requires up to $(1{,}440 \times 1{,}200) \approx 10^6$ memory locations for Model #1

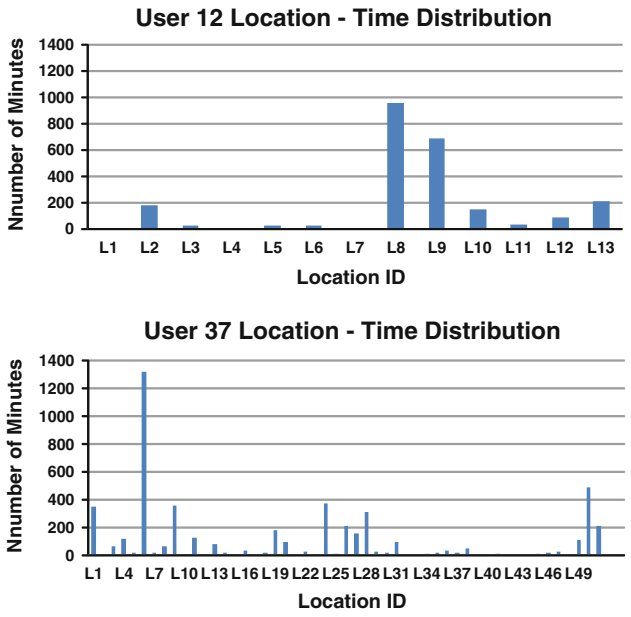

Fig. 10 The average number of minutes per day the user spent in each location

and up to $(1{,}440 \times 1{,}200 \times 1{,}200) \approx 10^9$ memory locations for Model #2. We propose and analyze two different solutions to reduce the size of the models' representations:

1. The first is called the Row-Merge algorithm, where the rows in the user model table $LOC\text{-}IN\text{-}TIME_i$ are combined if they fit this condition: For each $t_k \in NT$:
$$\forall l_j \in UCL_i \,(\text{Count}\, LOC\text{-}IN\text{-}TIME_i(t_k, l_j) \neq 0) \geq ThV, \quad (6)$$
where ThV is a threshold value that guarantees no detection degradation occurs due to the Row-Merge process.
2. The second algorithm is based on the MDLP (Minimal Description Length Principle) [3, 18].

We now present in detail the algorithms along with their complexity analysis.

4.1 Row-Merge algorithm

As discussed in Sect. 3.1, three properties identify a profile:

- At any minute t_k, user u_i can exist in any location l_j identified in the L_i list.
- At any minute t_k, each user u_i has to be located in one of the locations l_j that are identified in the distinct locations list L_i. Yet, by considering the UCL_i list rather than the L_i list, our user profile will have certain time of the day that is not accounted for therefore the new rule is as follows:
$$\forall l_j \in IF_i, \exists\, t_k \quad \text{where}\, \text{Prob}_i(t_k, l_j) = 0.$$
- Based on Model #1, the sum of all the probability values in every user profile is equal to 1:
$$\forall u_i, \sum_{k}^{NT} \sum_{j}^{|UCL_i|} LOC\text{-}IN\text{-}TIME_i(t_k, l_j) = 1.$$

Observation: Although there is no rule preventing a user from being at any place at any time, the patterns indicated different findings. The user profile shows that at a certain time of the day (late night to early morning for example), some users are always at one place, and the probability value for the rest of locations equals zero.

Figure 10 shows the distribution of the time the user spends in each location every day, and it indicates that each user has only few locations that he spends most of the time at. For instance, although user u_{12} has 13 distinct locations in his profile, there are only three locations u_{12} visits every day (l_2, l_8, l_9). Similarly, user u_{37} has 51 distinct locations in UCL_{37}, while he spends most of his time in seven locations (l_1, l_6, l_9, l_{19}, l_{25}, l_{28}, l_{50}).

Based on the above observation, we conclude that if we consolidated the time periods where a given user has been in few locations (less than ThV) into one row, we could significantly reduce the size of the matrix representing that user's profile. The main idea is to consolidate consecutive rows where the total number of $LOC\text{-}IN\text{-}TIME_i(t_k,l_j) = 0$ in that row is less than ThV. The Row-Merge approach is formalized by Algorithm 2.

Algorithm 2 : Data Reduction, Row–Merge Algorithm

```
INPUT: LOC-IN-TIME_i
OUTPUT: Reduced_Model_Data(RMD_i)
OUTPUT: Time_Index_i
Allocate a matrix with one column and NT rows
Read the first line from LOC-IN-TIME_i
Initialize a record counter RN = 1
Initialize the time–index Time_Index_i[1] = 1
for k = 2 to NT do
   Read a row k from the data set LOC-IN-TIME_i
   EL = Count(LOC-IN-TIME_i(t_k,:) ≠ 0) {Identify the
   number of cells in the row k that are ≠ 0}
   if EL ≤ ThV then
      for j = 1 to |UCL_i| do
         LOC-IN-TIME_i(RN, l_j) = LOC-IN-TIME_i(RN, l_j)
         +LOC-IN-TIME_i(t_k, l_j) {Merge the rows}
      end for
   else
      RN = RN + 1
      Time_Index_i[RN] = k
      for j = 1 to |UCL_i| do
         LOC-IN-TIME_i(RN, l_j) = LOC-IN-TIME_i(t_k, l_j)
      end for
   end if
end for
Eliminate all rows in Time_Index_i that equal zero
Allocate matrix RMD with RN rows and |UCL_i| columns
New_t_k = 1
for k = 1 to RN do
   Read a row k from the data set LOC-IN-TIME_i
   RMD_i(New_t_k, l_j) = LOC-IN-TIME_i(t_k, l_j)
   New_t_k ++
end for
Done
```

The data input is the user profile consisting of NT rows and $|UCL_i|$ columns, while the output is a matrix with

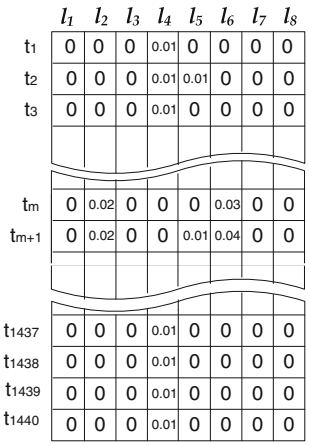

Fig. 11 Example user profile

reduced number of rows *RN* and $|UCL_i|$ columns. The first step is to read the user profile. For each row, we count the total number of cells that are greater than zero as illustrated in Line 11. If the count is less than the threshold value *ThV*, then this row will be merged with the previous one in the $LOC\text{-}IN\text{-}TIME_i$ table as shown in Line 13. If the count of probability values that are greater than zero is greater than *ThV*, we keep this row and update the $Time\_Index_i$ as shown in Lines 15–21. Upon completion, we initialize the $Reduced\_Model\_Data(RMD_i)$ with the new number of rows as shown in Line 24 and save the new user model in this matrix as illustrated in Lines 26–30. The complexity of this algorithm is $O(NT \times |UCL_i|)$.

