Mobi-Watchdog: You Can Steal, But You Can't Run!

Guanhua Yan
Information Sciences (CCS-3)*
Los Alamos National Laboratory
Los Alamos, NM 87545, USA
ghyan@lanl.gov

Stephan Eidenbenz
Information Sciences (CCS-3)
Los Alamos National Laboratory
Los Alamos, NM 87545, USA
eidenben@lanl.gov

Bo Sun
Department of Computer Science
Lamar University
Beaumont, TX 77710, USA
bsun@cs.lamar.edu

ABSTRACT

Recent years have witnessed widespread use of mobile devices such as cell phones, laptops, and PDAs. In this paper, we propose an architecture called *Mobi-Watchdog* to detect mobility anomalies of mobile devices in wireless networks that track their locations regularly. Given the past mobility records of a mobile device, Mobi-Watchdog uses clustering techniques to identify the high-level structure of its mobility and then trains a HHMM (hierarchical hidden Markov model). Mobi-Watchdog raises an alert by requesting the device holder to reauthenticate himself when it finds an observed mobility trace significantly deviates from the trained model. The time complexity of the original generalized Baum-Welch algorithm [4], which is used for HHMM parameter reestimation, scales linearly with T^3, where T is the number of locations in an observed sequence. Such a high computational cost can significantly impede deployment of Mobi-Watchdog in large-scale wireless networks in practice. To achieve better scalability, we modify this algorithm to make it scale linearly with T instead. Experimental results with realistic mobility traces demonstrate that Mobi-Watchdog detects mobility anomalies with high probability and reasonably low false alarm rates. We also show that Mobi-Watchdog has very low computational overhead, which makes it a viable candidate for mobility anomaly detection in large wireless networks.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network management; C.4 [**Computer Systems Organization**]: Reliability, availability, and serviceability; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Security, Performance

*Los Alamos National Laboratory Publication No. LA-UR-07-7753

Keywords

Wireless networks, Mobility, Anomaly detection, HHMM

1. INTRODUCTION

Recent years have witnessed widespread use of mobile devices such as cell phones, laptops, PDAs, and BlackBerry handhelds. Albeit being convenient, mobile devices are prone to loss or theft. Protecting the services associated with them and the information stored on these devices emerges as an acute security concern. It is reported that 37% of smart phone users keep confidential business data on their phones [9]. Hence, it becomes an important problem to enhance the security of the increasingly popular mobile devices.

In this paper, we address how to detect mobility anomalies of mobile devices in wireless networks. As many wireless mobile networks (e.g., cellular networks) track the location of each mobile device in them, mobility anomaly detection offers an opportunity for an early warning system, which is similar in spirit to credit card fraud detection systems. A typical application of mobility anomaly detection is that once the network suspects that the cell phone is being used by another person, it requires that the cell phone should reauthenticate itself to the network by sending the password; failure to do so automatically suspends the cell phone service or even disables the access to the storage of some critical data on the phone. This can be implemented as an *optional* service so that people concerned with their mobility privacy simply do not subscribe it. Such a service provides people who lost their mobile devices another level of service protection before they eventually report their losses to their service providers. Recent high-profile stolen cell-phone billing cases suggest that the experience after a mobile device is lost or stolen can be tremendously troublesome [18]. Moreover, mobility anomaly detection can also be performed by the mobile device itself, as long as this function cannot be easily disabled. For example, the mobility anomaly detection module can run in the Operating System kernel or be implemented by tamper-resistant hardware on a high-end mobile device.

As mobile devices are often carried by human beings, the validity of a mobility anomaly detection system rests on *how predictable human mobility is.* Obviously, if people move extremely randomly in their daily life, any mobility anomaly system is doomed to be useless. A recent study which tracks the locations of 100,000 cell phone users over six months however reveals that people's mobility patterns follow a high degree of temporal and spatial regularities; half of the people in the study rarely stray away from a six-mile circle and 83 percent of them almost always stay within a 37-mile wide circle [5]. This result suggests that a mobile device's history mobility records can be used to detect mobility anomalies when it is stolen or lost.

Motivated by this, we propose an architecture called *Mobi-Watchdog* that uses a model-based solution to detect mobility anomaly of mobile devices: we train a model from the history mobility traces of a mobile device and then use it to determine whether a newly observed mobility trace is anomalous. An effective model-based mobility anomaly detection scheme calls for models that accurately characterize human trajectories, which have been observed to exhibit hierarchical structures [7]. As hierarchical hidden Markov models (HHMMs) have been shown able to characterize well many real-world processes with hierarchical structures [4], we use them in Mobi-Watchdog to detect mobility anomalies. More specifically, we address how to construct an HHMM from observations on a mobile device mobility and how to initialize the model parameters. The time complexity of the original generalized Baum-Welch algorithm [4], which is used for HHMM parameter reestimation, scales linearly with T^3, where T is the number of locations in an observed sequence. Such high computational cost obviously impedes deployment of Mobi-Watchdog in large-scale wireless networks for mobility anomaly detection. To achieve better scalability, we modify this algorithm to make it scale linearly with T instead. Experimental results with realistic mobility traces demonstrate that Mobi-Watchdog can effectively detect mobility anomalies with reasonably low false alarm rates. We also show that Mobi-Watchdog has very low computational overhead, which makes it a viable candidate for mobility anomaly detection in large wireless networks.

Related work. Mobility modeling of mobile devices in wireless networks has been actively studied recently [6][11]. The common goal of this line of research is to develop mobility models that are able to generate synthetic traces for simulation purposes or predict future locations from past mobility traces. The solution proposed in this paper addresses a different problem, although our work also needs to model the mobility of mobile devices. A few approaches have also been proposed to detect mobility anomalies in wireless cellular networks. Sun et al. presented a LZ-based algorithm in [17] and our experimental results show that their approach performs poorly in detecting mimicry attacks. Instance-based learning techniques have also been developed in [16] to detect mobility anomalies. Compared with model-based detection schemes, instance-based techniques require reading past mobility traces when performing anomaly detection, which imposes extra computational overhead.

The remainder of this paper is organized as follows. In Section 2, we present an overview of Mobi-Watchdog and explain the rationale of using HHMMs to detect mobility anomaly in Mobi-Watchdog. Section 3 provides a brief introduction to HHMMs and also presents the challenges of applying HHMMs for mobility anomaly detection. Section 4 discusses how Mobi-Watchdog constructs an HHMM from normal mobility traces and calculates its parameters. Section 5 presents how Mobi-Watchdog decides whether a test mobility trace is anomalous based on a trained HHMM. We evaluate the effectiveness and efficiency of Mobi-Watchdog in Section 6 and then conclude this paper in Section 7.

2. OVERVIEW OF MOBI-WATCHDOG

Mobi-Watchdog can be used in any wireless mobile networks that track locations of mobile devices regularly. An obvious example is the widespread cellular networks. Also, a Wi-Fi network can monitor the mobility history of a mobile device through the sequence of access points (APs) that it has used. Without loss of generality, we use set $C = \{c_i\}|_{1 \le i \le |C|}$ to denote the entire set of locations in the mobile network being considered. Here, location symbol c_i is a cell identifier in a cellular network or an AP in a Wi-Fi network.

The normal mobility history of a mobile device can be abstracted as a set of location sequences $Q = \{H_j\}|_{1 \le j \le |Q|}$, where $H_j = \tilde{c}_1(j)\tilde{c}_2(j)...\tilde{c}_{|H_j|}(j)$ with $\tilde{c}_k(j) \in C$ and $1 \le k \le |H_j|$. $|Q|$ and $|H_j|$ is the number of location sequences in Q and the length of location sequence H_j, respectively. A location sequence can be, for example, a sequence of cells that a cell phone has traversed within a day. Set Q is maintained by Mobi-Watchdog for each subscribing mobile device, which, when given a test location sequence $\widehat{H}$, decides whether it is anomalous[1].

Mobi-Watchdog uses a model-based approach to detect mobility anomaly of mobile devices. More specifically, Mobi-Watchdog trains a model from a mobile device's normal mobility history and decides whether $\widehat{H}$ significantly deviates from it. Obviously, an ideal model should accurately characterize the mobility history collected under normal conditions. We choose those models able to characterize human mobility accurately, given the fact that mobile devices are often carried by human beings. Fortunately, human mobility has been shown to exhibit high regularity from both analysis of 100,000 cell phone users' mobility traces [5] and road traffic surveys [14].

From the data collected from a number of surveys, it is observed that *hierarchical structures* are typical of human behavior [7][8][15]. At a high level, human behaviors in a typical day can be classified into different activities, such as working, staying at home, shopping, exercising, and so on. For instance, for a typical person, two to four activities cover 75% and eight activities cover 85% of all movements observed over four weeks [3]. On the other hand, a human activity is associated with one or more places and paths between them. These places can be homes, work places, bus stops, parking lots, and friends' houses, restaurants, etc. Given these observations, it is natural to build a hierarchical model to capture both a mobile user's activities in a typical day at a high level and also her movements associated with each activity at a low level. Fortunately, some mathematical models have already been proposed to characterize the hierarchical structures in the real world. HHMMs, first introduced by Fine et al. in [4], are such examples. HHMMs have already been successfully applied in handwriting recognition [4] and Chinese lexical analysis [19].

