Detecting Mobility Context over Smartphones using Typing and Smartphone Engagement Patterns

Soumyajit Chatterjee[*], Adrija Bhowmik[†], Arun Singh[‡], Surjya Ghosh[§], Bivas Mitra[¶], Sandip Chakraborty[‖]
[*‡§¶‖]Department of CSE, IIT Kharagpur, India, [§]Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[†]Department of CSE, NIT Durgapur, India
Email: {[*]soumyachat,[‡]singh.arun}@iitkgp.ac.in, [†]ab.16u10388@btech.nitdgp.ac.in
{[§]surjya.ghosh,[¶]bivasmitra, [‖]sandipchkraborty}@gmail.com

***Abstract*—Most of the latest context-based applications capture the mobility of a user using Inertial Measurement Unit (IMU) sensors like accelerometer and gyroscope which do not need explicit user-permission for application access. Although these sensors provide highly accurate mobility context information, existing studies have shown that they can lead to undesirable leakage of location information. To evade this breach of location privacy, many of the state-of-the-art studies suggest to impose stringent restrictions over the usage of IMU sensors. However, in this paper, we show that typing and smartphone engagement patterns can act as an alternative modality to sniff the mobility context of a user, even if the IMU sensors are not sampled at all. We develop an adversarial framework, named *ConType*, which exploits the signatures exposed by typing and smartphone engagement patterns to track the mobility of a user. Rigorous experiments with in-the-wild dataset show that *ConType* can track the mobility contexts with an average micro-F$_1$ of 0.87 (± 0.09), without using IMU data. Through additional experiments, we also show that *ConType* can track mobility stealthily with very low power and resource footprints, thus further aggravating the risk.**

***Index Terms*—Smartphone; Typing; Mobility**

I. Introduction

The majority of the current commercially available smartphone operating systems allow hassle-free access to inertial measurement unit (IMU) sensors like accelerometer, gyroscope, etc., which are primarily used for activity detection of users. Subsequently, the applications that log IMU information does not ask the user for explicit permissions to access these sensors [1]. This allows a majority of the state-of-the-art mobility context detection techniques to leverage on the high-frequency continuous sampling of these IMU sensors for application usage [2], [3]. Although such uncontrolled access of IMU sensors has benefited us with seamless services from several context-dependent and location-sensitive applications, however, it has also led us to the more significant concerns about the *location privacy* of a user [1], [4].

Location privacy is usually defined as the ability of a user to control the access of third parties to her (his) location information[1]. With smartphone usage becoming more and more prevalent, several independent research works have started looking into the potential sources that can breach the location privacy of a user [5]. Most of these research works point at one common source of such leakage – the IMU sensors [6], [7]. Although, IMU sensors are explicitly used to detect the mobility context of a user accurately, however, accurate mobility information coupled with spatio-temporal information can disclose the approximate location of a user at different times of the day [4]. Some of the recent papers have also revealed that such a privacy breach is possible even if the specific location services like GPS information is turned-off [8]. This further raises the concern, as unlike GPS, applications capturing IMU sensor logs do not need explicit permissions from the user [1]. Indeed, a standard solution, in this context, would be to impose restrictions on IMU usage. However, smartphones being a device which receives inputs from several modalities like camera, microphone, and screen interactions like typing, swyping etc., can have other alternate sources of information which, if appropriately exploited, may reveal the mobility context details of a user. This mobility information can then be augmented with existing techniques to sniff a user's location.

Thus, in this paper, we explore the possibility of identifying an alternative modality that can provide continuous monitoring of user mobility, even if the IMU sensors are not sampled at all. In this context, we find *screen-interactions* like typing to be a possible source of such information. Notably, the keystroke dynamics during typing can be a good indicator of the mobility context of a user; for instance, while walking or commuting in a crowded bus, people are likely to get distracted frequently, hence prefer to type short messages quickly (or slowly, depending on the individual) or tend to commit frequent mistakes [9]. Hence, smartphone typing may act as a promising indicator to capture mobility traits. Although typing on a smartphone may not be possible during all possible activities, nevertheless, a user may type while walking or traveling in a vehicle, apart from the static condition.

With several smartphone applications interacting with users through typing, a user typically spend quite a significant amount of time interacting with her smartphone through screen interactions (at least an hour everyday [10], [11]). This observation also opens up the vulnerability posed by keystroke dynamics for sniffing the mobility context *during typing*. However, a user cannot type during activities like running or cycling, so an adversary (say, a malicious application) can use typing patterns to detect the mobility contexts *only* when it is

[1]https://www.igi-global.com/dictionary/privacy-trust-management-schemes-wireless/17408 Access: February 23, 2020

possible. Considering the set of activities available under the Google Activity Recognition API (static, walking, running, in-vehicle, on bicycle, tilting, unknown) [12], we observe that typing may be feasible under three activities – static, walking and in-vehicle; so we may consider the keystroke dynamics to track these three activities. This approach essentially shows that even if someone completely restricts IMU sampling, then also one can glean information about mobility contexts from their typing and smartphone engagement patterns.

