Virtual Safe: Unauthorized Movement Detection for Mobile Devices

Dakun Shen, Ian Markwood, Dan Shen, Yao Liu
University of South Florida, Tampa, FL
dakun@mail, imarkwood@mail, danshen@, yliu21@mail.usf.edu

***Abstract*—The prevalence and monetary value of mobile devices, coupled with their compact and, indeed, mobile nature, lead to frequent theft due to a lack of proper anti-theft mechanisms. Currently there only exist damage control efforts such as remote wiping the device's memory or GPS tracking, but nothing to notify users of theft while it takes place. We propose such a mechanism which utilizes the unique motion patterns inherent to humans and differentiate our work from other motion behavior studies by using it as first-order authentication and developing matching methods fast enough to act as an actual anti-theft system. We test our system with the aid of 45 volunteers and demonstrate detection of unauthorized movement within 10 to 20 steps with an accuracy of 96.4% to 97.5%, while simultaneously distinguishing owners as themselves with 97.8% accuracy.**

I. Introduction

Evolutions in mobile technology have enabled mobile devices to become extensively personal and valuable items, their loss or theft furthermore compromising important sensitive information. Advocacy group Consumer Reports writes that 2014 sustained more than 3.1 million instances of mobile device theft, despite myriad anti-theft measures available for users [15]. These measures all focus on devices' retrieval or prevention of information leakage, instead of attempting to make a timely alert of the actual theft.

For example, included in Apple's iOS 7 is a popular scheme which allows users to remotely wipe and lock their devices, once a theft is discovered. A thief is unable, then, to read any of the deleted data. Apple also offers a free app called Find My iPhone which enables users to collect GPS information from their devices for the opportunity to find and recover them. These examples illustrate the reactive nature of current approaches and the void of proactive defenses. They require discovery of the theft before any security actions can be made, and therefore allow the thief full physical access of the device for a potentially notable amount of time. Security researchers understand this to be the most powerful type of adversary.

In this sense, there are no true anti-theft mechanisms for mobile devices, just damage control schemes. Accordingly, in this paper we develop a means to detect stealing behavior, that is, ongoing unauthorized movement of a device. Entitled Virtual Safe, we liken it to a physical safe for storing valuables, as it enables the user to set down a device and restricts other individuals from removing it without causing alarm. Because a thief has to walk away with a stolen mobile device, we employ motion pattern, or gait, authentication to verify the identity of the person in possession of the device immediately whenever it is moved. Human gait has been studied and used in several previous behavioral biometric schemes using various classification algorithms different from ours to authenticate users (e.g., [2]–[4]). Notably, these approaches all focus on empirically proving that human gait is an effective metric to identify human beings while none to our knowledge handle the time demand involved with our anti-theft application. Of all the related papers we could find, only one comments on the effect of the training time on system performance, and none comment on the time required for testing. For anti-theft, it is critical for the detection scheme to render a very timely decision on the identity of the current device holder.

We thus offer a detection system for performing authentication whenever the mobile device is moved, able to notify the owner before a thief escapes. Of the sensors offered in modern devices, we use the accelerometer, which monitors the device's acceleration due to human movement. This provides us occasion to begin authentication, as both a device owner and any potential thieves necessarily cause a reading on this sensor as soon as they interact with the device. Once this occurs, we compare the current user's acceleration data with that of the owner to give a match score, which if smaller than a threshold indicates unauthorized movement.

A naive strategy would be to directly apply existing classification tools such as the Kolmogorov-Smirnov (K-S) test [12], Total Variation Distance [1], or Support Vector Machine (SVM) [5] to the raw accelerometer data, but these tools involve non-trivial operations like empirical probability fitting and Quadratic Programming. Because a quick theft detection scheme needs a minimized number of complex operations, wherever possible we use less expensive calculations, like arithmetic addition and multiplication, that can be easily implemented and executed on a mobile device.