As an example, assume that Fig. 11 represents a user profile for user u_i with eight distinct locations $l_1, l_2, \ldots, l_8$. The size of this user profile is $1{,}440 \times 8 = 11{,}520$ numeric values. In this example, we illustrate the Row-Merge process for a threshold value $ThV = 1$ which indicates that all rows in the user profile that have at most one probability value that is greater than zero will be merged with the previous row. Figure 12 details this process.

The total reduction in the matrix size is equal to the total eliminated rows $(1{,}440 - Q)$ multiplied by 8, the total number of distinct location.

4.2 MDLP algorithm

In this section, we aim to compress the size of the user model by applying the Minimum Description Length Principle (MDLP), based on the following insight: *any regularity in the data can be used to compress the data, that is, to describe it using fewer symbols than the number of symbols needed to describe the data literally. The more regularities there are, the more the data can be compressed* [15].

Observation: The challenge in this approach is to identify regularity in our user model that is not completely regular;

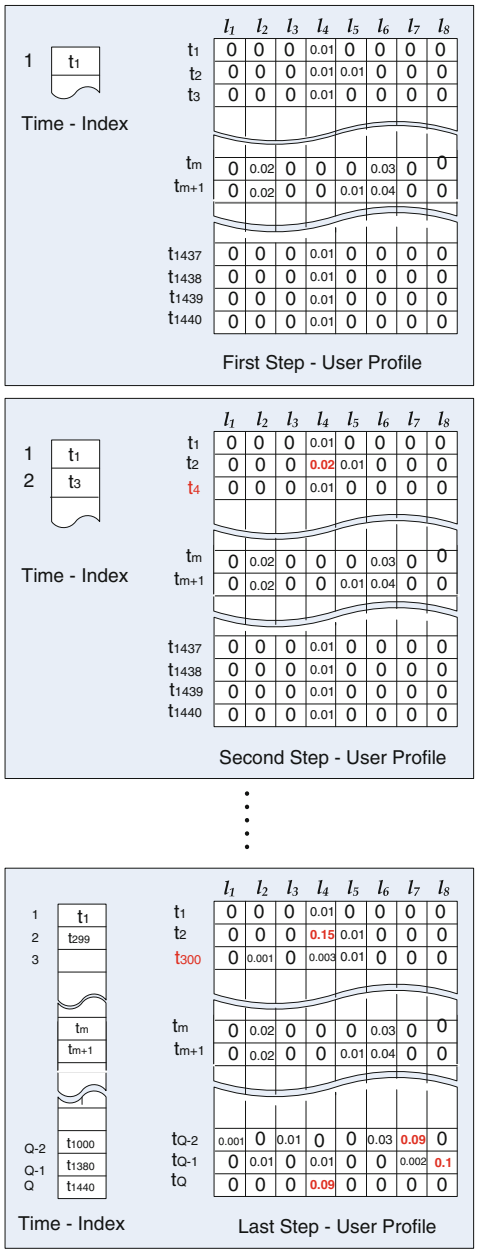

Fig. 12 Row-Merge process

there is no single subset $LOC\text{-}IN\text{-}TIME''_i$ in the user model $LOC\text{-}IN\text{-}TIME_i$ that we could identify as a hypothesis to regenerate the user model. In fact, it is acceptable to assume that our user model could consist of several subsets $LOC\text{-}IN\text{-}TIME''_{i_1}, LOC\text{-}IN\text{-}TIME''_{i_2}, \ldots, LOC\text{-}IN\text{-}TIME''_{i_r}$ that combined could be utilized to regenerate the user model $LOC\text{-}IN\text{-}TIME_i$ and therefore are used for model compression. On the other hand, we understand that there will always be some regular data sets in our user model which we will not be able to compress.

The Algorithm: The single user model for u_i is $LOC\text{-}IN\text{-}TIME_i$, where each row corresponds to a minute in the day

t_k, and each column corresponds a user distinct location l_j, and each entry $LOC\text{-}IN\text{-}TIME_i(t_k, l_j)$ records the probability for the given user u_i to visit the location l_j at time t_k. Given this, we focus on the temporal regularity in this data set in order to perform our user model compression. We would like to find a function H_i that partitions the time into consecutive intervals, based on user model matrix, and minimizes the loss when combining/merging two rows.

To facilitate our discussion, we assume that we maintain the *row sum* ($R_i(k)$) and the *time frequency* ($T_i(k)$), for each row k in the user profile $LOC\text{-}IN\text{-}TIME_i(t_k, l_j)$, where

$$R_i(k) = \sum_{j=1}^{|UCL_i|} LOC\text{-}IN\text{-}TIME_i(t_k, l_j) \tag{7}$$

and $\forall\, records \in model\_data$:

$$T_i(k) = Record\_Count \quad \text{where } (t_{k-1}) < t_k \leq (t_{k+1}) \tag{8}$$

The *time frequency* represents the number of records in the data log that represents each time interval, for one minute intervals.

Then, the function $H_i(k)$ for each row in the user u_i profile is calculated as

$$\begin{aligned} Red_i(t_k, l_j) &= \frac{LOC{-}IN{-}TIME_i(t_k, l_j)}{R_i(k)} \text{ and} \\ H_i(k) &= -\sum_{j=1}^{|UCL_i|} \big(Red_i(t_k, l_j)\big) \times \log\big(Red_i(t_k, l_j)\big). \end{aligned} \tag{9}$$

Clearly, this resembles the well-known *entropy*—a common measure for the information loss [15]. Combining two consecutive rows k and $k + 1$ and merge them into one interval $k, k + 1$, the cell value for each cell in the model is

$$\begin{aligned} LOC\text{-}IN\text{-}TIME'_i(t_k, l_j) = {} & LOC\text{-}IN\text{-}TIME_i(t_k, l_j) \\ & + LOC\text{-}IN\text{-}TIME_i(t_{k+1}, l_j) \end{aligned}$$

and $R'_i(k) = R_i(k) + R_i(k + 1)$; therefore, the function $H_i(k, k + 1)$ is calculated as

$$H_i(k, k+1) = -\sum_{j=1}^{|UCL_i|} \big(Red_i(t_k, l_j)\big) \log\big(Red_i(t_k, l_j)\big). \tag{10}$$

The information loss is measured by the difference between the entropy when we combine two rows into one and the combined weighted entropy sum from the two rows

$$\begin{aligned} & T_i(k) \times H_i(k) + T_i(k+1) \times H_i(k+1) - (T_i(k) + T_i(k+1)) \\ & \quad \times H_i(k, k+1) + (|UCL_i| - 1) \times logDS. \end{aligned}$$

where ($DS = |model\_data| - RIF_i$) as calculated in Sect. 3.1.1.

Assuming that the matrix has NT rows and $|UCL_i|$ columns, the complexity of the matrix model can be written as $(|UCL_i| - 1) \times logDS$.