We further explore this vulnerability posed by the screen-interactions by developing an adversarial framework named *ConType*, which can track the mobility of a user ('static,' 'walking' and 'in-vehicle') using typing and smartphone engagement patterns. Notably, we do not capture any already-explored privacy sensitive information like the actual key-logging events (except a few special keystrokes like the backspace, space bar, etc.); instead, we extract keystroke features like tap duration, inter-tap duration, keystroke pressure, and velocity, etc. We extract a set of interesting features from these data, which help us in classifying the above three mobility contexts. At the outset, *ConType* uses a machine learning model to track the three mobility contexts from these features.

We have implemented *ConType* as an Android application (Android 8.0 Oreo or higher) which comprises of a custom-made keyboard (for keystroke feature extraction) along with the *ConType* decision engine. The prototype has been thoroughly tested in-the-wild with 25 users from approximately 3-million valid typing instances. Considering the mobility contexts returned by Google Activity Recognition API as the baseline, we observe that *ConType* can sniff various mobility contexts with an micro-F_1 score of ≈ 0.87 (± 0.09), even if the IMU sensors are completely blocked from the usage. Additionally, we observe that *ConType* also has a very low energy footprint; this shows that information can be leaked without much user attention, such as huge battery drainage or resource blockage.

II. Related Work

Mobility context detection is not a new topic in the research community; over the past decade, there have been a plethora of research works in this field [13], [14]. Most of these works exploits IMU sensors like accelerometer and gyroscope or the traces from other sensors like GPS and acoustic for fine grained mobility context (say, vehicle type) detection [15], [16]. However, despite all these successes in detecting the mobility just by using IMU sensors and GPS information, there has been a consistent criticism of these sensors for leaking out private information regarding the location of users [7]. Most of these works concentrate on detecting the mobility first and then subsequently use the mobility along with the spatio-temporal information to predict the location of the user [4], [8]. Furthermore, these works have also shown that even with location services (or GPS) explicitly turned off, these schemes allow the smartphone applications (malicious or not) to detect the location of the user with an appreciable accuracy [1], [8].

Subsequently, several research works have come up with proper definitions and threat models that deal with such privacy breaches involving leakage of location information [5]–[7]. However, a common approach suggested by most of these works is in devising policies that restrict the usage of IMU sensors. Understanding these limitations, we search for an alternative modality which can be still provide the mobility information even though IMU sensors are not used.

On the other hand, although typing patterns have been used to detect soft-biometric traits, to the best of our knowledge, we have found no related literature in this field except [17] which analyses the probability of usage of a particular group of applications on smartphones in different physical activity contexts like stationary, walking, and in-vehicle. This work concludes that the likelihood of usage of specific groups of applications differ across various physical activity contexts and also summarizes the major application types used in different physical activity sessions. For example, they have found that the usage of social apps is more when the users are stationary whereas the communication apps (like the phone call, VoIP apps, etc.) are used more often when the users are in motion (both walking or in-vehicle).

III. Data Collection Framework

We first design an experimental apparatus for collecting keystroke data from smartphone users in-the-wild. Although the majority of the users use the default Android keyboard like the Gboard app developed by Google, it does not have the support for keystroke feature logging. Further, to validate the performance of *ConType*, we also need the ground truth context data. Therefore, we developed a custom keyboard for *ConType* (shown in Figure 1), which contains (i) a *Keyboard Logger*: a QWERTY keyboard to collect application usage and keystroke interaction data and (ii) the *Ground Truth Collection Unit*: an integrated ground truth context collection system. Also, the app uploads the collected data automatically on the server for further processing. In the following, we illustrate the details of each component of the developed app.

A. Keyboard Logger

To capture the smartphone engagement data of different users, we design a software keyboard, shown in Figure 2, for smartphones based on Android *Input Method Editor* (IME) facility. The keyboard provided by this application is similar in functionality to the general QWERTY keyboard, provided by default. In addition to this standard functionality, the keyboard has additional capabilities of capturing user's keystroke interaction patterns during typing. As key-logging has a significant privacy concern, the app does not log any content and only captures the keystroke interaction data illustrated in Table I.

B. Ground Truth Collection Unit

This component of the developed Android application automatically records the ground-truth mobility contexts leveraging on the *Google Activity Recognition* API[2]. This API returns

[2]Any activity recognition model can be fitted in place of this API. There is no strict dependence on this API.

TABLE I: Raw data captured using the keyboard app. Here t represents the time instance of a keypress event.

Factors	Notation	Significance
Key-press Event Time-stamp	s_t	The time-stamp of the key-press event.
Application	a_t	The name of the application with which the user is interacting through typing at time instance t.
Special Key-codes	k_t	The key-codes of some specific keys like 'backspace', 'space bar', and some special characters like '@,' '*' and '\^.'
Pressure	p_t	The amount of pressure exerted by the user on the smartphone touchscreen during typing.
Velocity	v_t	The velocity of typing at time instance t in $pixels/sec$.
Tap Duration	d_t	The amount of time a particular key is kept pressed while typing.
Inter-Tap Duration	r_t	This denotes the amount of time between two subsequent key-press events. This is calculated from key-press event time-stamp.