Additionally, a quick detection method should operate on a limited set of data to reduce the total number of comparisons necessary. For this reason we work to identify the most representative walking patterns for a user, those that are strongly correlated with the rest of the raw accelerometer data. Through construction of an algorithm which compares only the most representative walking patterns using elementary arithmetic, in lieu of using sophisticated classification tools to classify the whole raw data, we reduce the processing time to its minimum.

Specifically, we have created motion synchronization techniques to extract step cycles from the raw acceleration data for comparison between individuals. We have also created

a representative matching algorithm to find and compare "signature" step cycles for a behavior for improved accuracy and reduced processing time. Our case study shows that this matching method is theoretically 300 times faster than the traditional strategy of comparing all possible data. We find by real world experiment that the proposed system distinguishes between our 45 volunteers with 96.4% to 97.5% accuracy. Additionally, each particular participant in our experiment was static enough in walking habits to be identified as themselves 97.8% of the time. Unauthorized movement of mobile devices is detected within 10 to 20 steps, and battery usage overhead is around 4.7% for the typical user.

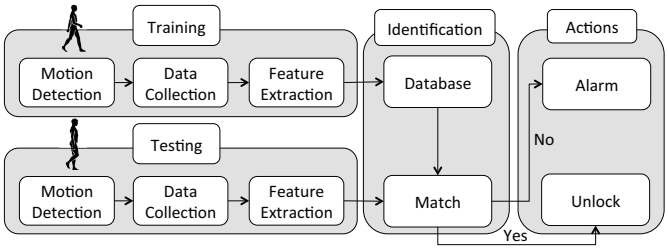

Fig. 1. System construction

II. Related Work

Related work falls generally into solutions to be used once theft is realized and solutions to strengthen the (un)locking mechanism on devices. We have already differentiated our work from these types, as a solution to detect ongoing theft before any (un)locking takes place. The Introduction discusses related work in walking gait authentication and the unique challenges and goals of our system in comparison with these efforts, so this section will focus on other theft-reactant and authentication-strengthening work.

Most theft-reactant applications currently available are based on a combination of GPS, Wi-Fi positioning and cell tower triangulation to track location. GadgetTrack is one popular anti-theft application implemented both on Android and iOS systems [9]. Beside its tracking scheme, it can encrypt photos and contacts on a stolen device and store them in a certified secure data center, wiping any local personal information in the process. A more blunt measure is proposed by Gao *et.al* which locks SIM cards to their respective mobile phones [7]. If the phone is stolen, the owner can call the service operators to disable the lost SIM card, which disallows any usage of the device with that SIM card or any replacements.

In the group of strengthening authentication mechanisms, behavioral and physiological biometrics are increasing in utilization. These attempt to protect sensitive information on stolen devices from being compromised [11] [14]. Li *et al.* implement re-authentication for mobile devices using users' finger movement classified by support vector machines [11]. This system can continuously authenticate the owner during normal phone usage. It monitors and learns the owner's finger movement patterns to compare with those of the current user. Similarly, Shahzad *et al.* propose a gesture based user authentication scheme applying human-distinguishing features such as finger velocity, device acceleration, and stroke time [14]. It is important to note that these methods only start collecting data and doing re-authentication after the phone's password has been entered, while ours does authentication when the thief walks away with it and presumably before the thief gets to a safe location to begin cracking the phone. Finally, Apple's Touch ID represents a physiological biometric unlocking system to authorize users and purchases. A User needs only touch the device's Home button, and the system will authenticate the fingerprint.

III. System Overview

Our system architecture was laid out in Figure 1. The Training module functions as a means to populate the Identification Database with the device owner's unique motion fingerprint. There, the Motion Detection component is responsible for discovering that the device is currently experiencing normal motion behaviors. After the Data Collection component then amasses raw accelerometer data representing the owner's motion behavior, the Feature Extraction component deconstructs this data into features and then assembles it into this fingerprint. The Testing module performs analogous actions to prepare a new fingerprint for the current user to be sent to the Identification module. There, this test fingerprint is compared in the Match component to the owner's fingerprint as retrieved from the Database component. Finally, the decision made by the Match component is sent to the Action module as an Unlock action or an Alert action.