Given this, we can apply the MDLP (Minimal Description Length Principle) which is the sum of the model complexity and the overall entropy of the matrix

$$\sum_{k=1}^{NT} [H_i(k) \times T_i(k)] + [(|UCL_i| - 1) \times logDS]. \tag{11}$$

A single greedy algorithm can simply choose the two consecutive rows which can produce the smallest information loss and then group them into one row. This procedure can be performed until the overall MDLP measure cannot decrease.

The MDLP algorithm has $O(NT \times |UCL_i|)$ complexity. Algorithm 3 formalizes this approach.

Algorithm 3 : Data Reduction, MDLP Algorithm

1: INPUT: $LOC\text{-}IN\text{-}TIME_i$, $T_i[NT]$
2: OUTPUT: $Reduced\_Model\_Data_i(RMD_i)$
3: OUTPUT: $Time\_Index_i$
4: Allocate a matrix with one column and NT rows for $Time\_Index_i$
5: Initialize the time–index $Time\_Index_i[1] = 1$
6: **for** $k = 1$ *to* NT **do**
7: $\quad R_i[k] = \sum_{j=1}^{|UCL_i|} LOC\text{-}IN\text{-}TIME_i(t_k, l_j)$
8: $\quad H_i[k] = -\sum_{j=1}^{|UCL_i|} \left(\frac{LOC\text{-}IN\text{-}TIME_i(t_k, l_j)}{R_i[k]}\right) \times \log\left(\frac{LOC\text{-}IN\text{-}TIME_i(t_k, l_j)}{R_i[k]}\right)$
9: **end for**
10: Initialize the reduced matrix: $RMD_i[1] = LOC\text{-}IN\text{-}TIME_i[1]$
11: For any consecutive two rows k and $k + 1$
12: $H'_i[k] = -\sum_{j=1}^{|UCL_i|} \left(\frac{RMD_i((k+1), l_j) + LOC\text{-}IN\text{-}TIME_i(t_k, l_j)}{R_i[(k+1)] + R_i[k]}\right) \times \log\left(\frac{RMD_i((k+1), l_j) + LOC\text{-}IN\text{-}TIME_i(t_k, l_j)}{R_i[(k+1)] + R_i[k]}\right)$
13: **if** $(T_i[(k + 1)] \times H_i[(k + 1)] + T_i[k] \times H_i[k] - (T_i[(k + 1)] + T_i[k]) \times H'_i[k] + [(|UCL_i| - 1) \times logDS]) \geq 0$ **then**
14: $\quad$ Merge Rows
15: **end if**
16: Repeat 12 until no consecutive rows can be merged
17: Done

As an example, assume that Fig. 10 represents a user profile for user u_i with eight distinct locations $1_1, l_2, \ldots, 8_8$. The size of this user profile is $1{,}440 \times 8 = 11{,}520$ numeric values. In this example, we illustrate the MDLP process, and Fig. 13 details this process as follows.

- First, we initialize a $Time\_Index_i$ matrix.
- We calculate the $H_i(k)$ value for each row k in the database and save it in the H_{matrix}.
- We run the MDLP algorithm against this user profile starting with the first two rows t_1 and t_2. In this example, $H_i(1) = 0$, $H_i(2) = 1$ and $H_i(1,2) = 0.93$.
- We calculate the information loss value:

$$T_i(1) \times H_i(1) + T_i(2) \times H_i(2) - (T_i(1) + T_i(2)) \times H_i(1,2) + [(|UCL_i| - 1) \times logDS].$$

  In this case, the information loss value is > 0, which indicates that we can combine these two rows together, and subsequently the first row in the reduced matrix will be

$$LOC\text{-}IN\text{-}TIME_i(t_1, l_j) = LOC\text{-}IN\text{-}TIME_i(t_1, l_j) + LOC\text{-}IN\text{-}TIME_i(t_2, l_j).$$

- We update both H_{matrix} to have the $H_i(1) = H_i(1,2)$, the *time frequency* to have $T_i(1) = T_i(1) + T_i(2)$, and the $Time\_Index_i$ matrix to have the first cell has the value t_2 to indicate that the first row in the reduced user profile represents the minutes t_1 and t_2 of the day.
- We advance to the third row and examine the H_{matrix} values for the rows t_1 and t_3. In this case, $H_i(1) = 0.93$ and $H_i(3) = 0$ while $H_i(1,3) = 0.81$, and therefore,

$$T_i(1) \times H_i(1) + T_i(3) \times H_i(3) - (T_i(1) + T_i(3)) \times H_i(1,3) + [(|UCL_i| - 1) \times logDS].$$

  In this case, the value is less than zero, and we cannot combine the rows.
- We repeat this process and scan the user profile row by row until we achieve a matrix with $Q' \times 8$ values and a $Time\_Index_i$ matrix with Q' values.

The total reduction in the matrix size is equal to the total eliminated rows multiplied by the total number of distinct locations $(1,440 - Q') \times 8$.

5 Experimental results

This section has two main parts. (1) We provide a detailed evaluation for our attack detection algorithms. We test our ability to build user profiles based on spatio–temporal traces and to detect anomalous behavior based on these profiles. We examine and compare the test results for both methods explained in Sect. 2.1.2. (2) We evaluate the effects of each data reduction algorithm in terms of reducing the user profile size and illustrate that this reduction has small impact on the detection accuracy.

5.1 Anomaly detection results

As discussed in Sect. 2.1.2, we used the Reality Mining [11] and Geolife mobility [38] traces for this evaluation.

Each user log was divided into two equal contiguous data sets: training data set (*model_data*) and testing data set (*test_data*) as described in Sect. 3.1. For each user, we randomly selected 100 samples from the *test_data* log with T duration. We repeated each test for four different T values (5, 15, 30, and 60 min). The T value is the detection delay.

5.1.1 Results for Model #1

For each user, we constructed models and calculated trust values P_{trust} following the steps described in Sect. 3.1. Attacker behavior traces are not presently available. However, traces for different users are available.

In our previous study [35], we explained the detailed results based on the Reality Mining data set where we demonstrated that the Model #1 of our system is capable of detecting an attack with a 94.4 % accuracy rate within 15 min as shown in Fig. 14. This figure indicates that in case of theft, our system has 94.4 % chance of notifying the device owner of the theft within 15 min and 92.0 % chance of notifying the device owner of the theft within 5 min.

In this study, we evaluate the system's ability to perform attack detection based on the Geolife data set. In Sect. 3, we have indicated that we have limited our user locations to an area of (138 × 110) miles, and we mapped each (7 × 8) meter to an area ID based on a (±0.0001°) change in each coordinate. Based on this information, and after eliminating all the records that do not fit within the Beijing area, we had a total 65 users with 100–1,200 distinct locations. The average was 780 distinct locations. We mapped these locations into area IDs (1–1,200).

Looking closer at the available location data, we notice a big difference among users and their trajectories. Figure 15 shows four different randomly selected users with their respective location histograms. This figure indicates that individuals tend not to travel very far, and when they do, they do not stay long. Although few users share some area IDs, the percentage of them visiting these areas is different, and the time of the day for these visits is different too. We reported similar observations for the Reality Mining data set [35].