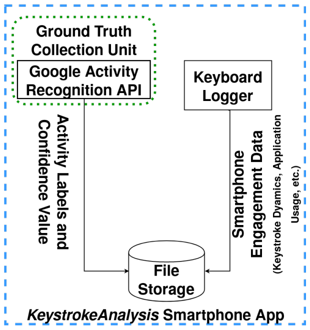
Ground Truth Collection Unit
Google Activity Recognition API
Keyboard Logger
Activity Labels and Confidence Value
Smartphone Engagement Data
(Keystroke Dyamics, Application Usage, etc.)
File Storage
KeystrokeAnalysis Smartphone App

Fig. 1: Data Collection Unit

Fig. 2: *ConType* Keyboard

eight physical activity labels with an associated *confidence value* to mark the likelihood of occurrence of the specified physical activity. The eight physical activity labels returned by the API include - (a) Still (Static), (b) Walking, (c) Running, (d) On Foot, (e) On Bicycle, (f) In-Vehicle, (g) Tilting, and (h) 'Unknown.' We perform an initial noise removal by logging only those activity labels which have the associated confidence value ≥ 70. As mentioned earlier, typically users can type over a smartphone during three among the above eight activities – static, walking and in-vehicle; so *ConType* uses an opportunistic approach to derive these three activities from the keystroke interaction patterns.

C. Privacy and Trust Concerns during Keystroke Logging

As keylogging has a serious privacy concern, we would like to highlight that *ConType* does not log any alphanumeric keys, and the logging is restricted to the keystroke interaction patterns only, specifically the features enumerated in Table I. Also, the logging application does not export the logged raw data to any other application for further analysis. The communication to the *ConType* server is done via a secure channel over the Internet. The users have also been informed about the data being collected by the *ConType* keylogger unit. Precisely, we have followed the well-accepted standards of privacy non-intrusive logging strategy mentioned in [11] to prevent any privacy violation during the deployments.

IV. Pilot Study

In this section, we show the results from a thorough pilot study to find out the opportunities and challenges for extracting out the mobility context of a user from her typing patterns.

A. Typing Patterns for Tracking Mobility Context

To explore the vulnerabilities exposed by the keystroke dynamics as a signature for mobility context detection, we recruited 13 volunteers (primarily college students who use different modes of transports daily during their travel from home to college and back) for the pilot study. The volunteers belong to the age group 18-35 (8-males and 5-females) which introduce sufficient diversity in the collected data [18]. We ask the volunteers to type texts over different Android apps which they regularly use and provide data[3] in different mobility contexts with one and two-handed modes of typing. All these participants were asked to provide typing data without any constraints, and hence, no typing or time constraints were imposed on them.

B. Opportunities

The main intuition behind exploiting keystroke interaction patterns as a potential source for tracking mobility context information is the fact that the typing behavior of humans changes with the change in their mobility. Recent studies show that keystroke interaction patterns can detect the soft-biometric characteristics, such as age, handedness, etc. [18], [19] of a user. We leverage on those studies to explore the potential of inter-tap duration (ITD) to detect mobility contexts across different users. Here, the ITD is measured by the difference of time between the subsequent keystrokes made by the user on the designed keyboard app.

We compute the distribution of ITD of the users, participating in the pilot study, under the three targeted mobility contexts – static, walking and in-vehicle. Fig. 3 shows the distribution of inter-tap duration of two representative users in a diverse mobility context. We observe a considerable amount of distinction in the distribution of inter-tap duration for a user concerning the various mobility context. This gives us an indication that such features can be used to differentiate the mobility contexts of a user.

C. Challenges

Albeit promising, typing patterns exhibit challenges in extracting the mobility context. We observe that even for the same user, the typing patterns vary significantly. Precisely, for a typical individual, the typing interval varies between 80-500 ms [20], which eventually results in the variation. As shown

[3]Prior consents have been taken following ethical norms of data collection.

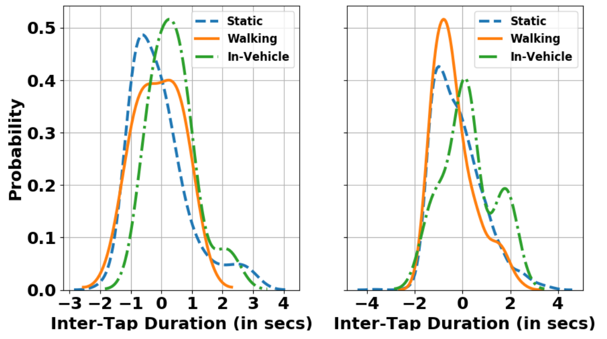

Fig. 3: Distribution of inter-tap duration (in log scale) in different mobility contexts for **2** different users.

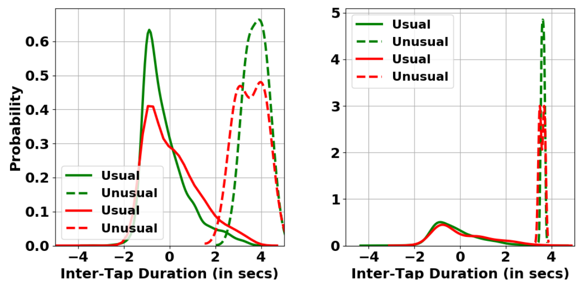

Fig. 4: Variation in the inter-tap duration (in log scale) for the 2 different users (represented by two different colors) in same mobility context – Static (left) and Walking (right).

in Figure 4, even for the same users, we see that the data forms separate clusters based on a simple feature like ITD, and the distribution of the feature across these two clusters are profoundly different albeit the underlying mobility context is same. From further investigation, we could understand that these variations are because of different modes of typing with normal and unusual modes of typing instances present for the same user even in the same mobility context. For instance, a user may generally type using both the hands simultaneously; however, under some unusual circumstances like while traveling in a crowded metro, she sometimes uses one hand for typing and the other hand for support. In Fig. 4, we mark the cluster as 'usual' for which we get a maximum number of samples and the second cluster as 'unusual' with lesser number of samples indicating rare behavior.