It is worth mentioning that Mobi-Watchdog does not require any prior knowledge about a mobile user's activities and her paths between different locations to train the HHMM model. It, instead, applies unsupervised learning techniques to drive the model. From a practical point of view, this

[1]An adversary can turn off a mobile device to evade mobility anomaly detection. But the network can treat a long-time intractable device as behaving anomalously and requires it to reauthenticate itself when it is turned on.

is desirable because Mobi-Watchdog only needs to passively monitor the user's mobility records, which are already available in some wireless networks (e.g,. cellular networks). In the following section, we give a brief introduction to HHMMs.

3. PRIMER ON HHMMS

Hierarchical hidden Markov models are structured multi-level stochastic processes. An HHMM is characterized by a sextuple: $\lambda = (\mathcal{S}, \mathcal{V}, \mathcal{R}, \mathcal{A}, \Pi, \mathcal{B})$, where

- $\mathcal{S} = \{s_1, s_2, ..., s_n\}$: a set of individual states in the model. For each state s in $\mathcal{S}$, we also define its type $\delta(s)$ as any of *internal*, *end*, and *production*.
- $\mathcal{V} = \{v_1, v_2, ..., v_m\}$: a set of observation symbols.
- $\mathcal{R}$: state parentship matrix. The states in $\mathcal{S}$ form a tree structure in an HHMM. Let $p(s_i)$ denote the parent state of state s_i. For each r_{s_i,s_j} in $\mathcal{R}$, where $1 \leq i, j \leq n$, r_{s_i,s_j} is 1 if $s_i = p(s_j)$ or 0 otherwise. If a state is an internal state (i.e., $\delta(s) = internal$) then it must have child states; otherwise, if a state is an end or production state (i.e., $\delta(s) = end$ or $production$), then it must not have child states. Let $D(s_i)$ be the entire set of child nodes of internal node s_i. For an internal state s_i, let $e(s_i)$ be its child state whose type is *end*.
- $\mathcal{A}$: horizontal state transition probability matrix per state. We use q_t to denote the state at time t. Then, for each $a(s_i, s_j)$ in $\mathcal{A}$, where $1 \leq i, j \leq n$, $p(s_i) = p(s_j)$ and $\delta(s_i) \neq end$, we have: $a(s_i, s_j) = \mathbb{P}\{q_{t+1} = s_j \mid q_t = s_i\}$. If the current state q_t is an end state, the HHMM forces it to enter its parent state immediately. This is sometimes called *forced transition*.
- Π: initial vertical state transition vector per internal state. For each $\pi(s_k, s_i) \in \Pi$, where $1 \leq i, k \leq n$, $s_k = p(s_i)$ and $\delta(s_i) \neq end$, it gives the probability that the model immediately enters child state s_i after entering state s_k. Vertical state transitions do not go to end states.
- $\mathcal{B}$: observation symbol probability vector per production state. For each $b_{s_k}(v_j)$ in $\mathcal{B}$, we have the following: $b_{s_k}(v_j) = \mathbb{P}\{v_j \text{ is produced at time } t \mid q_t = s_k\}$, where $1 \leq k \leq n$, $1 \leq j \leq m$, and $\delta(s_k) = production$. Only at a production state does the HHMM produce an observation symbol.

Then naturally follow three questions: (1) given an HHMM and an observation sequence, how do we determine the probability that the model produces the observation sequence? (2) given an HHMM and an observation sequence, what is the most likely state transition path in the model that produces the observation sequence? (3) how do we adjust the model parameters $\mathcal{A}$, $\mathcal{B}$, and Π with new observations? To answer these questions, Fine et al. in [12] generalized the original algorithms that address these three questions for standard HMMs: the generalized forward-backward procedures compute the production probability of an observation sequence, the generalized Viterbi algorithm finds the most probable state transition path, and the generalized Baum-Welch method estimates the model parameters for the HMMs (Hidden Markov Models). The time complexity of these algorithms is $O(n \cdot T^3)$, where T is the length of the observation sequence and n is the number of states in the model. Murphy et al. later developed an algorithm based on DBN (dynamic Bayesian Network) representations [10] that has better empirical time complexity but is more difficult to implement.

In our problem domain, the two major challenges in applying HHMMs for mobility anomaly detection are two-fold: **First, without prior knowledge about a mobile user's activities and her regular paths, how should we construct an HHMM that accurately characterizes her mobility patterns? Second, given the high computational costs associated with HHMMs, how should we improve existing algorithms so that Mobi-Watchdog can be applied in large-scale wireless networks?** In the following section, we present an approach to constructing HHMMs that addresses these two challenges.

4. MOBILITY MODEL CONSTRUCTION

In this section we discuss how Mobi-Watchdog trains an HHMM based on a mobile device's history mobility record. Note that solutions to the three questions mentioned in Section 3 do not address how to learn the structure of the model (i.e., $\mathcal{S}$ and $\mathcal{R}$). In the following, we give an algorithm that does not require any a priori knowledge regarding the owner of the mobile device, such as her home or office location. Instead, our solution derives all necessary information directly from the given set of location sequences Q for the mobile device. The algorithm has three phases: high-level model construction (Phase I), low-level model construction (Phase II), and parameter reestimation (Phase III). As parameter initialization is crucial to estimate parameters for HMM-family models [12], we also discuss how to choose initial estimates of HHMM parameters.

4.1 Phase I: High-Level Model Construction

Mobi-Watchdog uses clustering techniques to identify high level structures from the history mobility traces.

Graph generation. Suppose that the HHMM we are trying to build is rooted at node $s^{(1)}$. We define the *level* of a node as its depth plus 1. Hence, the level of node $s^{(1)}$ is 1. We use the clustering techniques to determine the states at level 2 in the hierarchical model. At this level, a state represents a high-level activity in the mobile device's behavior. First, we create as follows an undirected graph $G(Y, E)$, where Y represents the set of nodes in set C and E represents the set of edges: We add an edge (c_a, c_b) between nodes c_a and c_b in graph G, if the mobile device traverses locations c_a and c_b consecutively, regardless of their order, at least once among all location sequences in Q. For simplicity, nodes of degree 0 are removed from graph G. The weight of the edge (c_a, c_b), denoted by $\widehat{w}(c_a, c_b)$, is defined by:

$$\begin{aligned}\widehat{w}(c_a, c_b) = \\ &\textstyle\sum_{1\leq j\leq|Q|}\sum_{1\leq i<|H_j|} 1\{\tilde{c}_i(j) = c_a \wedge \tilde{c}_{i+1}(j) = c_b\} + \\ &\textstyle\sum_{1\leq j\leq|Q|}\sum_{1\leq i<|H_j|} 1\{\tilde{c}_i(j) = c_b \wedge \tilde{c}_{i+1}(j) = c_a\}\end{aligned}$$

where $1\{\cdot\}$ is the indicator function. Denote the nodes in graph $G(Y, E)$ by y_1, y_2, ..., $y_{|Y|}$. We then derive a Markov matrix $W \in R^{|Y|\times|Y|} = \{w_{i,j}\}|_{1\leq i,j\leq|Y|}$ as:

$$w_{i,j} = \frac{\widehat{w}(y_i, y_j)}{\sum_{1\leq k\leq|Y|}\widehat{w}(y_i, y_k)}. \quad (1)$$

Clustering. Next, we cluster nodes in graph G into different groups. We use the MCL (Markov CLuster) algorithm [1] here because it does not require a priori knowledge of the cluster structure. The MCL algorithm deterministically computes the clusters by simulating random walks within a graph by iteratively applying two basic operations: *expansion* and *inflation*. This operation simulates e steps of a random walk under the current transition matrix and the inflation operation increases the probability of intra-cluster walks. It is shown that the MCL algorithm converges quadratically around the equilibrium states; in practice, it needs only three to ten iterations to reach a fixed point state [2].

Level-2 state definition. For each cluster u generated from the MCL algorithm, we use $l(u)$ to denote the set of locations that belong to cluster u. Naively, we create a level-2 state for each cluster. We, however, find that a large number of clusters can result from the MCL algorithm and many of them have only one location. To reduce the number of states created at level 2, we first define the *exposure rate* of a cluster u, denoted by $\zeta(u)$, as follows:

$$\zeta(u) = \frac{\sum_{1\le j\le |Q|}\sum_{1\le k\le |H_j|} 1\{\tilde{c}_k(j)\in l(u)\}}{\sum_{1\le j\le |Q|} |H_j|}, \qquad (2)$$

where we recall $\tilde{c}_k(j)$ is the k-th location in the j-th location sequence. In other words, $\zeta(u)$ gives the ratio of the number of times that locations in cluster u appear among all the location sequences in Q to the sum of the lengths of all the location sequences. We only create individual states for those clusters with exposure rates no less than a given threshold θ times $|Q|$; for all those clusters whose exposure rates are below $\theta \cdot |Q|$, we create a single aggregate state $s_a^{(2)}$ at level 2. We also create an end state $s_{end}^{(2)}$ at level 2. Hence, $e(s^{(1)}) = s_{end}^{(2)}$.