Our test results with regard to accuracy exhibit also similar patterns. We were able to achieve a 95.6 % accuracy detection rate when $T = 5$ min and 93.8 % accuracy rate when $T = 15$ min as shown in Fig. 16. We notice that the detection accuracy for the 5-min delay is better in the Geolife than the 15-min detection delay. We can correlate this to the fact that this data set did not have many long

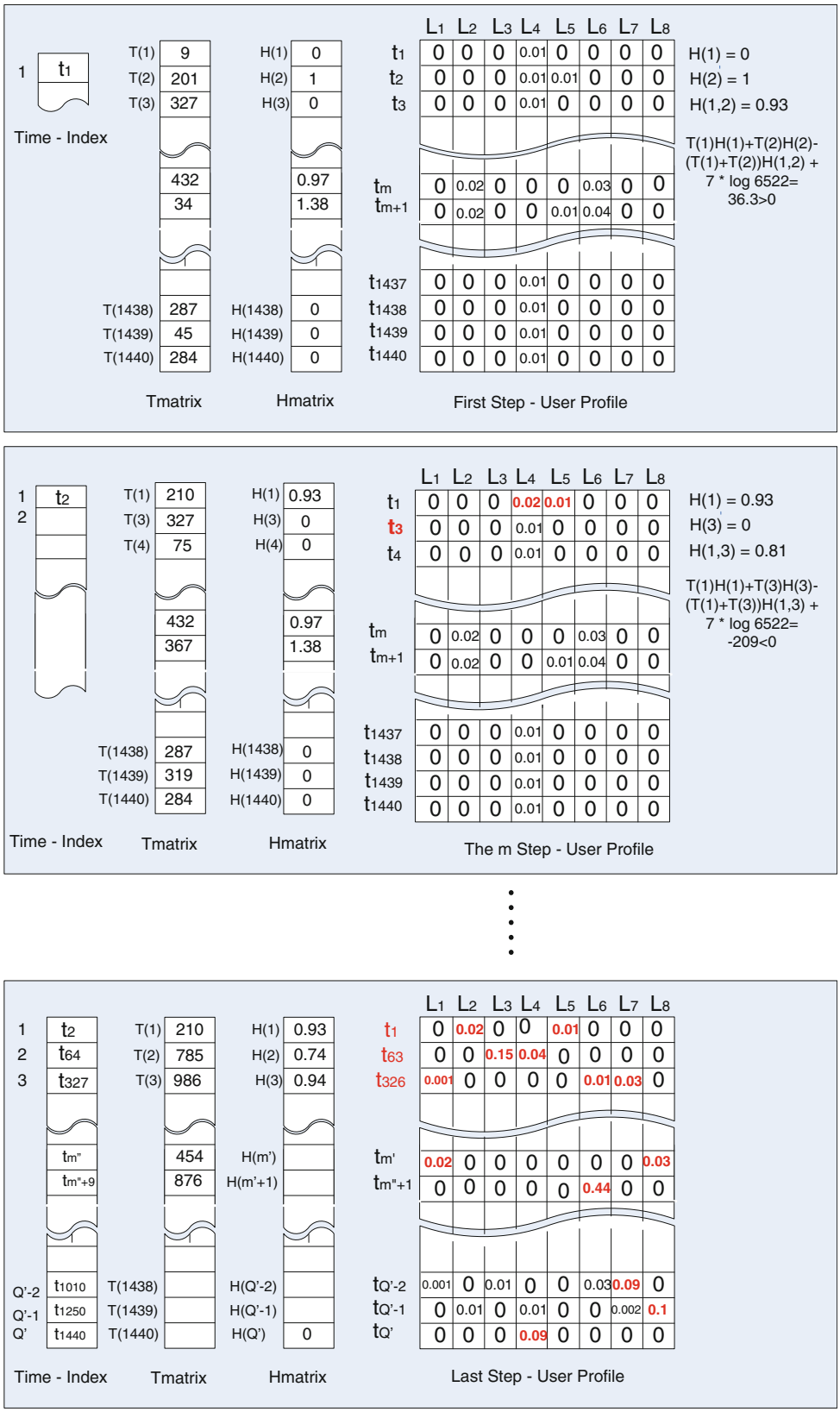

Fig. 13 MDLP reduction process

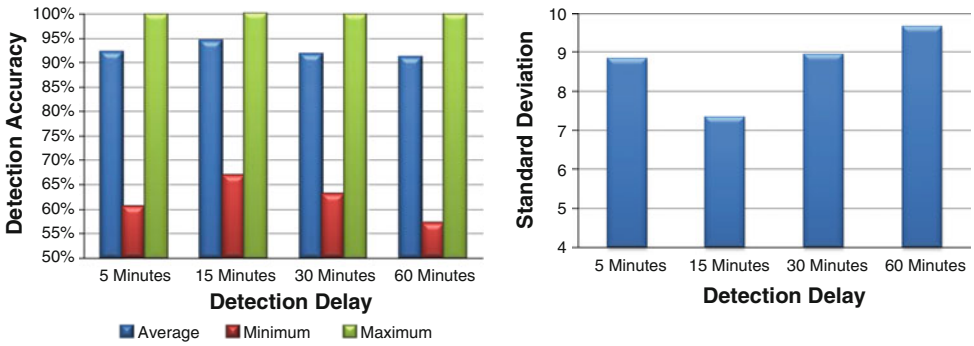

Fig. 14 Accuracy in anomaly detecting and standard deviation results in relation to trajectory size based on Model #1 and Reality Mining data set