Based on these observations, we develop the adversarial framework, named *ConType*, which uses the typing and smartphone engagement patterns to infer the mobility context of a user without sampling IMU data. The details are as follows.

V. Development of *ConType*

ConType uses an opportunistic approach to glean smartphone app engagement and typing patterns to extract the mobility context of a user whenever possible. As we have mentioned earlier, we consider the three mobility levels when a typing engagement is most feasible. The keystroke and app engagement pattern analysis model of *ConType* consists of two major components – the client (smartphone) and the server-side (see Figure 5). The client-side of *ConType* primarily contains the data collection apparatus, as described in Sec. III, whose primary objective is to gather the data, pre-process, and then upload it to the server for processing. On the other hand, the server-side of *ConType* detects the three mobility contexts by using the smartphone engagement data (as described in Table I) for predicting the mobility context. The details follow.

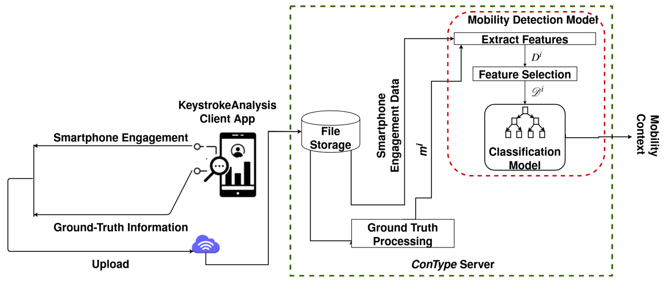

Fig. 5: *ConType* framework

A. Developing the ConType Server-Side

The typing and app engagement data collected through *ConType* keyboard is analyzed at the server for the inference of the mobility context of the user. It can be noted that the popular sensor-based context detection services, like the Google Activity Recognition API, also use a server-side module for processing the collected data [21]. The server builds up a model for activity classification, and as the data arrives from the clients, it returns one of the three activity levels – static, walking and in-vehicle, entirely based on the smartphone engagement patterns. The step-wise procedure for training and inference of the activity levels from the typing and app engagement data are as follows.

B. Sessionizing Typing Data

For feature extraction from the typing data (as summarized in Table I), we first need to sessionize the raw data [10]. The raw typing data which contains keystroke interaction patterns like pressure (p_t), velocity (v_t), tap (d_t) and inter-tap duration (r_t), needs to be aggregated to form blocks so that appropriate features can be extracted. We identify that choosing a proper session length is challenging. A small session length is still prone to noise whereas a large session length reduces the dataset size exponentially. In order to circumvent, we introduce *Activity Session Length* (ASL), which demarcates the typing data with the same ground-truth context labels obtained from the *Google Activity Recognition* API. For a user i, we denote the j^{th} activity session by N^i_j with the corresponding session length n^i_j. However, as ASL can vary randomly across different time intervals, we choose a more principled session length to sessionize the typing data, by computing the minimum ASL (m^i) observed from the collected data in the training phase as m^i=min$\{n^i_1, n^i_2, \ldots, n^i_{j-1}, n^i_j\}$. Here the session length m^i

indicates the smallest stable time window in which the user i stays in a specific mobility context. We then use this computed minimum ASL to sessionize the data, obtained for the user i, during runtime.

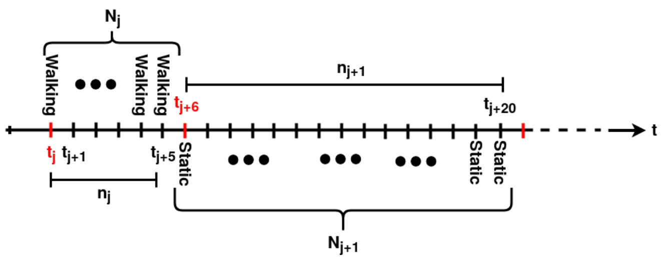

Fig. 6: Sessionizing ground truth

Thus, the session length m_i for a user i is used to form blocks B_t^i in the typing data, with t as the starting instance of the typing event. Therefore, a block of sessionized typing data for the i^{th} user can be represented as $B_t^i = \{E_t^i, E_{t+1}^i, \ldots, E_{t+m^i-1}^i\}$, where $E_t = \{a_t, k_t, p_t, v_t, d_t, r_t\}$ is the tuple containing the 7 keystroke interaction data.

C. Feature Extraction.

From each of the blocks B_t^i with session length m_i, starting at time t, we construct the following features.

1) Application Usage (D_0): This feature is obtained by taking the application (a_t) that most frequently appeared within a block B_t^i. The main intuition behind constructing this feature is to capture the distribution of application usage over different mobility contexts.