Level-2 parameter initialization. We use set $S^{(2)}$ to denote the set of states created at level 2. For each location y in set Y, we use $\varphi(y)$ to denote the level-2 state that it is assigned to. We then have:

$$\pi(s^{(1)}, s_i^{(2)}) = \frac{1}{|Q|} \times \sum_{1\le j\le |Q|} 1\{\varphi(\tilde{c}_1(j)) = s_i^{(2)}\}, \qquad (3)$$

where $s_i^{(2)} \in S^{(2)}$. In other words, the vertical transition probability from the root state to state s in $S^{(2)}$ is the normalized number of times that a location sequence in Q starts with a location assigned to state s.

For simplicity, we define $\varphi(\tilde{c}_{|H_j|}(j)) = null$, where $1 \le j \le m$ and $null$ is a virtual state. We define $z(s^{(2)})$, where $s^{(2)}$ is a non-end level-2 state, as follows:

$$z(s^{(2)}) = \sum_{1\le j\le |Q|}\sum_{1\le k\le |H_j|} 1\{\varphi(\tilde{c}_k(j))=s^{(2)} \wedge \varphi(\tilde{c}_{k+1}(j))\ne s^{(2)}\}.$$

Hence, $z(s^{(2)})$ gives the total number of times that the model departs from state $s^{(2)}$ among all the location sequences in Q. We then have the following:

$$a(s_i^{(2)}, s_{end}^{(2)}) = \frac{\sum_{1\le j\le |Q|} 1\{\varphi(\tilde{c}_{|H_j|}(j))=s_i^{(2)}\}}{z(s_i^{(2)})}, \qquad (4)$$

where $s_i^{(2)} \in S^{(2)}$ and $s_i^{(2)} \ne s_{end}^{(2)}$. Similarly, we have the horizontal state transition probabilities as follows:

$$a(s_i^{(2)}, s_u^{(2)}) = \frac{\sum_{1\le j\le |Q|}\sum_{1\le k<|H_j|} 1\{\varphi(\tilde{c}_k(j))=s_i \wedge \varphi(\tilde{c}_{k+1}(j))=s_u^{(2)}\}}{z(s_i^{(2)})}$$

where $s_i^{(2)}, s_u^{(2)} \in S^{(2)}$, $s_i^{(2)} \ne s_u^{(2)}$, and neither is $s_{end}^{(2)}$.

4.2 Phase II: Low-Level Model Construction

In this phase, we identify location subsequences that appear frequently in Q. Then, we use them as the bedrocks to build low-level states in the HHMM.

Algorithm 1 Calculate the numbers of appearances for non-repetitive and homogeneous location subsequences.

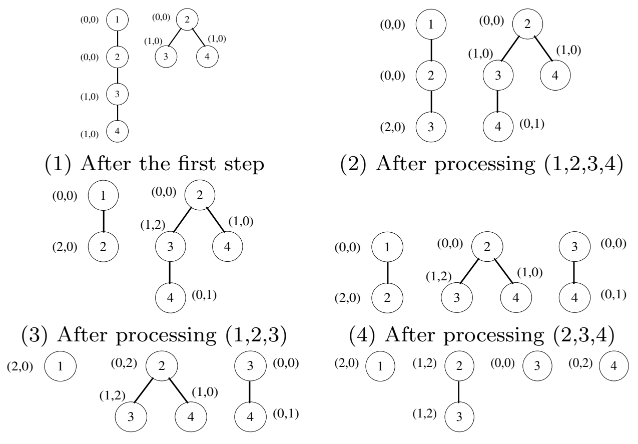
```
 1: $I \leftarrow \emptyset$
 2: for $j = 1$ to $|Q|$ do
 3:   $h \leftarrow$ empty subsequence
 4:   for $k = 1$ to $|H_j| + 1$ do
 5:     if $k > |H_j|$ or $\tilde{c}_k(j) \in h$ or $\varphi(\tilde{c}_{k-1}(j)) \ne \varphi(\tilde{c}_k(j))$
        then
 6:       If $h \in I$, then $\eta(h) \leftarrow \eta(h) + 1$; else add $h$ to $I$,
          $\eta(h) \leftarrow 1$
 7:       If $k \le |H_j|$, then $h \leftarrow \tilde{c}_k(j)$
 8:     else
 9:       $h \leftarrow h \circ \tilde{c}_k(j)$
10:     end if
11:   end for
12: end for
```

Feature extraction. We define a *non-repetitive* location subsequence as the one in which no location symbol appears more than once. We also define a *homogeneous* location subsequence as the one in which all locations are assigned to the same state at level 2 in the model. The goal of this step is to identify a set of non-repetitive and homogeneous location subsequences that appear at least $\gamma \cdot |Q|$ times in the location sequences in Q. In the following, we describe an approach that requires only a *single* traverse over each location sequence in set Q. It involves two steps. In the *first* step, we calculate the number of times that a non-repetitive and homogeneous location subsequence appears in the history mobility traces. We use counter $\eta(h)$ to denote the number of appearances of subsequence h. The pseudo-code of the first step is provided in Algorithm 1 ($\circ$ denotes the string concatenation operator). We use a trie to maintain each location subsequence added into set I (line 8). For each node t in the trie, the path from the root node to node t actually corresponds to a location subsequence. Let $\hbar(t)$ be such a location subsequence that corresponds to node t. We abuse notation η slightly by defining $\eta(t)$ as $\eta(\hbar(t))$. The output of the first step is a forest in which each tree represents a set of location subsequences starting from the same location.

In the *second* step, we simplify the forest derived from the first step to determine the set of non-repetitive and homogeneous location subsequences that appear at least $\gamma \cdot |Q|$ times. While Algorithm 1 is running, we use a linked list to keep all the leaf nodes of the same depth in the whole forest. Let $L(d)$ denote the list of leaf nodes with depth d. For each node t in a trie, we also keep variable $\tilde{\eta}(t)$, which is initialized to 0. $\eta(t)$ and $\tilde{\eta}(t)$ are called the *primary* and *secondary* counters of node t, respectively. The pseudo-code of the second step is given in Algorithm 2. For simplicity, given a location subsequence h, we use h^{-1} to denote the location subsequence after the first location symbol is removed.

Algorithm 2 Derive the set of subsequences with at least $\gamma \cdot |Q|$ appearances.

```
 1: $d_{max} \leftarrow$ max. depth among all tries from Algorithm 1
 2: for $d = d_{max}$ to 1 do
 3:   while $L(d)$ is not empty do
 4:     $t \leftarrow$ extract a leaf node from $L(d)$, $h \leftarrow \hbar(t)$
 5:     if $\eta(t) + \tilde{\eta}(t) < \gamma \cdot |Q|$ then
 6:       Find corresponding node $t'$ for $h^{-1}$; if it is not in
          the forest yet, update the forest and $L(d-1)$
 7:       $\tilde{\eta}(t') \leftarrow \tilde{\eta}(t') + \eta(t) + \tilde{\eta}(t)$, $t'' \leftarrow$ the parent node
          of node $t$
 8:       $\eta(t'') \leftarrow \eta(t'') + \eta(t)$
 9:       delete node $t$ from the forest
10:     end if
11:   end while
12: end for
```



(1) After the first step (2) After processing (1,2,3,4)

(3) After processing (1,2,3) (4) After processing (2,3,4)

(5) After processing (1,2) (6) After processing (2,4),(3,4)

Figure 1: Feature extraction ($\gamma \cdot |Q| = 3$). Beside each node t is the pair $(\eta(t), \tilde{\eta}(t))$.

We use an example to explain the two steps of the feature extraction. Suppose there is only one location subsequence in Q, which is $(1,2,3,4,1,2,3,2,3,2,4)$. For simplicity, it is assumed that all the locations belong to the same level-2 state. The algorithm works as shown in Fig. 1. Clearly, the algorithm is able to identify the only subsequence $(2,3)$ that appears 3 times. The example also helps explain why we need to maintain secondary counters. First, we note that subsequence $(1,2,3,4)$ does not have $\gamma \cdot |Q|$ appearances. We thus split it into two different subsequences $(1,2,3)$ and $(2,3,4)$. These two subsequences, however, both contain subsequence $(2,3)$. Hence, should we increase the primary counter of each of them, subsequence $(2,3)$ would be counted twice later for the same appearance. To avoid that, we increase the primary counter of $(1,2,3)$ and the secondary counter of $(2,3,4)$ by the number of appearances of $(1,2,3,4)$. Later when we process subsequence $(2,3,4)$, regarding its secondary counter, we *only* consider the subsequence derived after its first location symbol is removed, which is $(3,4)$. As the last location symbol (4 in our example) is always included when the secondary counter of a node is considered, no common subsequence is counted twice.