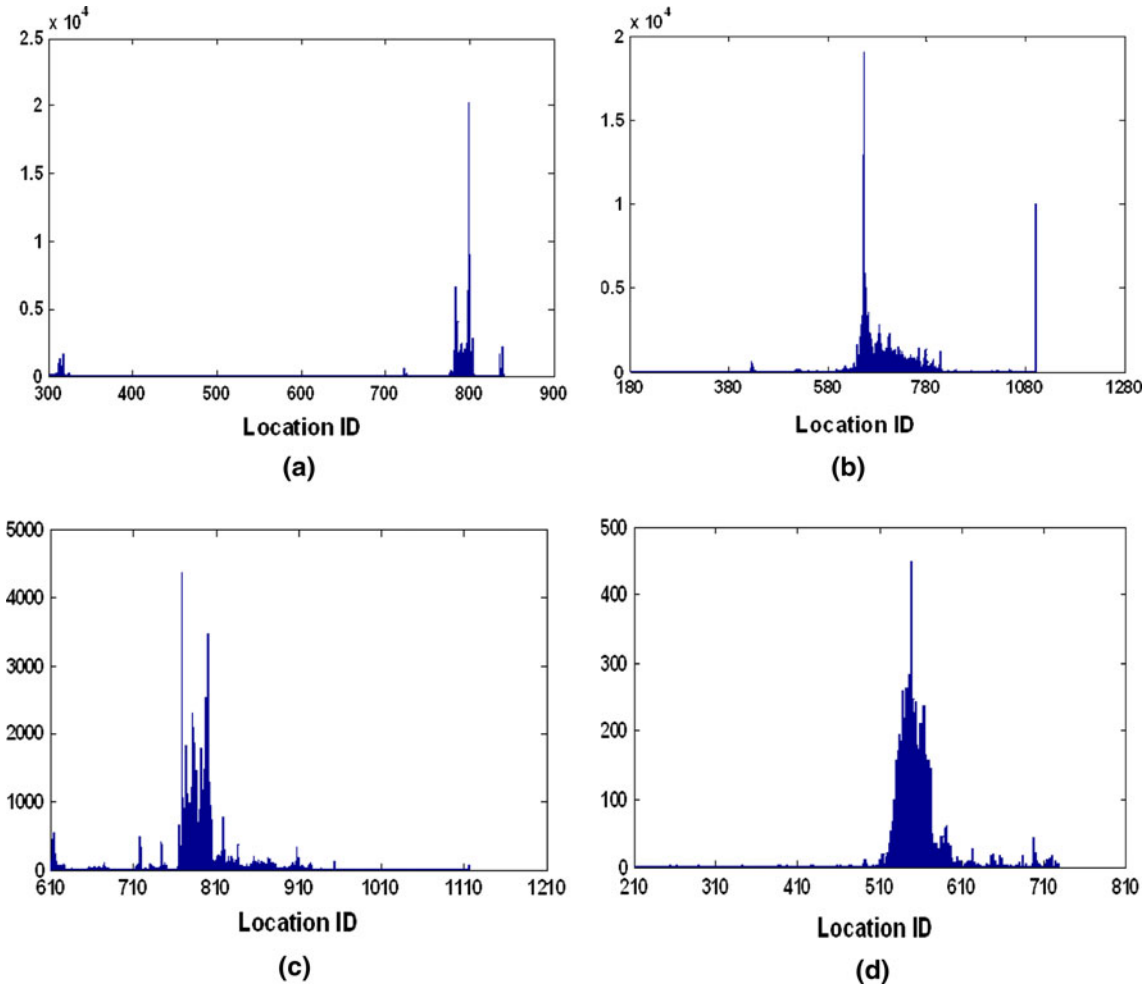

Fig. 15 Distinct location histogram for the Geolife data set. **a** user 20, **b** user 62, **c** user 123, **d** user 170

time sequences to use as a test sequence, which resulted in a decline in the detection accuracy associated with the longer delay. On the other hand, the granularity of the location information is finer for the Geolife than the Reality Mining data set, which also contributes to a higher detection rate for smaller values of T.

5.1.2 *Results of Model #2*

We followed the same steps described in [35] to calculate the FAR′ values based on Model # 2 user profile and probability analysis. We demonstrated in that study based on the Reality Mining data set that Model #2 is capable of

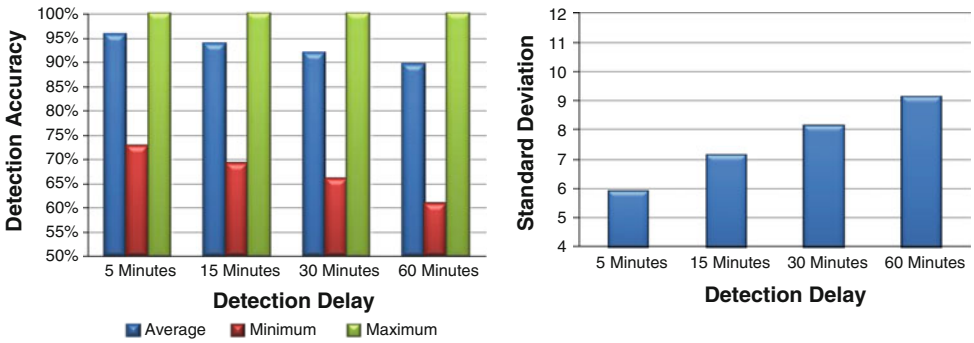

Fig. 16 Accuracy in anomaly detecting and standard deviation results in relation to trajectory size based on Model #1 and Geolife data set

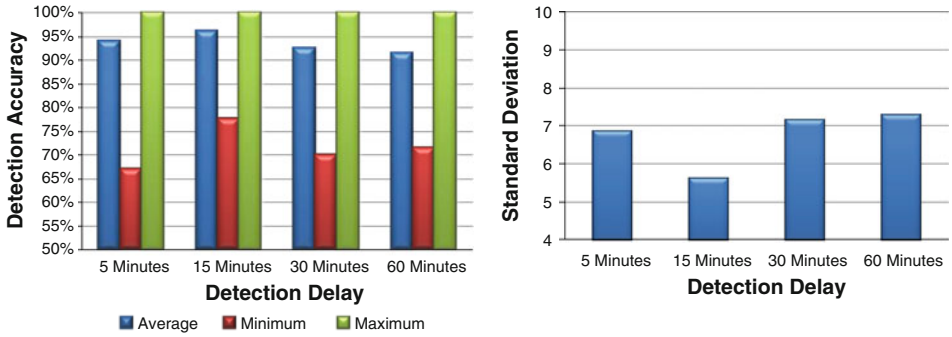

Fig. 17 Accuracy in anomaly detecting and standard deviation results in relation to trajectory size based on Model #2 and Reality Mining data set

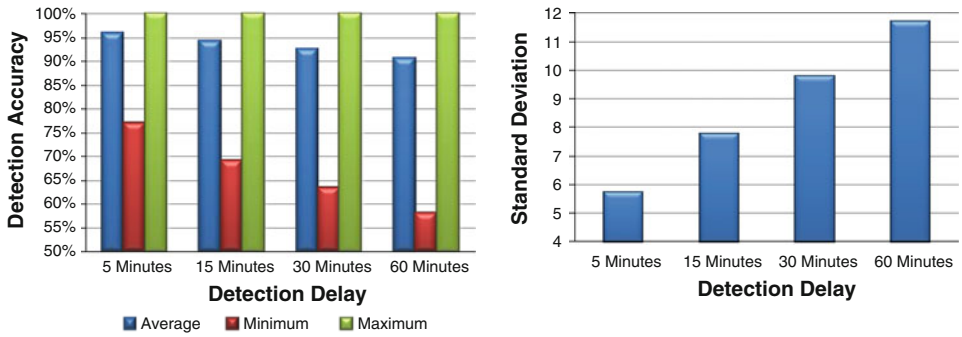

Fig. 18 Accuracy in anomaly detecting and standard deviation results in relation to trajectory size based on Model #2 and Geolife data set

detecting an attack with a 96.13 % accuracy rate within 15 min as shown in Fig. 17.

The Geolife data set allowed high detection accuracy too as shown in Fig. 18. The detection accuracy range was 96.0–90.5 %. Lower P'_{trust} values are associated with the longer traces, which indicates that it is uncommon for most users to make large day-to-day changes in motion patterns affecting short intervals within a trace. However, longer intervals are more likely to change from day to day.