2) Keystroke Interaction Statistics (D_1-D_{20}): These features are computed by applying statistical operations like mean, standard deviation, median, min and max on the keystroke interaction data, such as pressure(p_t), velocity (v_t), tap (d_t) and inter-tap duration (r_t) within a block B_t^i. These features capture the change in the typing interactions across different mobility contexts.

3) Word-based Features (D_{21}-D_{25}): This feature is intended to capture the general human tendency to commit errors & type shorter messages while in motion. For computing these features, we leverage on the keycodes (k_t) and use them to obtain the error counts and word typing velocity in a session. We first count the number of backspaces (D_{24}) and special characters (D_{25}) like '*' and ';' to count errors. Next, we count the number of blank spaces in the typing session, and subsequently compute the number of words typed (D_{21}) in a session. Finally, we calculate the maximum word length (D_{23}) in a block, and obtain the word typing velocity (D_{22}) as the ratio of number of words (D_{21}) and session length m_i.

4) Typing Mode-based Feature (D_{26}): From the pilot study (Sec. IV-B), we realize that one of the primary sources of variation in the keystroke interaction is due to the difference between normal (say, typing in one hand) and the unusual (say, two-handed modes of typing) mode of typing [22]. Importantly, the typing mode as a feature (say, handedness) can contribute to the ease of typing, which may indicate the mobility context of the user. However, obtaining the exact mode of typing (one hand or two hand) in an in-the-wild collected dataset is difficult; therefore, it becomes nearly impossible to develop a supervised model for identifying these rare typing instances. Thus, we develop an unsupervised model for classifying the *normal* and *unusual* modes of typing. Inspired by the recent literature [22], we rely on the two primary indicators which can act as prominent discriminators of typing mode (handedness) – (a) tap duration (d_t) and (b) inter-tap duration (r_t). We perform a k-means clustering on the un-sessionized typing data (with $k = 2$) based on the two features described above. Once the clusters are obtained, we perform *Mann-Whitney Test* [23] to check for the significance of the clusters. If the clusters are significant (p-value < 0.05), we classify the individual typing instances in two different clusters; the smaller cluster is characterized as the unusual mode of typing for that particular user. However, if the clusters are not significant, we consider all the data points as a single cluster. Finally, we compute the feature ζ_t (binary variable) in a block B_t^i, which indicates the typing instance at t as normal or unusual based on its presence in the identified clusters.

5) Temporal Information (D_{27}): This feature captures the temporal behavior of continuing in a particular mobility context for a specific user. We model the transition of mobility context of an user from one session to the next as a *Markov* model. The state space of this Markov model contains the mobility context classes – 'static,' 'walking' and 'in-vehicle'. The state transition probability from one state to another indicates the probability to move from one mobility class to another within a typing session. The process to learn the state transition probabilities, which is represented as a 3×3 matrix P, is described as follows. We first calculate the total number of mobility context transitions over the entire sessionized typing data, let this be denoted by η. Now, the probability of a transition from state e to f can be denoted by $P_{ef} = \frac{\eta_{ef}}{\eta}$. Here, $e,f \in \{$static, walking, in-vehicle$\}$ and η_{ef} is the total number of times there is a transition from e to f in the sessionized data. Finally, we compute the feature by considering the maximum transition probability obtained from the product of the activity label for the previous typing session, represented in the form of a one-hot vector, and the transition matrix.

D. Prediction of the Mobility Context

ConType typically uses a Random-Forest based learning model to predict the mobility context of a user. However, as the signatures for mobility contexts are sometime dependent on factors like age, gender, etc., we can use a personalized feature selection to boost up the performance of *ConType*. For this, we need ground-truth from the IMU data; therefore, when the IMU data is available for some time for some mobility contexts (for example, scenarios like when IMU was available before the user turned it off), we choose the top-5 features using a personalized feature selection algorithm [4]. The feature

[4]https://scikit-learn.org/stable/modules/feature_selection.html Last Accessed: February 23, 2020

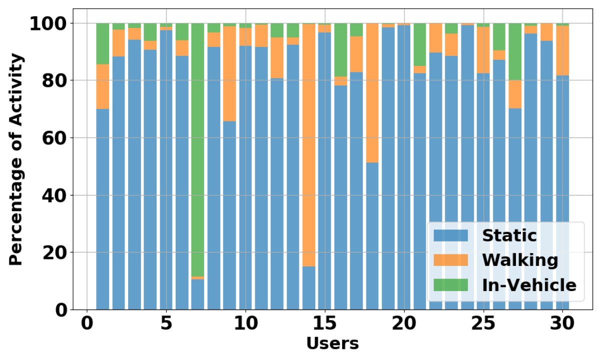

Fig. 7: Ground-truth mobility classes

selection algorithm essentially uses the keyboard interaction data and associated ground-truth context labels, collected during the training phase, to rank the features according to their importance using an Ensemble classifier [24] (total number of estimators = 250). Finally, once the features are selected judiciously for a user, in the training phase, the selected features are used to develop a personalized learning model (based on Random-Forest or Support Vector Machine), which is then used to predict the mobility during runtime.

VI. Performance Evaluation

Since the problem of predicting mobility contexts is inherently a multi-class problem, we use the well-adopted metric micro-F_1 [25] to evaluate the accuracy of context detection by the *ConType* framework. We first discuss the detailed experimental setup and then analyze the performance of *ConType* from various angles.