We define a *non-trivial* tree to be a tree that has more than one node in it. The output from feature extraction is a forest $\mathcal{F}$ in which each connected component is either (1) a non-trivial tree in which the location subsequence represented by each leaf node has at least $\gamma \cdot |Q|$ appearances, or (2) a single node. All non-repetitive and homogeneous location subsequences that have at least $\gamma \cdot |Q|$ appearances can easily be derived from the output forest $\mathcal{F}$.

The time complexity of our algorithm for feature extraction is $O(N \cdot |Y| \cdot \log(|Y|))$, where N is the sum of the lengths of all the subsequences in Q and $|Y|$ we recall is the total number of different locations in Q.

Level-3/4 state definition. Based on the type of a connected component in forest $\mathcal{F}$, we define states in the HHMM accordingly as follows.

Case (1): non-trivial trees. For each non-trivial tree T, we define an internal state $s^{(3)}$ at level 3. The parent state of $s^{(3)}$ is the level-2 state to which all locations represented by nodes in tree T are assigned. For each node t in T, we create a production state $s^{(4)}$ at level 4 and let $s^{(3)}$ be the parent state of $s^{(4)}$. We add a vertical transition probability of 1 from the level-3 state $s^{(3)}$ that corresponds to T to the level-4 state that corresponds to the root node of tree T. For level-3 state $s^{(3)}$, we also define an end state at level 4, $s^{(4)}_{end}$. So $e(s^{(3)}) = s^{(4)}_{end}$. For each leaf node t in T and its corresponding level-4 state $s^{(4)}$, we add a horizontal transition from $s^{(4)}$ to end state $s^{(4)}_{end}$ with probability 1.

If node t_1 is the parent node of node t_2 in tree T and their corresponding states at level-4 are $s^{(4)}_1$ and $s^{(4)}_2$, respectively, we add a horizontal transition from $s^{(4)}_1$ to $s^{(4)}_2$ and define the transition probability $a(s^{(4)}_1, s^{(4)}_2)$ as:

$$a(s^{(4)}_1, s^{(4)}_2) = \frac{\Gamma_\eta(t_2) + \tilde{\eta}(t_2)}{\max\{\Gamma_\eta(t_1) + \tilde{\eta}(t_1), \sum_{t \in D(t_1)}\{\Gamma_\eta(t) + \tilde{\eta}(t)\} + \eta(t_1)\}}.$$

If the corresponding level-4 state of node t in tree T is $s^{(4)}$, we create a horizontal transition from state $s^{(4)}$ to end state $s^{(4)}_{end}$ with probability:

$$a(s^{(4)}, s^{(4)}_{end}) = 1 - \frac{\Gamma_\eta(t_2) + \tilde{\eta}(t_2)}{\max\{\Gamma_\eta(t_1) + \tilde{\eta}(t_1), \sum_{t \in D(t_1)}\{\Gamma_\eta(t) + \tilde{\eta}(t)\} + \eta(t_1)\}}.$$

Case (2): single nodes. For each single node t, if $\eta(t) + \tilde{\eta}(t) \geq \gamma \cdot |Q|$, we create a production state at level 3; this state produces the location symbol corresponding to node t with probability 1. The parent node of this production state is the corresponding level-2 state to which the location represented by node s is assigned.

For each level-2 state $s^{(2)}$, we use $\widehat{X}(s^{(2)})$ to denote the entire set of single nodes t that represent locations belonging to state $s^{(2)}$ and where $\eta(t) + \tilde{\eta}(t) < \gamma \cdot |Q|$. If $\widehat{X}(s^{(2)})$ is not empty, we create an aggregate production state at level 3 for state $s^{(2)}$. The probability of producing a location symbol represented by node t at this state is $\frac{\eta(t) + \tilde{\eta}(t)}{\sum_{t' \in \widehat{X}(s^{(2)})}\{\eta(t') + \tilde{\eta}(t')\}}$.

Fig. 2 depicts the states created at levels 3 and 4 for the example shown in Fig. 1(6). The $s^{(1)}_2$ state is created for single nodes 1, 3 and 4; the $s^{(2)}_2$ state is created for the tree containing nodes 2 and 3.

Level-3 parameter initialization. To derive initial state transition probabilities at level 3, we use the longest matching method to map each location subsequence in set Q into a sequence of level-3 states in the HHMM. The approach is shown in Algorithm 3. The state sequence that corresponds to the j-th location subsequence in Q is given as Δ_j.

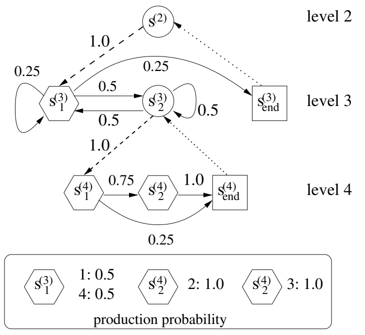


Figure 2: The HHMM created for the example in Fig. 1(6). (The level-1 state ignored)

Algorithm 3 Map each location subsequence in set Q to a level-3 state sequence.

```
1: for $j = 1$ to $|Q|$ do
2:     $h \leftarrow$ empty subsequence, state sequence $\Delta_j \leftarrow \emptyset$
3:     for $k = 1$ to $|H_j| + 1$ do
4:         if $h \circ \tilde{c}_k(j)$ is not valid in forest $\mathcal{F}$ or $k = |H_j| + 1$
           then
5:             $t \leftarrow$ the level-3 state that generates $h$, $\Delta_j \leftarrow \Delta_j \circ t$
6:         else
7:             $h \leftarrow h \circ \tilde{c}_k(j)$
8:         end if
9:     end for
10: end for
```

We set the horizontal state transition probability from level-3 states $s_i^{(3)}$ to $s_j^{(3)}$ when they share the same level-2 parent state (i.e., $p(s_i^{(3)}) = p(s_j^{(3)})$), as follows:

$$a(s_i^{(3)}, s_j^{(3)}) = \frac{\sum_{1\le k\le |Q|}\sum_{1\le u<|\Delta_k|} 1\{\Delta_k[u]=s_i^{(3)} \wedge \Delta_k[u+1]=s_j^{(3)}\}}{\sum_{1\le k\le |Q|}\sum_{1\le u\le |\Delta_k|} 1\{\Delta_k[u]=s_i^{(3)}\}}$$

For simplicity, we define $p(\Delta_k[|\Delta_k|+1]) = p(\Delta_k[0]) = null$, where $1 \le k \le |Q|$ and $null$ is a non-existing virtual state. Then, horizontal state transition probability from a level-3 state $s^{(3)}$ to the end state $s_{end}^{(3)}$ that has the same parent state as $s^{(3)}$ is given by:

$$a(s^{(3)}, s_{end}^{(3)}) = \frac{\sum_{1\le k\le |Q|}\sum_{1\le u\le |\Delta_k|} 1\{\Delta_k[u]=s^{(3)} \wedge p(\Delta_k[u+1])\neq p(s^{(3)})\}}{\sum_{1\le k\le |Q|}\sum_{1\le u\le |\Delta_k|} 1\{\Delta_k[u]=s_i^{(3)}\}}.$$

Finally, the vertical state transition from the parent state of a level-3 state $s^{(3)}$ to it is given as follows:

$$\pi(p(s^{(3)}), s^{(3)}) = \frac{\sum_{1\le k\le |Q|}\sum_{1\le u\le |\Delta_k|} 1\{\Delta_k[u]=s^{(3)} \wedge p(\Delta_k[u-1])\neq p(s^{(3)})\}}{\sum_{1\le k\le |Q|}\sum_{1\le u\le |\Delta_k|} 1\{p(\Delta_k[u])=p(s^{(3)}) \wedge p(\Delta_k[u-1])\neq p(s^{(3)})\}}.$$

4.3 Phase III: Parameter Re-estimation

In the previous section, we have discussed how to initialize low level parameters by the longest match approach. Parameters initialized as such, however, may not maximize the overall probability of all the observation sequences in Q. Our goal in Phase III is to train the HHMM so that it perfectly characterizes the observation sequences in Q, i.e.,

$$\max_\lambda \mathbb{P}\{Q|\lambda\} = \Pi_{j=1}^m \mathbb{P}\{H_j|\lambda\}, \tag{5}$$

where λ is the HHMM being trained.

We do not directly use existing parameter re-estimation algorithms [4] here for two reasons. *First*, we notice that observation symbols generated from a level-2 sub-model never overlap with those from another level-2 sub-model. It is thus unnecessary to train the whole HHMM from the root. *Second*, the high time complexity of the generalized Baum-Welch algorithm [4] is undesirable in a practical setting, but alternative solutions such as the one proposed in [10] are much more difficult to implement. Against this backdrop, we adapt the original algorithm in [4] and improve its efficiency by exploiting some domain-specific knowledge. Based on the level-2 state that a location symbol belongs to, we divide each location sequence H_j in Q into a sequence of location subsequences:

$$H_j = L_1^{(j)} \circ L_2^{(j)} \circ ... \circ L_{t_j-1}^{(j)} \circ L_{|H_j|}^{(j)}, \tag{6}$$

where $L_i^{(j)}$ $(1 \le i \le t_j)$ is a location subsequence in which all locations belong to the same level-2 state and no two successive subsequences $L_i^{(j)}$ and $L_{i+1}^{(j)}$ correspond to the same level-2 state. For brevity, each subsequence $L_i^{(j)}$ where $1 \le i \le t_j$ is called a *maximum level-2 subsequence*. We use $\phi(L)$ to denote the level-2 state that corresponds to a maximum level-2 subsequence L. We also use $L[i]$ to denote the i-th location symbol in subsequence L, and $L[i..j]$ to denote the i-th through the j-th location symbols in subsequence L.