5.1.3 Model comparison

As illustrated in Figs. 14, 16, 17, and 18, the average accuracy is slightly better for Model #2 than for Model #1 for small sample intervals (less than 30 min)—but the standard deviation is significantly better in the Reality Mining data set, while the improvement is slightly there for the "Geolife" data set.

However, the cost of obtaining this small improvement in accuracy ($\leq$2 %) for $T = 15$ min is expensive, considering the required memory to store the user profile: ($NT \times |UCL_i|$) for Model #1, and ($NT \times |UCL_i| \times |UCL_i|$) for Model #2, along with the respective time complexity for anomaly detection $O(TS \times NT \times |UCL_i|)$ for Model #1 and $O(TS \times NT \times |UCL_i| \times |UCL_i|)$ for Model #2. Therefore, our recommendation for this data set is to use Model #1 approach to achieve a high detection accuracy with lower memory requirements.

The simplicity of the resulting user models resulted in an efficient anomaly detection process supporting an average detection time 0.02 seconds, as shown in Fig. 19. A comparison between our results and those of existing systems is given in Table 1.

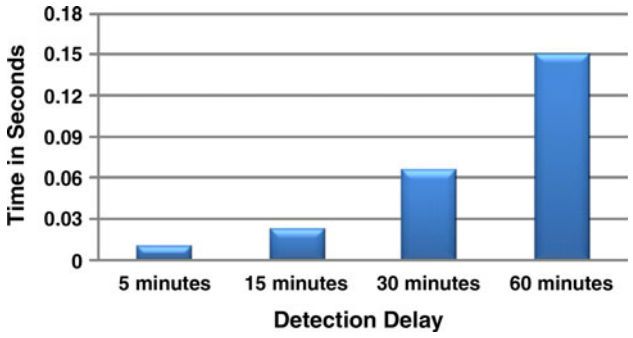

Fig. 19 Anomaly detection elapsed time according to sample interval

Table 1 Comparison with existing theft detection systems

	Our system	Gadget-Trak [12]	RecoveryCop [25]	LaptopCop [23]
Detection latency	15 min	N/A	N/A	N/A
Accuracy	96.13 %	N/A	N/A	N/A
Data protection	Yes	No	No	Yes
User intervention	No	Yes	Yes	Yes

5.2 Reduction results

In this section, we evaluate the efficiency of the Matrix Reduction methods based on running both the Row-Merge algorithm and the MDLP algorithm. Our evaluation is based on the user profiles built in the previous sections. Efficiency evaluation includes

1. reduction rate,
2. detection accuracy in relation to the reduced profile,
3. algorithm complexity, and
4. elapsed time required to perform the reduction process.

5.2.1 *Row-Merge algorithm*

Processing the Row-Merge matrix reduction algorithm for all the users' profiles in both data sets when $ThV = 1$ has shown an inconsistent reduction rate among the users. Figure 20 shows these results based on the Reality Mining data set. The reduction rate is high for the profiles with few distinct locations (<5), while it is low for the user profiles with more than five distinct locations. Yet, after performing the detection attack as described in Sect. 3.1, we noticed that the detection accuracy has not been impacted negatively, which is not surprising based on the analysis provided in the example in Sect. 4.1. Figure 21 illustrates the detection accuracy for all users based on the reduced user profile data when $ThV = 1$.

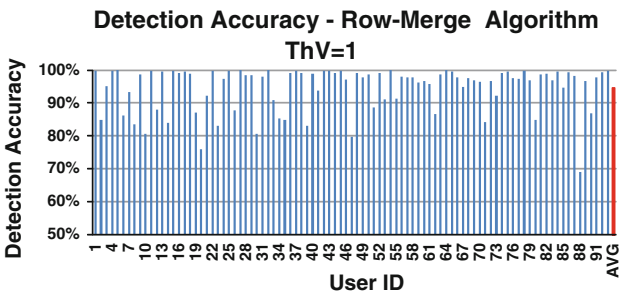

Fig. 21 The detection accuracy for the 93 users after applying the Row-Merge reduction algorithm

To further improve the reduction rate, we decided to explore the possibility of consolidating the rows that represent t_k where $ThV > 1$. For each user u_i, we examined several values for $ThV = 1, 2, 3, 4, 5, \ldots, / \frac{|UCL_i|}{3}$. The changes of the ThV value have improved the reduction rate and did not have a nominal negative impact on the detection accuracy. $ThV = \frac{|UCL_i|}{3}$ produced the best results, where we have seen consistent reduction rate and high detection accuracy.

5.2.2 *MDLP algorithm*

Processing the MDLP matrix reduction algorithm for the same users' profiles has shown a consistent reduction rate among all users as shown in Fig. 22 based on the Reality Mining data set. The total number of distinct locations had no effect on the reduction rate. In addition,

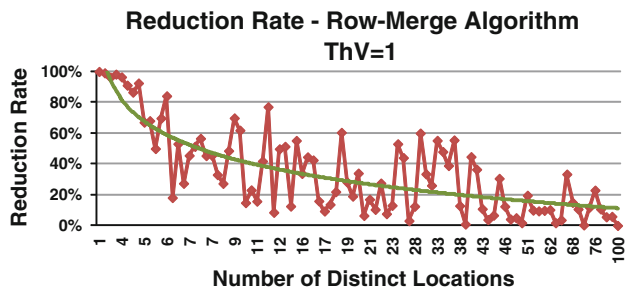

Fig. 20 The reduction percentage based on the number of distinct location in the profile

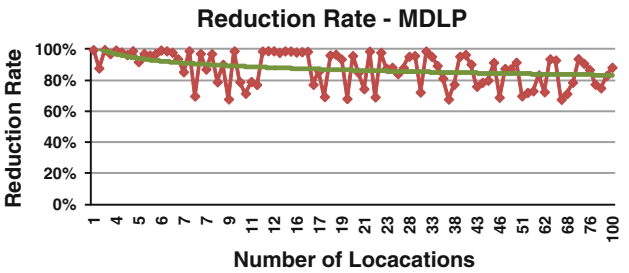

Fig. 22 The reduction percentage based on the number of distinct location in the profile based on the MDLP algorithm

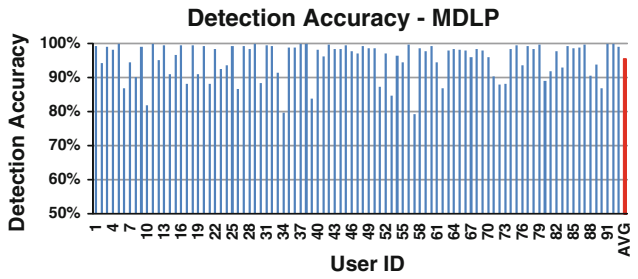

Fig. 23 The detection accuracy based on the MDLP reduction algorithm

the detection accuracy rate has not been impacted negatively (less than 1 % decrease in detection accuracy) as shown in Fig. 23.

We notice that the reduction rate for the MDLP algorithm is in the range (67.6–99.5 %) with an average of 87.6 %. This reduction came at a cost of only 1 % reduction in detection accuracy.

As we have indicated previously, these results are expected since users spend most of their times in a few locations, and they repeatedly visit these locations.