A. Data Collection & Preparation

We start by describing the data collection drive and preparing the dataset for evaluation of *ConType*.

1) In-the-Wild Study: We extend the data collection to gather in-the-wild data by recruiting volunteers from different parts of the country, mainly comprising of students who daily commute to their universities or academic institutions. We asked the volunteers to use the *ConType* app continuously for the entire duration of 2 months without imposing any other constraints. To encourage participation in the data collection drive, we have given a token incentive of US$ $\approx$ 42 to each participant. Although initially, 53 volunteers turned up for the data collection drive, many of them left the drive within the first 2 weeks, on reasons like usage of a different keyboard.

Finally, the collection drive had 30 participants who contributed data for the entire period, with a mean duration of ground-truth contexts across all users of 984.57 hours. Moreover, we observe that the participants in this experiment used a total of 392 different applications.

2) Typing Data Preparation: To sessionize the typing data, we use the session length of m_i for creating blocks for user i. Since session length m_i is obtained from the automated ground-truth labels received from the *Google Activity Recognition* API, there can be many labels which are spurious and do not account for the change in the stable mobility context. For example, say, the user was sitting in a chair while typing, then the user may walk for a while and take the next chair to sit again. Such an activity change for a short duration can not be captured through *ConType* as the user has not typed anything significant while walking. To remove such spurious labels, we consider only those ground-truth activity sessions which have a length $\geq$ 60 secs. Additionally, we compute the distribution of ground-truth activity labels across different users. As shown in Figure 7, it can be observed that almost all the users, except one, have a high number of typing samples under the 'Static' mobility class. Besides, we also observe that 73.33% of users have all the 3 mobility classes present in their data. Thus, to evaluate the system in a principled way, we first divide the entire dataset for each user into two disjoint training and test dataset. For this purpose, we use a split ratio of 70 to 30. Once the dataset is split into the training and the testing dataset, we balance the training dataset using *Synthetic Minority Over-Sampling Technique* (SMOTE) [26]. We then use the balanced data for feature selection and training the model. The trained model is then evaluated on the held-out test dataset.

During the evaluation of the system, we found that out of the 30 users only 25 users (with a total of $\approx$ 3 million valid typing instances) had sufficient distribution of data in all the mobility classes to train and test the models following the approach mentioned above. Hence for the rest of the experiments, we consider data from these users only.

B. Context Detection Accuracy

We first analyze the centralized model for *ConType*, where each user's data is tested on the model (random forest-based) trained with the data of all other users other than him (her). From the evaluation results reported in Figure 8, we observe that *ConType* provides a median micro-F_1 score of $\approx$ 0.81. However, it can be noted that the centralized model has a high standard deviation as for some users, the centralized framework fails to capture the personalized typing behavior. To evaluate the personalized feature selection model, we utilize the top 5 selected features from the typing and smartphone engagement engagement data, along with the ground-truth labels obtained from the automated ground truth collection unit. We choose to compare between *Random Forest* (RF) (with *number of estimators* = 10) and *Support Vector Machine* (SVM) (with *Random Bias Function* kernel) for the implementation of the model. We observe that *ConType* with RF-based model performs the best (has a higher median) and has an average micro-F_1 score $\approx$ 0.87 and low standard deviation of $\pm$ 0.09 across all users

We next analyze the class-wise feature importance across all the users. We represent a subset of those features in Fig. 9 which appear for at least 3 users across all the three mobility classes. From the figure, we can see that temporal information appears as an essential feature for almost all the

TABLE II: Analysis of percentage (%) CPU and memory footprint of *ConType*

Devices		**IMU**									***ConType***
		Static			**Walking**			**In-Vehicle**			
		0.1 sec	**15 secs**	**60 secs**	**0.1 sec**	**15 secs**	**60 secs**	**0.1 sec**	**15 secs**	**60 secs**	
Redmi 5	CPU	5.91 (±10.19)	2.96 (±2.62)	1.48 (±1.24)	22.92 (±17.50)	6.45 (±4.54)	1.95 (±0.44)	32.74 (±3.76)	10.35 (±10.67)	3.92 (±0.59)	**1.75** (±0.96)
	Mem	4.68 (±0.14)	4.75 (±0.13)	4.36 (±0.12)	3.62 (±0.14)	3.62 (±0.27)	4.21 (±0.18)	4.15 (±0.16)	4.38 (±0.13)	4.34 (±0.16)	**3.13** (±0.29)
Motorola Moto G5	CPU	23.95 (±5.28)	3.57 (±0.30)	3.33 (±0.70)	29.98 (±5.80)	8.86 (±6.27)	7.29 (±3.67)	27.75 (±5.84)	4.81 (±0.74)	3.75 (±0.82)	**2.07** (±0.42)
	Mem	5.55 (±0.02)	4.55 (±0.03)	4.62 (±0.07)	4.88 (±0.07)	4.61 (±0.10)	4.83 (±0.05)	4.88 (±0.13)	4.99 (±0.04)	5.03 (±0.02)	**3.14** (±0.11)