Calculate α, β, ξ and χ variables given an observation sequence L. Given the current parameterized HHMM λ, a maximum level-2 subsequence L, and its corresponding level-2 state $\phi(L)$, our modified Baum-Welch algorithm calculates four path variables: α, β, χ, and ξ. We also use $d_{max}^{\mathcal{F}}$ to denote the height of output forest $\mathcal{F}$. Recall that $p(s)$ gives the parent state of state s. $\alpha(L, i, i+k, s_j, p(s_j))$ is defined as the probability that observation $L[i..i+k]$ is produced and the model reaches state s_j after the $(i+k)$-th symbol given the condition that the model starts with state $p(s_j)$ at the i-th symbol. α-variables are calculated recursively as follows:

$$\alpha(L, i, i+k, s_j, p(s_j)) =$$
$$\begin{cases} \pi(p(s_j),s_j)\cdot b_{s_j}(L[i]) & ① \\ \left[\sum_{s_t\in D(p(s_j))} \alpha(L,i,i+k-1,s_t,p(s_j))a(s_t,s_j)\right]\cdot b_{s_j}(L[i+k]) & ② \\ \pi(p(s_j),s_j)\cdot\left[\sum_{s_t\in D(s_j)} \alpha(L,i,i,s_t,s_j)a(s_t,e(s_j))\right] & ③ \\ \sum_{l=\max\{0,k-d_{max}^{\mathcal{F}}\}}^{k-1} \left[\sum_{s_t\in D(p(s_j))} \alpha(L,i,i+l,s_t,p(s_j))a(s_t,s_j)\right]\times & \\ \quad \left[\sum_{s_r\in D(s_j)} \alpha(L,i+l+1,i+k,s_r,s_j)a(s_r,e(s_j))\right] & \\ \quad +\pi(p(s_j),s_j)\left[\sum_{s_r\in D(s_j)} \alpha(L,i,i+k,s_r,s_j)a(s_r,e(s_j))\right] & ④ \end{cases}$$

where the four conditions are: $k = 0$ and $\delta(s_j) = production$ (Case ①), $k > 0$ and $\delta(s_j) = production$ (Case ②), $k = 0$ and $\delta(s_j) = internal$ (Case ③), and $k > 0$ and $\delta(s_j) = internal$ (Case ④).

Comparing the above formula for α variables with the original generalized Baum-Welch algorithm, we note that the key difference is the first item in Case ④: instead of summing from 0 to $k-1$ in the original algorithm, our algorithm starts from $\max\{0, k - d_{max}^{\mathcal{F}}\}$. This is because in Case ④, state s_j must be a level-3 state, which represents some frequently observed mobility patterns. As the height of output forest $\mathcal{F}$ is $d_{max}^{\mathcal{F}}$, the maximum number of symbols generated by a level-3 state is at most $d_{max}^{\mathcal{F}}$. We thus do not need to evaluate any $\alpha(i+l+1, i+k, s_r, s_j)$ where $(i+k)-(i+l+1)+1 > d^{\mathcal{F}}_{max}$.

Importantly, such a slight modification enables us to reduce the time complexity of computing α variables from $O(N \cdot M \cdot T^3)$ in the original algorithm to $O(N \cdot M \cdot T \cdot d^{\mathcal{F}}_{max})$, where T is the number of symbols in the observation sequence, N is the number of states in the model, and M is the maximum number of child states of any state in the model [2]. In our solution, for any $\alpha(L, i, i+k, s_j, p(s_j))$, if state s_j is a level-3 state, we only compute α-variables for $i = 1$ (because other α-variables are not needed); if state state s_j is a level-4 state, we compute α-variables only when $0 \le k < d^{\mathcal{F}}_{max}$ and $1 \le i \le T$. Based on the observation from our experimental results, $d^{\mathcal{F}}_{max}$ is usually much smaller than the number of symbols that a typical location sequence could have. Hence, our modification significantly improves the computation of α-variables.

Other variables in the model, including β, ξ and χ, can be calculated with reduced computational complexity, which are shown in Appendix A.

Parameter re-estimation. Equation (8) describes how to compute the probability of a maximum level-2 subsequence given the sub-model rooted at its corresponding level-2 state. We now discuss how to compute the probability of a location sequence in Q under the current parameterized model λ. For any location sequence H_j that is dissembled as in Equation (6), we have:

$$\mathbb{P}\{H_j \mid \lambda\} = \pi(s^{(1)}, \phi(L_1^{(j)})) \times \Pi_{i=1}^{|H_j|} \mathbb{P}\{L_i^{(j)} | \phi(L_i^{(j)})\} \times \Pi_{i=1}^{|H_j|-1} a(\phi(L_i^{(j)}), \phi(L_{i+1}^{(j)})) \times a(\phi(L_{|H_j|}^{(j)}), e(s^{(1)})). \quad (7)$$

The high-level description of parameter re-estimation is given in Algorithm 4. More details are given in Appendix B. The approach in which we update the model parameters differs from the original solution [4] in two ways. First, we do not recompute level-2 parameters as derived in Phase I (see Section 4.1). Second, the parameter re-estimation process involves multiple observations; different from existing work, our approach treats a maximum level-2 subsequence as a separate observation and all maximum level-2 subsequences are evenly weighted with respect to their corresponding level-2 states.

In the algorithm, we terminate the iterations if either of the two conditions is satisfied: (1) the number of iterations reaches predefined variable MAX_TIMES; or (2) the improvement of the solution between two successive iterations is below a predefined threshold ϵ.

5. MOBILITY ANOMALY DETECTION

With a trained HHMM λ for a mobile device, we now discuss how to use it to detect mobility anomalies. Suppose that the test location sequence is $\widehat{H}$, which has $|\widehat{H}|$ location symbols in it. Similar to Equation (6), we first dissemble it into a sequence of maximum level-2 subsequences. Then, we compute $\mathbb{P}\{\widehat{H} \mid \lambda\}$, the probability that observation $\widehat{H}$ is produced under model λ as in Equation (7). We define $\varpi(h)$, where h is a location sequence, as $\frac{log(\mathbb{P}\{h|\lambda\})}{log(|h|)}$. We also say that $\varpi(h)$ is the *ϖ-value* of sequence h. We decide

[2] In the original paper [4], factor M is ignored.

Algorithm 4 Re-estimate parameters in HHMM λ

```
1:  $prev\_x \leftarrow uninitialized$
2:  for $i = 1$ to $MAX\_TIMES$ do
3:      $x \leftarrow \sum_{j=1}^{|Q|} log(\mathbb{P}\{H_j \mid \lambda\})$
4:      if $prev\_x \neq uninitialized$ and $|\frac{x - prev\_x}{prev\_x}| < \epsilon$ then
5:          terminate
6:      else
7:          $prev\_x \leftarrow x$
8:          Reestimate parameters as shown in Appendix B
9:      end if
10: end for
```

whether observation $\widehat{H}$ is anomalous by testing $\varpi(\widehat{H})$: if it is greater than a predefined threshold ω, $\widehat{H}$ is normal; otherwise, it is anomalous. We call ω the *anomaly threshold parameter*. Here we normalize the probability by the length of the observed location sequence so that the model is not biased against long sequences.

The selection of threshold ω imposes significant impact on the accuracy of the detection result: if ω is too large, the detection rate becomes too low; otherwise, if ω is too small, the false positive rate becomes too high. In our approach, we associate ω with the production probabilities of those trained sequences in Q. More specifically, we sort the ϖ-values of all the training location sequences in Q, and let ω be the $(\lfloor \kappa \cdot |Q| \rfloor + 1)$-th largest value among them.

Note that if we apply the previous detection scheme, it is still possible to produce high false positive rates under three special scenarios: *overfitting*, *traveling*, and *moving*. Next, we discuss the details of these scenarios and propose some modifications on the detection strategy.

Scenario 1: overfitting. Overfitting due to insufficient training data can negatively affect the detection accuracy. For instance, a user of mobile devices can appear shortly at a location that has never been seen in previous mobility traces or drive along a route that he just knows from his friends. If this event occurs, the production probability of the observed mobility trace is always 0, even if such an event occurs infrequently in the test trace.

To reduce false alarms due to such rare events, we slightly modify the trained HHMM by introducing *low probability transitions* and *a wildcard location symbol.* After the model has been trained, for each level-2 or level-3 state s_j, we update the parameters as follows: if the vertical probability to state s_j is 0, then we let it be σ times the lowest positive vertical probability from its parent state to any of its sibling states; if the horizontal probability from states s_j to s_k is 0, then we let it be σ times the lowest positive horizontal probability from state s_j to any of its sibling states; after all these, we renormalize all the transition probabilities.