As stated in the Introduction, our challenge is to accurately identify a user within a short amount of time, which requires both to use inexpensive calculations for low computational complexity and to process only a small amount of data for limited required comparisons. More complex tools demand more than acceptable time for comparison between motion patterns, and an unreasonably large number of motion patterns to analyze exacerbates this. These efficiency issues are targeted first in the Feature Extraction component, which houses our pattern synchronization method for processing the raw acceleration data into representative motion patterns. The Match component further handles these considerations with a collection of matching methods with differing strengths, which optimize the comparison process by identifying and considering only the most representative motion patterns for a user. In the subsections below, we further enumerate these integral components conceptually to illustrate the functionality of our system, followed after by technical details in corresponding sections.

A. Pattern Synchronization Method

Existing tools like Dynamic Time Warping [10] and the K-S Test are intuitive solutions to compare between two motion patterns which may vary in time or speed, but first require extraction of these patterns from measurements. This is the goal of our pattern synchronization method, namely the automatic parsing of individual steps from the raw accelerometer data.

Consider Figure 2, showing the results of our feature extraction process. The two persons' data are visibly different; in fact, the step cycles (12 steps) extracted from person A are roughly 100 data points in length, where the step cycles from person B are around 50 data points in length.

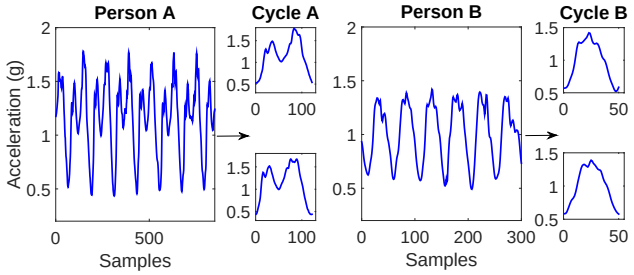

Fig. 2. Examples of two walking patterns

Our step parsing technique addresses several issues. We first define a step cycle as the data between two negative peaks in the accelerometer data. This arises from our observation that as a human lifts a leg, the body is overcoming gravity. The device observes a large downward peak in acceleration, referred to in set theory as a local minimum, in its highest position when gravity is felt the least.

Next, we must select an appropriate representative step cycle which we will use to divide the rest of the data into cycles. We may not simply partition the data into cycles based on the local minima only, because some minima are created by accelerometer noise, and others are caused by irregular human behavior. Consequently, instead of defining a cycle as whatever data appears between two successive local minima, we start with one minima and select from a set of nearby minima to find which resultant cycle width best partitions the data. This exact process is described in Section VI on Feature Extraction, but conceptually involves testing different cycle widths for cutting the data, and finding which width allows a sufficiently high similarity between cycles.

B. Behavior Classification

In the event that a user alters motion behaviors (from walking to running, for example) during a data collection period, we must classify each motion cycle into its appropriate behavior class. To handle this, we perform the pattern synchronization as previously described, resulting in a representative cycle, a set of cycles which correlate highly with it, and the remainder of the data. On the testing side, we simply use the representative and its similar cycles for matching, discarding the rest. For training, however, all non-outlier data is useful, so nothing is discarded. We place those similar cycles in the database as a single behavior, appending them to an existing set if there is also strong correlation between the two sets. We then remove them from the collected data and repeat this process amongst the remaining data until none remains and the database has some number of new behaviors added. In all, a user's database will hold several behaviors, each with correspondent cycles.

C. Matching Methods

After running pattern synchronization on test data and collecting a set of cycles corresponding to one behavior, we attempt to match it to some behavior in the database to verify the user's identity as device owner. Initially, we do this in a conceptually simple traditional manner by comparing all test cycles with every cycle in the database, and its broad data coverage results in high accuracy. Nevertheless, this would be impractical for our application, as the processing time would be a function of the database size, and hence larger databases would allow thieves more time to escape.