5.2.3 *Row-Merge algorithm versus MDLP algorithm*

In the previous subsections, we have illustrated that these two algorithms provide comparable data reduction percentages, in addition to a comparable detection accuracy rates as shown in Fig. 24. This figure shows the reduction rate and the detection accuracy based on the reduction algorithms for both the Reality Mining data set and the Geolife data set. We notice that the Row-Merge algorithm reduction rate is related to the *ThV* value, for example, when $ThV = 1$ the reduction rate average was

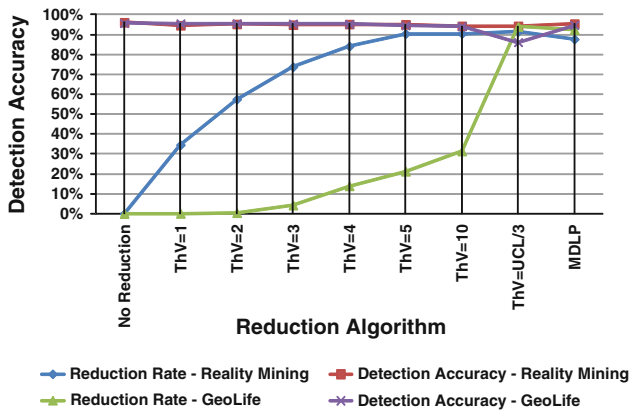

Fig. 24 A comparison of the reduction rate and detection accuracy results based on different *ThV* values for the Row-Merge algorithm and the MDLP algorithm

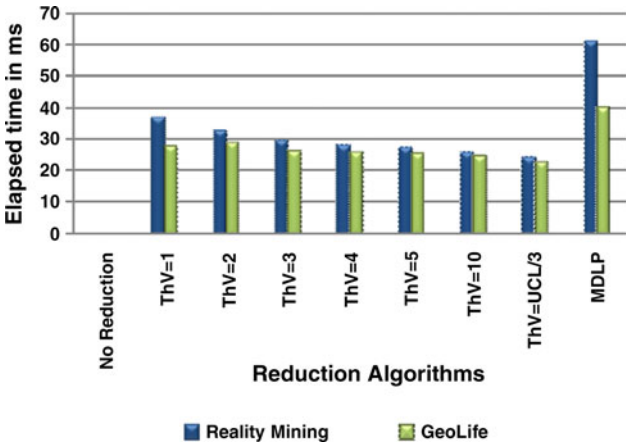

Fig. 25 The elapsed time to perform the matrix reduction process

as low as 34.5 % for Reality Mining data set, while it was 0.069 % for the Geolife data set. This rate has improved with the increase of the *ThV* value where the reduction rate reached the 91 % for the Reality Mining data set, and 93 % for the Geolife data set when $\left(ThV = \frac{|UCL_i|}{3}\right)$. The main reason for this difference in the reduction rate between the two data sets is the number of distinct locations associated with each user; thus, we achieved a comparable results when we used $\left(ThV = \frac{|UCL_i|}{3}\right)$. Similar results were achieved by the MDLP, which performed very well on both data sets.

Regardless of the reduction rate, the detection accuracy rate was not affected significantly after data reduction for both algorithms and data sets. Therefore, it is hard to compare these two algorithms based on the reduction rate and the detection accuracy only.

Upon examining the algorithm time complexity, we noticed that both algorithms have $O(NT \times |UCL_i|)$ complexity. However, the MDLP algorithm has higher overhead because it accesses the user model several times for every reduction process; the first time to calculate the *H* function values, the second time to perform a first round of reduction, the third time to perform a second round, and if needed, forth and fifth time. On the other hand, the Row-Merge algorithm goes over the user model only one time and performs the reduction. In addition, the Row-Merge algorithm performs a simple comparison at every row with for the total number of nonzero values in the row with a pre-defined value *ThV*, and based on the results, the algorithm either merges the rows for reduction or skip to the next row. This simple logic makes the Row-Merge algorithm more efficient than the MDLP algorithm that required double the time to obtain the same reduction rate, and similar accuracy as shown in Fig. 25.

6 Related work

Spatio–temporal data management and efficient query processing techniques have been the topics of intensive research in the field of Moving Objects Databases [16]. In particular, trajectory analysis and similarity detection have yielded numerous research results in the recent years [9, 13, 22]. Several results from this arena have goals similar to ours. For example, Mouza and Rigaux [10] propose regular expression-based algorithms for detecting mobility patterns. However, those patterns do not explicitly model the temporal dimension of the motion, that is, the focus is more on routes than trajectories.

In order to improve application awareness during trajectory data analysis, Alvares et al. [2] proposed adding semantic information during trajectory preprocessing. Hung et al. [20] proposed the complementary approach of using a probabilistic suffix tree to measure separation among users trajectories. Xie et al. [32] addressed the problem of predicting social activities based on users' trajectories. In addition, Trestian et al. [30] used association rule mining to investigate the relationships between geographic locations and user habits for mobile devices.

Detecting malware in mobile devices usage is a topic that has been tackled via various formalism. Using temporal logic of causal knowledge as language, malicious behavior signatures were proposed by Bose et al. [5] for mobile devices running Symbian OS. A complementary approach based on diffusion over bipartite graphs was presented by Alpcan et al. [1], and another approach that studies Bayesian networks, RBF, KNN, and random forest is presented by Damopoulos et al. [8]. Fraud detection based on usage behavior has also been addressed [6], where the underlying classifier is based on artificial neural networks. While in our earlier work [34], we attempted to use file-access patterns to detect malicious use; in this work, our focus was on detecting deviations from individual spatio–temporal patterns.

A cloud-based framework to detect intrusions and to provide fast response for the mobile device is introduced by Houmansadr et al. [19]. Their goal is complementary to our approach of enabling the mobile devices themselves to detect a potential theft by comparing user's trajectories.

Sun et al. [29] proposed mobile intrusion detection based on the Lempel–Ziv compression algorithm and Markov Chains. The proposed technique used three-level Markov Chains and did not consider the association between time of the day and the location. Their ability to detect attack using the proposed technique is limited to the times at which the user is making phone calls and moving faster than 60 miles per hour. Yan et al. [33] improved on this work, yet the delay in detecting attack was 24 h, since the traces were obtained once a day, with a sampling period of 30 min. Our technique has an attack detection latency of 15 min. Hall et al. [17] proposed an intrusion detection method based on mobility traces. Their focus was on public transportation traces in which the paths are pre-defined.

7 Concluding remarks

We presented an approach for detecting anomalous use of mobile devices. Our system uses spatio–temporal mobility data to build models that have high anomaly detection accuracy. Combining the spatio–temporal model (for users with few locations) and trajectory-based model (for users with many locations) allowed an average attack detection rate of 94.48 % for Model #1 and 96.13 % for Model #2, with a detection latency of 15 min. To further improve the efficiency of this system, we applied a couple of data reduction algorithms (Row-Merge, MDLP), which enabled us to obtain high reduction rate while still capable of detecting attacks with a 94 % accuracy.