For a level-4 state s_j, we do not update the vertical transition probability from its parent node to s_j, even if it is 0. We, however, do not allow zero horizontal transition probability from s_j to the end state under the same parent state. If $a(s_j, e(p(s_j))) = 0$, we do the following:

$$\hat{a}(s_j, s_k) = \begin{cases} \frac{\sigma \min_{s_r \in D(p(s_j))}\{a(s_j,s_r)|a(s_j,s_r)>0\}}{1+\sigma \min_{s_r \in D(p(s_j))}\{a(s_j,s_r)|a(s_j,s_r)>0\}} & \text{if } s_k = e(p(s_j)) \\ \frac{a(s_j,s_k)}{1+\sigma \min_{s_r \in D(p(s_j))}\{a(s_j,s_r)|a(s_j,s_r)>0\}} & \text{if } s_k \neq e(p(s_j)) \end{cases}$$

It is also possible that a location symbol in the test sequence does not appear in the history mobility traces. We thus introduce a wildcard symbol '*' to indicate any symbol that has not been observed in the training dataset. Recall that at level 2 state $s_a^{(2)}$ produces the location symbols in all the clusters with exposure rates lower than than the given threshold θ. Let $s_{a,a}^{(3)}$ indicate the level-3 state that is a child state of $s_a^{(2)}$ and produces location symbols with less than $\gamma \cdot |Q|$ appearances. We let $s_{a,a}^{(3)}$ produce the wildcard symbol with probability σ times the lowest probability that a symbol is produced under state $s_{a,a}^{(3)}$ (normalization is also necessary here); if $s_{a,a}^{(3)}$ does not exist before, then we add such a state, introduce some low transition probabilities to it as described above, and let its production probability of the wildcard symbol be 1.

Scenario 2: traveling. Another special case that can lead to high false positive rates is when people travel with their mobile devices during holidays or some business trips. In our approach, we exploit the fact that false alarms tend to occur in consecutive days when people travel with their mobile devices. Hence, when mobility anomalies are detected consecutively, our approach only generates a new alarm for every τ consecutive observations. For example, if τ is 5 and through days 1 to 10, the HHMM detects mobility anomaly on each of these days, alarms are generated only on days 1 and 6. Apparently, a large τ can reduce false positive rates but may also delay the detection of true mobility anomalies.

Scenario 3: moving. A mobile device user may change her home or work location sometimes. Such moving behavior poses another challenge to mobility anomaly detection. When people move, the history mobility traces become less valuable. To prevent high false positive rates under such circumstance, our current solution requires model retraining after frequent observations of false alarms. In our implementation, if more than 20% of the past 10 or more observations produce false alarms, or if more than 15% of the past 20 or more observations produce false alarms, we retrain the model by incorporating the new observations.

6. EVALUATION

We now evaluate how effectively Mobi-Watchdog detects mobility anomalies in wireless networks. We use the realistic mobility traces collected from the Reality Mining project [13] in our experiments[3]. The dataset provides cell-level mobility traces of 89 people, who were either faculty or students in the MIT Media Laboratory or students studying at the adjacent MIT Sloan Business School. The original mobility traces were collected by the smart phones and a person's daily mobility record can thus have hundreds or even thousands of cell numbers. In reality, however, a cellular network often tracks a user's location either at a coarse spatial level (e.g., location areas) or at a coarse temporal level (e.g., periodic cell reporting) so that the mobility management cost can be minimized [20]. In our experiments, we sample each person's cell-level locations every 30 minutes and use the results as her mobility records witnessed by the network.

Some daily mobility traces are incomplete in the original dataset, possibly due to data corruption. We remove those daily records that have less than 10 cells in the sampled dataset. Among the mobility traces of 89 people, we choose 68 of them that span at least 60 days. We split each person's mobility traces into halves: one for training purpose and the other for testing purpose. The following parameters are constantly set in our experiments: $\theta = 0.1$ (Sec. 4.1), $\gamma = 0.5$ (Sec. 4.2), $MAX\_TIMES = 10$ (Sec. 4.3), $\epsilon = 0.01$ (Sec. 4.3), and $\sigma = 0.1$ (Sec. 5).

[3]Actually, the MIT Reality Mining dataset is the only public cell-level human mobility traces we know of.

6.1 Accuracy

Detection rates of standard attacks. To evaluate the effectiveness of our method in detecting mobility anomaly, we first use the mobility model derived from each of those 68 people's training datasets to check whether mobility sequences in any other's test dataset are anomalous. We call such attacks *standard attacks*. We vary parameter κ, which controls the anomaly threshold (see Section 5), between 0 and 0.1.

Fig. 3 depicts the average detection rates of the scenarios in which a different person holds the mobile device. The impact of κ on the detection rate is obvious: a higher κ leads to a higher detection rate. This is because the higher κ is, the lower the anomaly threshold ω is. From Fig. 3, we observe that for most of the settings, the detection rate is above 90%, which suggests that Mobi-Watchdog can be very effective in a more realistic setting. As the mobility traces we used in our experiments resulted from people that worked or studied at the same place, we believe that the detection rate will be higher if the test mobility traces come from a more diverse set of people.

Detection rates of mimicry attacks. In the second set of experiments, we evaluate the effectiveness of our scheme in detecting mimicry attacks. We assume that in a mimicry attack, the number of appearances of mobile devices in each individual cell can be imitated statistically but the order of appearances remains unknown to the adversary. In the experiments, we randomly change the order of those cells on a daily basis in each person's sampled training dataset, assuming that the adversary has perfect knowledge on what cells visited by the testing mobile device are visible to the network.

It is expected that such a strong assumption on the adversary's knowledge should lead to lower detection rates of mimicry attacks than those of standard attacks. This is confirmed by the experimental results illustrated in Fig. 4. As we vary κ between 0 and 0.1, the detection rate of mimicry attacks falls between about 50% and 70%. A closer examination on the training mobility traces reveals that some daily mobility records have frequent appearances of the same cell ID. Hence, reshuffling the order of the cell IDs in these daily mobility records can not change the original training traces significantly. This also contributes to the lower detection rate of mimicry attacks.

False positive rates. We use the mobility model derived from each person's training datasets to check whether mobility sequences in the same person's test dataset are anomalous. In this set of experiments, we also vary parameter κ between 0 and 0.1. In addition, we test how the false positive rates respond to different τ values. Recall that a new alarm is generated for every τ consecutive observations that have been found anomalous (see Scenario 2 in Section 5).

The results are illustrated in Fig. 5. As increasing parameter κ lowers the anomaly threshold, it is not surprising that false alarm rates monotonically grow as κ increases, re-

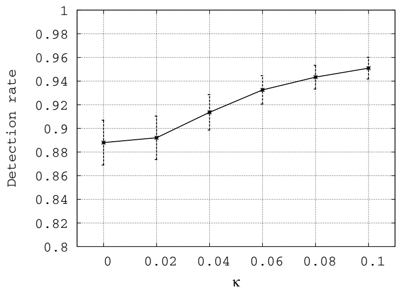


Figure 3: Detection rates of standard attacks (95% confidence interval)

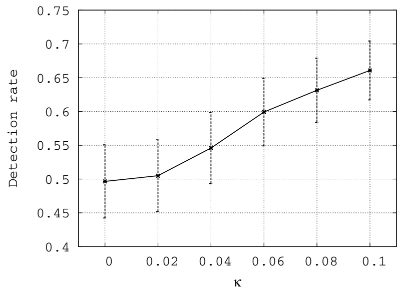


Figure 4: Detection rates of mimicry attacks (95% confidence interval)

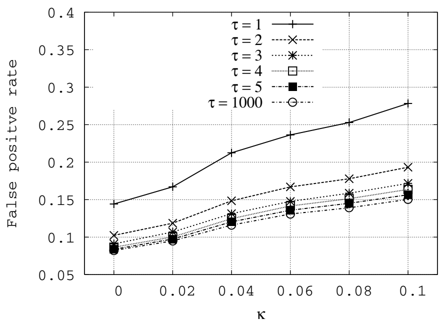


Figure 5: False alarm rates

gardless of how we choose τ. On the other hand, increasing parameter τ helps reduce the false alarm rates, and this is obvious especially when τ is small. For instance, when κ is 0, the false positive rate falls from 14% to 9% if we change τ from 1 to 3.

The experimental results reveal that Mobi-Watchdog detects mobility anomalies with high accuracy but with only limited false alarm rates. For example, when we choose κ to be 0 and τ to be 3, our approach can detect about 89% of the standard attacks and 50% of the mimicry attacks with the average false alarm rate being only 9%[4]. In a practical setting where mobility anomaly detection is an optional service, one may decide whether to use this service or choose some key model parameters to balance her acceptable false alarm rate against the anomaly detection rate.