We develop a randomization method to choose for comparison a selection of cycles, of static number, to reduce processing cost significantly. Importantly, the decrease in accuracy from using only a few sample cycles is not noticeable. We furthermore design a third matching method which identifies the most important step cycles for comparison. This method retains roughly the same accuracy as the traditional method, runs at least as quickly as the randomization method, and stores a much smaller database than both other methods.

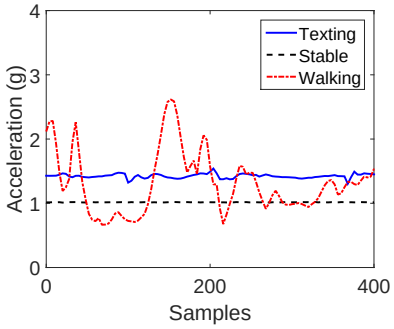

Fig. 3. Device States

IV. Motion Detection

When a device is sitting on a desk or being operated by a stationary user, the data recorded by its accelerometer contains no valuable information that can be used to identify users. Hence we should ignore this data rather than attempting process it into motion step cycles. However we do need to detect the beginning of a motion pattern in a timely fashion. This is important during training and testing to avoid wasting any time in the user identification process. In order to do that, our approach should distinguish between a state of rest or operation and a state of actual motion.

Rest or Operation State: Figure 3 shows accelerometer readings for a device sitting on a desk, being operated by a non-walking user, and being carried by a walking user respectively. The dash line represents the device is sitting on a desk. There is no change of acceleration, which makes it easy to identify. The solid line represents the device is being operated. The subtle changes of accelerometer readings of this case is also easy to distinguish.

Actual Motion State: In contrast to these two cases, a large acceleration is sensed when the device is being moved by the

user, which is represented by the dash-dot line in Figure 3. Hand movement differs as it is normally a one-time motion like picking up a device from the desk, or taking a device out of a pocket. A one-time motion leads to a single large acceleration spike, and so the cycle identification algorithm as introduced in Section III-A will not find multiple similar cycles from the accelerometer raw data.

Other Consideration: Some users enjoy playing video games that use the accelerometer as control input. If a certain motion happens to be repeated frequently enough to appear within the cycle detection window used in Section VI (this is unlikely), our algorithm would identify these motions as motion cycles and add them to the database for the user.

V. Data Collection

The next step after Movement Detection for both Training and Testing modules is Data Collection. On the training side, our approach entails the device collecting accelerometer data each time the user inputs the correct unlock password, for ten minutes or until user termination, and only if the accelerometer registers movement. Our experiment indicates desirably low error rates appear after collecting training data for one week, as discussed in section VIII on Evaluation.

In the event that a user begins to suffer higher false alarm rates, due to changing habits, this training approach may be undertaken anew to regain high accuracy. Also, should a user wear strikingly different clothes or hold the mobile device in different pockets from time to time, training will need to be performed for each such case. This will not force the user to frequently retrain, but rather to simply add some new behaviors to the database from time to time.

For day-to-day testing, after the training period is complete, our approach requests gathering of accelerometer data whenever significant motion is detected. As previously mentioned, we do not wait until the device is unlocked as in second-order authentication schemes, in favor of immediate theft detection.

VI. Feature Extraction

We here enumerate the technical details of the feature extraction component of our system. In doing pattern synchronization for input data, we identify step cycle width and partition the data accordingly for later use in the database and matching components. The complications enumerated in Section III-A are all addressed by this algorithm. At a high level, we find eligible local minima and pick a starting point, testing other nearby minima for their ability to define a suitable data partition width, and, if none work, repeating this for other starting points as necessary until a good cycle is found.

A. Possible Cycle Delimiter Search

Here we wish to take the raw accelerometer data and isolate from its local minima a set of points which may separate out the step cycles. As a mobile device may be oriented in any direction, we first remove the directional components of acceleration using $a=\sqrt{a_x^2+a_y^2+a_z^2}$, where a_x, a_y and a_z represent the values read from each axis [6].