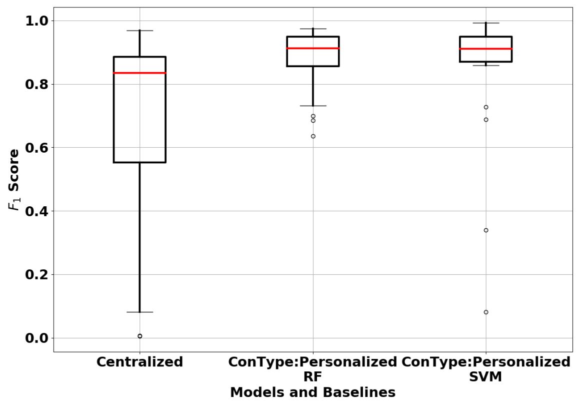

Fig. 8: Accuracy of *ConType*

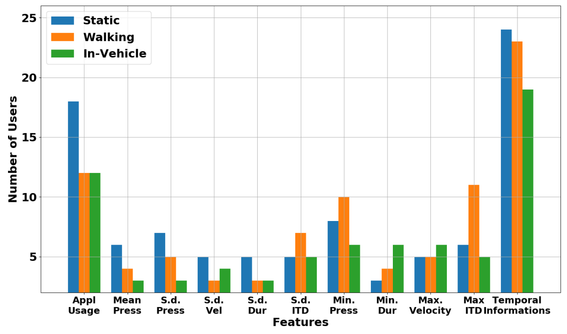

Fig. 9: Distribution of top-5 features

users across all the mobility conditions. Other vital features include application usage patterns and keystroke interaction statistics. Besides, the features like max duration, median pressure, and min ITD served as essential features, for a significant number of users, to discern the mobility classes static, walking, and in-vehicle, respectively.

C. Profiling ConType

We next profile *ConType* to see the resource footprint of the application. It can be noted that the risk of privacy breach gets aggravated if the proposed technique has less resource footprint, thus keeping the user unaware about the background analysis that is running in her device. We look the CPU, memory and the energy consumption footprint of *ConType*, as discussed next.

1) Resource Consumption: We subsequently measure the resource and energy footprints of *ConType* to analyze its traceability in terms of resource and power consumption while detecting the mobility context. We have profiled *ConType* by measuring the CPU and Memory consumption using Android debugging tools like `dumpsys` and `top` respectively. We evaluate the framework on two commercially available smartphones – Redmi 5 (price \$112) and Motorola Moto G5. As earlier, we have used the Android app *AndroSensor* [27] for continuous sensor polling at different polling frequencies.

From the results shown in Table II, we observe that for both the devices, *ConType* has a relatively low resource footprints in terms of both CPU and memory consumption. *ConType* uploads the data to the server in batches for analysis and mobility detection. In this process, the client app compresses the entire data into a zipped archive and uploads it to a remote server. We separately compute this overhead and observe that during this phase, the app consumes CPU in the range 6.5-11% depending on the network connectivity and the number of apps running in the foreground. Moreover, we also observe that while not in use, the client keyboard application does not consume any excess CPU or memory resources.

2) Power Consumption: We further analyze the energy footprint of *ConType*, while typing, using a High Voltage Power Monitor (HVPM) from Monsoon solutions[5] and compare it with the following two cases – (a) *idle*: the smartphone (Motorola Moto G5) is kept untouched without any screen activity or foreground app running but with WiFi connectivity, (b) *sensor based polling*: continuous polling of the IMU sensors during the three mobility contexts with varying update intervals of – 0.1 sec, 15 secs and 60 secs for each of the mobility contexts. We record the power consumption logs for each of the baselines for 2 minutes each.

We next perform a pairwise *Wilcoxon Rank-sum Test* [28] (with $\alpha = 0.01$) on the power consumption logs obtained for each of the above mentioned cases. From the corresponding box-plot shown in Fig. 10, we see that *ConType* consumes significantly (all the pairwise comparisons yield p-value $<$ 0.01) lesser energy footprint than all other baselines. Notably, the power consumption of *ConType* is just 1.5% more than

[5]https://www.msoon.com/high-voltage-power-monitor (Access:February 23, 2020)

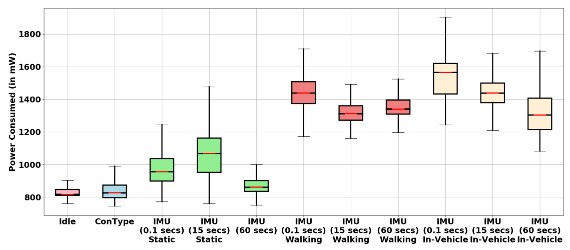

Fig. 10: Power consumption of *ConType*

the power consumption when the smartphone kept in idle state. This allows *ConType* to sniff mobility context of the user much more stealthily in comparison to the IMU sensors which have a high energy and resource footprint.

VII. Conclusion

As smartphones have become smarter, several context-dependent services have been developed over this platform. While a large number of smartphone users tend to use apps that run continuous mobility context recognition as a background service, information leakage concerning the location privacy of a user is a concerning issue. Majority of the methodologies developed over the last few years for evading such privacy breaches point to a single solution of restricting the IMU usage. This paper shows that there can be alternate modalities, like screen interactions (in the form of typing and smartphone engagements), that can be exploited to track mobility context even if the IMU-based sampling is completely avoided. The developed adversarial framework, *ConType*, gleans the smartphone engagement and typing patterns to track the mobility context of a user with no IMU sampling involved. Furthermore, the developed framework has a very low resource and energy footprints, thus make it more difficult to trace while tracking the mobility of a user.