One possible extension is to enrich the model by allowing nonzero probabilities to capture the cases of the owner visiting new locations. In addition, we would like to investigate the quality of accuracy, along with other trade-offs that may be involved when our approach is dealing with prolonged disconnections from a server. We also plan to expand this study to incorporate/couple different context dimensions (e.g., call-patterns and application logs) in order to improve the detection accuracy.

Acknowledgments This work was supported in part by the National Science Foundation under awards TC-0964545, CNS-0720691, CNS-0347941, and CNS-0910952.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Alpcan T, Bauckhage C, Schmidt AD (2010) A probabilistic diffusion scheme for anomaly detection on smartphones. In: Information security theory and practices: security and privacy of pervasive systems and smart devices. Springer, Berlin, pp 31–46
2. Alvares LO, Bogorny V, Kuijpers B, Fernandes JA, Moelans B, Vaisman A (2007) A model for enriching trajectories with semantic geographical information. In: Proceedings of the 15th ACM international symposium on advances in geographic information systems, vol 22. ACM, pp 1–8
3. Barron A, Rissanen J, Yu B (1998) The minimum description length principle in coding and modeling. IEEE Trans Inf Theory 44(6):2743–2760
4. Bayir MA, Demirbas M, Eagle N (2009) Discovering spatio-temporal mobility profiles of cellphone users. In: IEEE international symposium on world of wireless, mobile and multimedia networks & workshops. IEEE Computer Society, pp 1–9

5. Bose A, Hu X, Shin KG, Park T (2008) Behavioral detection of malware on mobile handsets. In: Proceedings of the 6th international conference on mobile systems, applications, and services. ACM, pp 225–238
6. Boukerche A, Annoni Notare MSM (2002) Behavior-based intrusion detection in mobile phone systems. J Parallel Distrib Comput 62(9):1476–1490
7. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv (CSUR) 41(3):1–58
8. Damopoulos D, Menesidou SA, Kambourakis G, Papadaki M, Clarke N, Gritzalis S (2011) Evaluation of anomaly-based ids for mobile devices using machine learning classifiers. Secur Commun Netw 5(1):3–14
9. Dodge S, Weibel R, Forootan E (2009) Revealing the physics of movement: comparing the similarity of movement characteristics of different types of moving objects. Comput Environ Urban Syst 33(6):419–434
10. du Mouza C, Rigaux P (2005) Mobility patterns. GeoInformatica 9(4):297–319
11. Eagle N, Pentland AS, Lazer D (2009) Inferring social network structure using mobile phone data. Proc Nat Acad Sci (PNAS) 106(36):15274–15278
12. GadgetTrak. http://www.gadgettrak.com/
13. Gómez L, Kuijpers B, Vaisman A (2008) Querying and mining trajectory databases using places of interest. In: New trends in data warehousing and data analysis, vol 3. pp 1–26
14. González MC, Hidalgo CA, Barabási AL (2008) Understanding individual human mobility patterns. Nature 453(7196):779–782
15. Grünwald PD, Myung IJ, Pitt MA (2005) Advances in minimum description length: theory and applications. MIT press, Cambridge, MA
16. Güting RH, Schneider M (2005) Moving objects databases. Morgan Kaufmann, San Francisco, CA
17. Hall J, Barbeau M, Kranakis E (2005) Anomaly-based intrusion detection using mobility profiles of public transportation users. In: IEEE international conference on wireless and mobile computing, networking and communications, vol 2. IEEE Computer Society, pp 17–24
18. Hansen MH, Yu B (2001) Model selection and the principle of minimum description length. J Am Stat Assoc 96(454):746–774
19. Houmansadr A, Zonouz SA, Berthier R (2011) A cloud-based intrusion detection and response system for mobile phones. In: IEEE/IFIP 41st international conference on dependable systems and networks workshops. IEEE Computer Society, pp 31–32
20. Hung CC, Chang CW, Peng WC (2009) Mining trajectory profiles for discovering user communities. In: Proceedings of the 2009 international workshop on location based social networks. ACM, pp 1–8
21. Isobe Y, Seto Y, Kataoka M (2001) Development of personal authentication system using fingerprint with digital signature technologies. In: Proceedings of the 34th annual Hawaii international conference on system sciences. IEEE Computer Society
22. Jeung H, Liu Q, Shen HT, Zhou X (2008) A hybrid prediction model for moving objects. In: IEEE 24th international conference on data engineering. IEEE Computer Society. IEEE, pp 70–79
23. Laptop Cop Software. http://www.laptopcopsoftware.com/index.html
24. Map Tools. http://maptools.com/UsingLatLon/LatLon.html
25. Mobile Security Monitoring. Windows mobile security monitoring software. http://www.recoverycop.com/index.html
26. Moon YS, Leung CC, Pun KH (2003) Fixed-point (gmm)-based speaker verification over mobile embedded system. In: Proceedings of the 2003 ACM SIGMM workshop on biometrics methods and applications. ACM, pp 53–57
27. Open Street Map. http://www.OpenStreetMap.org
28. Rhee I, Shin M, Hong S, Lee K, Kim SJ, Chong S (2011) On the Levy-walk nature of human mobility. IEEE/ACM Trans Netw (TON) 19(3):630–643
29. Sun B, Yu F, Wu K, Xiao Y, Leung VCM (2006) Enhancing security using mobility-based anomaly detection in cellular mobile networks. IEEE Trans Veh Technol 55(4):1385–1396
30. Trestian I, Ranjan S, Kuzmanovic A, Nucci A (2009) Measuring serendipity: connecting people, locations and interests in a mobile 3G. In: Proceedings of the 9th ACM SIGCOMM conference on internet measurement conference. ACM, pp 267–279
31. World fact book. http://www.cia.gov/library/publications/the-world-factbook/index.html
32. Xie K, Deng K, Zhou X (2009) From trajectories to activities: a spatio–temporal join approach. In: Proceedings of the 2009 international workshop on location based social networks. ACM, pp 25–32
33. Yan G, Eidenbenz S, Sun B (2009) Mobi–watchdog: you can steal, but you can't run!. In: Proceedings of the second ACM conference on wireless network security. ACM, pp 139–150
34. Yazji S, Chen X, Dick RP, Scheuermann P (2009) Implicit user re-authentication for mobile devices. In: Proceedings of the 6th international conference on ubiquitous intelligence and computing. Springer-Verlag, Berlin, pp 325–339
35. Yazji S, Dick RP, Scheuermann P, Trajcevski G (2011) Protecting private data on mobile systems based on spatio–temporal analysis. In: Proceedings of the 1st international conference on pervasive and embedded computing and communication systems. SciTePress, pp 114–123
36. Zheng Y, Li Q, Chen Y, Xie X, Ma WY (2008) Understanding mobility based on GPS data. In: Proceedings of the 10th international conference on ubiquitous computing. ACM, pp 312–321
37. Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of the 18th international conference on world wide web. ACM, pp 791–800
38. Zheng Y, Xie X, Ma W-Y (2010) Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–40