Comparison with LZ-based solution. A mobility anomaly detection scheme that is based on the Lempel-Ziv (LZ) algorithms was proposed in [17]. We use the same configurations as in the experiments in [17] and have the following results when the LZ-based solution is used on the same datasets: the average false positive rate is 12%, the average detection rate of standard attacks is 76%, and the average detection rate of mimicry attacks is only 12%. Given the same false alarm rate level, Mobi-Watchdog detects both standard and mimicry attacks with higher accuracy than the LZ-based solution.

[4]Suppose that once an alarm is raised, the network simply requires the user of the mobile device to reauthenticate herself by resending the password to it. A false positive rate of 9% means that a user needs to send her password only about 3 times within a month, which we believe is acceptable in many cases.

Step	Execution Time
Phases I&II	0.045 sec
Phase III	0.296 sec
Detection	0.019 sec

Table 1: Computational cost of Mobi-Watchdog in mobility anomaly detection

6.2 Computational Cost

We evaluate the computational cost of Mobi-Watchdog in mobility anomaly detection as follows. We perform the same set of experiments in detecting mimicry attacks. For each setup, we run Mobi-Watchdog 20 times. We use an ordinary PC with 2GHz CPU and 1.5GB memory. The Operating System is Redhat Enterprise Linux WS release 4. In the experiments, the average numbers of observations in each training dataset and each test dataset are both 66. We measure the execution times spent on model construction (Phases I and II in Section 4), parameter re-estimation (Phase III in Section 4), and anomaly detection on the test dataset (Section 5), respectively. The results are illustrated in Table 1. Clearly, Mobi-Watchdog is very efficient: the training phase finishes within less than 0.4 second and the testing phase takes 0.0003 second per day. Consider a wireless network that has 10 million subscribers. With an ordinary commodity PC, Mobi-Watchdog takes less than one hour to perform mobility anomaly detection for all of them each day. The results suggest that Mobi-Watchdog is a viable solution to mobility anomaly detection in large-scale wireless networks with many mobile devices.

7. CONCLUSIONS

In this paper we propose an architecture called Mobi-Watchdog to detect anomalous mobility patterns in wireless mobile networks. Provided past mobility traces of a mobile device, Mobi-Watchdog uses clustering techniques to identify the high-level structure of its mobility and then trains a parameterized HHMM. Mobi-Watchdog raises alerts by requesting the device holder to reauthenticate himself when an observed mobility trace significantly deviates from the trained model. Experimental results show that Mobi-Watchdog can detect mobility anomalies with high probability and reasonably low false alarm rates.

The model-based scheme in Mobi-Watchdog can further be improved. Currently Mobi-Watchdog detects mobility anomalies by looking at transition probabilities among locations visited by a mobile device. A more comprehensive model takes staying times at each location into account. A natural extension can be using continuous-time hierarchical hidden Markov models. It is important, though, to strike a balance between model complexity and system scalability. A complicated model, although providing better detection accuracy, can impose high computational cost which renders it impractical for large-scale wireless mobile networks.

Mobility anomaly detection is not a panacea for all security-related issues with mobile devices. First, if the thief of a mobile device is familiar with the owner's mobility pattern, he can simply follow the same route as the owner, which makes any mobility anomaly detection system completely futile. Second, for users whose mobility patterns exhibit a lot of irregularities (e.g., taxi drivers), Mobi-Watchdog may produce high false alarm rates. In our design, however, Mobi-Watchdog is intended to provide an optional protection service to those people whose daily itineraries follow similar routes. Third, as Mobi-Watchdog is currently designed to detect mobility anomalies on a daily basis, it may not prevent those malicious behaviors that are performed before an alert is flagged. Even so, Mobi-Watchdog is still useful for those users who lost their mobile devices but are unable to report the incidents in time to their service providers for service deactivation. In the future, we plan to improve Mobi-Watchdog by incorporating other techniques into it, such as lightweight device-resident intrusion detection systems and service-aware traffic anomaly detection tools.

8. REFERENCES

[1] S. V. Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.

[2] A. J. Enright, S. V. Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7), 2002.

[3] F. Erbas, K. Kyamakya, J. Steuer, and K. Jobmann. On the user profiles and the prediction of user movements in wireless networks. In *Proceedings of IEEE PIMRC*, 2002.

[4] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32, 1998.

[5] M. C. González, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 2008.

[6] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic models for mobile ad hoc networks. In *Proc. of MobiCom*, 2003.

[7] J. Kim. Realistic mobility modeling and simulation for mobile wireless network in urban environments. Master's thesis, University of Deleware, 2005.

[8] Lin Liao and Dieter F. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Artificial Intelligence*, 2007.

[9] `http://www.washingtonpost.com/wp-dyn/content/article/2005/07/24/AR2005072%401135.html`.

[10] K. Murphy and M. Paskin. Linear time inference in hierarchical HMMs. In *Advances in Neural Information Processing Systems 13*. MIT Press, Cambridge, MA, 2002.

[11] M. Musolesi, S. Hailes, and C. Mascolo. An ad hoc mobility model founded on social newtork theory. In *Proceedings of MSWiM'04*, 2004.

[12] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, February 1989.

[13] Reality mining project. http://reality.media.mit.edu/.

[14] S. Schonfelder. Some notes on space, location and travel behavior. In *Swiss Transport Research Conference*, Monte Verita, Ascona, 2001.

[15] J. Scourias and T. Kunz. An activity-based mobility model and location management simulation framework. In *Proceedings of MSWiM '99*, New York, NY, USA, 1999.

[16] B. Sun, Y. Xiao, and R. Wang. Detection of fraudulent usage in wireless networks. *IEEE Trans. on Vehicular Technology*, 56(6), 2007.

[17] B. Sun, F. Yu, K. Wu, Y. Xiao, and V. C. M. Leung. Enhancing security using mobility-based anomaly detection in cellular mobile networks. *IEEE Trans. on Vehicular Technology*, 55(3), 2006.

[18] `http://cbs5.com/consumer/wireless.runaround.2.451677.html`.

[19] H. Zhang, H. Yu, D. Xiong, and Q. Liu. HHMM-based chinese lexical analyzer. In *Proceedings of the 2^{nd} SIGHAN workshop on Chinese language processing*, 2003.

[20] J. Zhang. *Handbook of Wireless Networks and Mobile Computing*, chapter Location Management in Cellular Networks. John Wiley & Sons, Inc., 2002.

Appendix A: Calculate β, ξ and χ variables

$\beta(L, i, i+k, s_j, p(s_j))$ is defined as the probability that observation $L[i..i+k]$ is produced given the condition that the model starts with state s_j at the i-th symbol and reaches state $p(s_j)$ after the $(i+k)$-th symbol. β-variables are calculated recursively as follows:

$$\beta(L, i, i+k, s_j, p(s_j)) =$$

$$\begin{cases} b_{s_j}(L[i])\cdot a(s_j, e(p(s_j))) & ① \\ b_{s_j}(L[i])\times\left[\sum_{s_t\in D(p(s_j))} a(s_j,s_t)\beta(L,i+1,i+k,s_t,p(s_j))\right] & ② \\ \left[\sum_{s_t\in D(s_j)} \pi(s_j,s_t)\beta(L,i,i,s_t,s_j)\right] a(s_j,e(p(s_j))) & ③ \\ \sum_{l=\max\{0,k-d^{\mathcal{F}}_{max}\}}^{k-1} \left[\sum_{s_t\in D(s_j)} \pi(s_j,s_t)\beta(L,i,i+l,s_t,s_j)\right] & \\ \qquad \left[\sum_{s_r\in D(p(s_j))} a(s_j,s_r)\beta(L,i+l+1,i+k,s_r,p(s_j))\right] & \\ \qquad + \left[\sum_{s_r\in D(s_j)} \pi(s_j,s_r)\beta(L,i,i+k,s_r,s_j)\right] a(s_j,e(s_j)) & ④ \end{cases}$$

The conditions in the above equation are the same as in the computation of α-variables (i.e., Case ① to ④). Similar to α-variables, β-variables can be calculated within time $O(N\cdot M\cdot T\cdot d^{\mathcal{F}}_{max})$. Then, the likelihood of a maximum level-2 subsequence L under level-2 state $\phi(L)$ can be calculated as follows:

$$\begin{aligned} \mathbb{P}\{L|\phi(L)\} &= \sum_{s_t\in D(\phi(L))} \alpha(L,1,|L|,s_t,\phi(L))a(s_t,e(\phi(L))) \\ &= \sum_{s_t\in D(\phi(L))} \pi(\Phi(L),s_t)\beta(L,1,|L|,s_t,\phi(L)). \qquad (8) \end{aligned}$$