Acknowledgment

This research was supported by TCS under the project titled "Behavior Modeling In Multi-sensor Environments - Integrating Environment Sensing, Human Sensing & Social Sensing for Rich Insights" (VN/BK/18-19/AUG/65). We would like to show our gratitude to Dr. Lama Nachman, Intel Corporation, for shepherding our paper and providing us several insightful comments on preparing the final accepted manuscript. We would also like extend our gratitude to LRN Foundation and COMSNETS Association for providing travel grants.

References

[1] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "Inferring user routes and locations using zero-permission mobile sensors," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 397–413.

[2] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio, "A smartphone-based fall detection system," *Pervasive and Mobile Computing*, pp. 883–899, 2012.

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones."

[4] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*. IEEE, 2012, pp. 1–9.

[5] L. Zhou, S. Du, H. Zhu, C. Chen, K. Ota, and M. Dong, "Location privacy in usage-based automotive insurance: Attacks and countermeasures," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 196–211, 2019.

[6] A. Gkoulalas-Divanis, Y. Saygin, and D. Pedreschi, "Privacy in mobility data mining," *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 1, pp. 4–5, 2011.

[7] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir, "The perils of user tracking using zero-permission mobile apps," *IEEE Security & Privacy*, vol. 15, no. 2, pp. 32–41, 2017.

[8] A. Mosenia, X. Dai, P. Mittal, and N. K. Jha, "Pinme: Tracking a smartphone user around the world," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 420–435, 2018.

[9] A. Exler, V. Diener, A. Schankin, and M. Beigl, "Detection of a smartphone user's distraction based on typing and touch gestures," in *PervasiveHealth*. ACM, 2019, pp. 242–250.

[10] S. Ghosh, N. Ganguly, B. Mitra, and P. De, "TapSense: combining self-report patterns and typing characteristics for smartphone based emotion detection," in *MobileHCI*, 2017.

[11] D. Buschek, B. Bisinger, and F. Alt, "Researchime: A mobile keyboard application for studying free typing behaviour in the wild," in *CHI*. ACM, 2018, p. 255.

[12] "Activity recognition api," https://developers.google.com/location-context/activity-recognition/, [Online; accessed 21-Sept-2019].

[13] Ö. Yürür, C. H. Liu, Z. Sheng, V. C. Leung, W. Moreno, and K. K. Leung, "Context-awareness for mobile sensing: A survey and future directions," *IEEE Commun Surv & Tut*, vol. 18, no. 1, pp. 68–93, 2014.

[14] Y. Kim, A. Ghorpade, F. Zhao, F. C. Pereira, P. C. Zegras, and M. Ben-Akiva, "Activity recognition for a smartphone and web-based human mobility sensing system," *IEEE Intell. Syst*, vol. 33, no. 4, pp. 5–23, 2018.

[15] Y. Zhou, B. P. L. Lau, C. Yuen, B. Tuncer, and E. Wilhelm, "Understanding urban human mobility through crowdsensed data," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 52–59, 2018.

[16] J. Wahlström, I. Skog, and P. Händel, "Smartphone-based vehicle telematics: A ten-year anniversary," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2802–2825, 2017.

[17] A. Mathur, L. M. Kalanadhabhatta, R. Majethia, and F. Kawsar, "Moving beyond market research: Demystifying smartphone user behavior in india," *ACM IMWUT*, p. 82, 2017.

[18] O. Miguel-Hurtado, S. V. Stevenage, C. Bevan, and R. Guest, "Predicting sex as a soft-biometrics from device interaction swipe gestures," *Pattern Recognition Letters*, pp. 44–51, 2016.

[19] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics for keystroke dynamics: Profiling individuals while typing passwords," *Computers & Security*, pp. 147–155, 2014.

[20] H. Gascon, S. Uellenbeck, C. Wolf, and K. Rieck, "Continuous authentication on mobile devices by analysis of typing motion behavior," in *Sicherheit*, 2014.

[21] R. Gouveia, E. Karapanos, and M. Hassenzahl, "How do we engage with activity trackers?: a longitudinal study of Habito," in *ACM Ubicomp*, 2015, pp. 1305–1316.

[22] S. Z. S. Idrus, E. Cherrier, C. Rosenberger, and P. Bours, "Soft biometrics for keystroke dynamics," in *ICIAR*, 2013, pp. 11–18.

[23] N. Nachar *et al.*, "The mann-whitney u: A test for assessing whether two independent samples come from the same distribution," *TQMP*, pp. 13–20, 2008.

[24] H. Deng and G. Runger, "Feature selection via regularized trees," in *IJCNN*, 2012, pp. 1–8.

[25] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, "Optimal thresholding of classifiers to maximize F1 measure," in *ECML PKDD*. Springer, 2014, pp. 225–239.

[26] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, pp. 321–357, 2002.

[27] "Androsensor," https://play.google.com/store/apps/details?id=com.fivasim.androsensor, [Online; accessed 21-Sept-2019].

[28] E. Whitley and J. Ball, "Statistics review 6: Nonparametric methods," *Critical Care*, p. 509, 2002.