For parameter reestimation, we define the following variables: $\eta_{in}(L, i, s_j, p(s_j))$ gives the probability that observation $L[1..i-1]$ is produced and the model reaches state s_j at observation $L[i]$ under model λ; similarly, $\eta_{out}(L, i, s_j, p(s_j))$ gives the probability that the model finishes with state s_j on observation $L[i]$ and observation sequence $L[i+1, |L|]$ is produced under model λ; $\xi(L, i, s_j, s_k, p(s_j))$, where $p(s_j) =$

$$\eta_{in}(L,i,s_j,p(s_j))=$$

$$\begin{cases}
\pi(s_j,p(s_j)) & \text{if } i{=}1 \text{ and } p(s_j){\in}S^{(2)} \\
\sum_{s_t\in D(p(s_j))} \alpha(L,1,i{-}1,s_t,p(s_j))a(s_t,s_j) & \text{if } i{>}1 \text{ and } p(s_j){\in}S^{(2)} \\
\eta_{in}(L,i,p(s_j),p(p(s_j)))\pi(p(s_j),s_j) & \text{if } i{=}1 \text{ and } p(s_j){\notin}S^{(2)} \\
\sum_{s=\max\{1,i-d^{\mathcal{F}}_{max}\}}^{i-1} \eta_{in}(L,s,p(s_j),p(p(s_j)))\Big[\sum_{s_t\in D(p(s_j))} \alpha(L,s,i{-}1,s_t,p(s_j))a(s_t,s_j)\Big] & \text{if } i{>}1 \text{ and } p(s_j){\notin}S^{(2)} \\
\qquad +\eta_{in}(L,i,p(s_j),p(p(s_j)))\pi(p(s_j),s_j) &
\end{cases}$$

$$\eta_{out}(L,i,s_j,p(s_j))=$$

$$\begin{cases}
a(s_j,e(p(s_j))) & \text{if } i{=}|L| \text{ and } p(s_j){\in}S^{(2)} \\
\sum_{s_t\in D(p(s_j))} a(s_j,s_t)\beta(L,i{+}1,|L|,s_t,p(s_j)) & \text{if } i{<}|L| \text{ and } p(s_j){\in}S^{(2)} \\
a(s_j,e(p(s_j)))\eta_{out}(L,i,p(s_j),p(p(s_j))) & \text{if } i{=}|L| \text{ and } p(s_j){\notin}S^{(2)} \\
\sum_{l=i+1}^{\min\{|L|,i+d^{\mathcal{F}}_{max}\}} \Big[\sum_{s_t\in D(p(s_j))} a(s_j,s_t)\beta(L,i{+}1,l,s_t,p(s_j))\Big]\eta_{out}(L,l,p(s_j),p(p(s_j))) & \text{if } i{<}|L| \text{ and } p(s_j){\notin}S^{(2)} \\
\qquad +a(s_j,e(p(s_j)))\eta_{out}(L,i,p(s_j),p(p(s_j))) &
\end{cases}$$

$$\xi(L,i,s_j,s_k,p(s_j))=$$

$$\begin{cases}
0 & \text{if } i{=}|L|,s_k{\neq}e(p(s_j)),\text{and } p(s_j){\in}S^{(2)} \\
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\alpha(L,1,|L|,s_j,p(s_j))a(s_j,e(p(s_j))) & \text{if } i{=}|L|,s_k{=}e(p(s_j)),\text{and } p(s_j){\in}S^{(2)} \\
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\alpha(L,1,i,s_j,p(s_j))a(s_j,s_k)\beta(L,i{+}1,|L|,s_k,p(s_j)) & \text{if } i{<}|L|,s_k{\neq}e(p(s_j)),\text{and } p(s_j){\in}S^{(2)} \\
0 & \text{if } i{<}|L|,s_k{=}e(p(s_j)),\text{and } p(s_j){\in}S^{(2)} \\
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\Big[\sum_{l=\max\{1,i+1-d^{\mathcal{F}}_{max}\}}^{i} \eta_{in}(L,l,p(s_j),p(p(s_j)))\alpha(L,l,i,s_j,p(s_j))\Big] & \text{if } i{<}|L|,s_k{\neq}e(p(s_j)),\text{and } p(s_j){\notin}S^{(2)} \\
\qquad a(s_j,s_k)\Big[\sum_{u=i+1}^{\min\{|L|,i+d^{\mathcal{F}}_{max}\}} \beta(L,i{+}1,u,s_k,p(s_j))\eta_{out}(L,u,p(s_j),p(p(s_j)))\Big] & \\
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\Big[\sum_{l=\max\{1,i+1-d^{\mathcal{F}}_{max}\}}^{i} \eta_{in}(L,l,p(s_j),p(p(s_j)))\alpha(L,l,i,s_j,p(s_j))\Big] & \text{if } i{<}|L|,s_k{=}e(p(s_j)),\text{and } p(s_j){\notin}S^{(2)} \\
\qquad a(s_j,e(p(s_j)))\eta_{out}(L,i,p(s_j),p(p(s_j))) &
\end{cases}$$

$$\chi(L,i,s_j,p(s_j))=$$

$$\begin{cases}
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\pi(p(s_j),s_j)\beta(L,1,|L|,s_j,p(s_j)) & \text{if } i{=}1 \text{ and } p(s_j){\in}S^{(2)} \\
0 & \text{if } i{>}1 \text{ and } p(s_j){\in}S^{(2)} \\
\frac{1}{\mathbb{P}\{L|\phi(L)\}}\eta_{in}(L,i,p(s_j),p(p(s_j))\pi(p(s_j),s_j) & \text{if } p(s_j){\notin}S^{(2)} \\
\qquad\Big[\sum_{u=i}^{\min\{i+d-1,|L|\}} \beta(L,i,u,s_j,p(s_j))\eta_{out}(u,p(s_j),p(p(s_j)))\Big] &
\end{cases}$$

Table 2: Computation of η_{in}, η_{out}, ξ, and χ variables

$$\begin{aligned}
\hat{\pi}(p(s_l),s_l) &= \begin{cases}
\dfrac{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\Big[1\{\phi(L_k^{(j)}){=}p(s_l)\}\chi(L_k^{(j)},1,s_l,p(s_l))\Big]}{\sum_{j=1}^{|Q|}\sum_{k=1}^{t_j} 1\{\phi(L_k^{(j)}){=}p(s_l)\}} & \text{if } p(s_l)\in S^{(2)} \\
\dfrac{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\Big[1\{\phi(L_k^{(j)}){=}p(s_l)\}\sum_{i=1}^{|L_k^{(j)}|}\chi(L_k^{(j)},i,s_l,p(s_l))\Big]}{\sum_{j=1}^{|Q|}\sum_{k=1}^{t_j}\Big[1\{\phi(L_k^{(j)}){=}p(s_l)\}\sum_{s_r\in D(p(s_l))}\sum_{i=1}^{|L_k^{(j)}|}\chi(L_k^{(j)},i,s_r,p(s_l))\Big]} & \text{if } p(s_l)\neq s^{(1)}\wedge p(s_l)\notin S^{(2)}
\end{cases} \\
\hat{a}(s_l,s_t) &= \frac{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\sum_{i=1}^{|L_k^{(j)}|}\xi(L_k^{(j)},i,s_l,s_t,p(s_l))}{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\sum_{s_r\in D(p(s_l))}\sum_{i=1}^{|L_k^{(j)}|}\xi(L_k^{(j)},i,s_l,s_r,p(s_l))} \\
\hat{b}_{s_l}(v) &= \frac{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\Big[\sum_{i=1}^{|L_k^{(j)}|}1\{L_k^{(j)}[i]{=}v\}\chi(L_k^{(j)},1,s_l,p(s_l))+\sum_{s_r\in D(p(s_l))}\sum_{i=2}^{|L_k^{(j)}|}1\{L_k^{(j)}[i]{=}v\}\xi(L_k^{(j)},i{-}1,s_t,s_l,p(s_l))\Big]}{\sum_{j=1}^{|Q|}\sum_{k=1}^{|H_j|}\Big[\sum_{i=1}^{|L_k^{(j)}|}\chi(L_k^{(j)},1,s_l,p(s_l))+\sum_{s_r\in D(p(s_l))}\sum_{i=2}^{|L_k^{(j)}|}\xi(L_k^{(j)},i{-}1,s_t,s_l,p(s_l))\Big]}
\end{aligned}$$

Table 3: Parameter reestimation

$p(s_k)$, gives the probability that state s_j is finished on observation $L[i]$ and s_k is started on observation $L[i+1]$ under observation sequence L and model λ; $\chi(L, i, s_j, p(s_j))$ gives the probability that s_j is started on observation $L[i]$ under model λ and observation sequence L. Their computation is given in Table 2. The time complexity of computing η_{in} and η_{out} variables is $O(N \cdot M \cdot T \cdot d^{\mathcal{F}}_{max})$. Recall that T is the number of symbols in the observation sequence, N is the number of states in the model, and M is the maximum number of children of any node in the model. The time complexities of computing ξ-variables and χ-variables are $O(N \cdot M \cdot T \cdot d^{\mathcal{F}}_{max})$ and $O(N \cdot T \cdot d^{\mathcal{F}}_{max})$, respectively.

Appendix B: Parameter reestimation

In each iteration of Algorithm 4, the parameters in HHMM λ that are not at level 2 (i.e., $p(s_l) \neq s^{(1)}$) are updated as shown in Table 3.