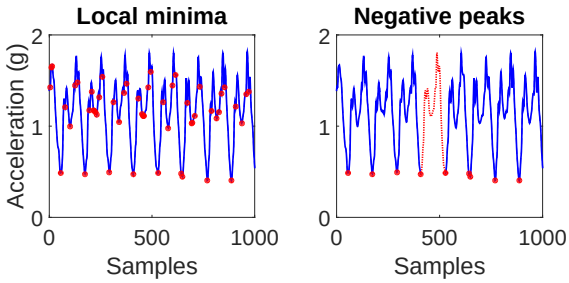

Fig. 4. Relative points and negative peaks

Owing to our usage of a sliding window later in the process, for our next steps we remove from consideration the first and last few moments of data. This padding is included in the Candidate Endpoint Test in Section VI-B but excluded from the Window Placement process as we would otherwise be searching outside of the dataset. This appears as dashed lines in the proceeding figures.

Next, local minima are found throughout the trimmed data, simply distinguished as those points whose neighbors are both of larger value. All points plotted on the acceleration data in Figure 4 are local minima, but as stated in Section III-A, we are concerned only with the large downward peaks. The minima on those downward peaks are our step delimiters, which are a subset of all local minima. Additionally, though not visible on these figures, some minima created by accelerometer noise or irregular human behavior are lower than our step delimiters.

With the understanding that all valid step delimiter minima should be some distance below 1g, the acceleration due to gravity, we set this threshold and analyze the set of local minima satisfying it. We sort the minima in ascending order, then apply a sliding window over these points to determine what range contains the highest number of minima. We found a sliding window of width $0.15\frac{m}{s^2}$ was able to isolate likely step delimiters from outliers most effectively. Figure 4 shows the resultant set of step delimiters for a sample. To proceed, we choose at random one such delimiting point.

B. Representative Cycle Search

This process finds a step cycle length which will effectively split the acceleration data along the cycle starting points. This cycle we call the representative cycle, as it will appear similar to most cycles in the data, thereby providing a template for this data partitioning. We begin with the set of likely cycle delimiters and one of them chosen as a test delimiter. In what follows, we search for what may be the other end of the cycle and test to see if that cycle is a representative cycle.

Window Placement: We restrict the rest of the data to a window around the test delimiter, containing candidate points for the other end of the cycle. This restriction narrows the search space for faster processing. Noting though that invalid points may still enter this set, it should contain multiple candidate endpoints and should be wide enough to accommodate any valid step cycle. For example, some small noise around

the edge of a step cycle might result in two local minima at that edge, one of which should be ignored. Our experiment employed a range of 200 data points on either side of the delimiter and found this befitting of all our volunteers' data sets.

Candidate Endpoint Test: We first test the closest candidate endpoint within the window and move outward to other points if necessary. We use this candidate's distance from the test delimiter to define the test block size. We partition the data set by this test block size and then measure the correlation between each constituent block. In this paper, we use the statistical correlation for its simplicity and rapidity, which is demonstrated in section VII-D. If the correlation is sufficiently high between a sufficient number of blocks, this indicates the test delimiter and candidate endpoint enclose a valid step cycle representative of most other step cycles within the data, i.e. the representative cycle. Conversely, if the correlation is low, another candidate stopping point within the window is selected and the process repeated.

Search Repetition: Should this search return no endpoint that defines a representative cycle, we choose another candidate from our set of possible step cycle delimiters. We repeat the window placement and candidate endpoint tests for this new delimiter, and if necessary, continue choosing others not already tested until one yields a representative cycle.

C. Data Partitioning

Having discovered a representative step cycle, we may now partition the data accordingly so that it may be stored in the Database for training or sent to the Matching component for testing. The representative cycle defines our working cycle size, by number of data points. As the accelerometer collects data at a roughly constant rate, this corresponds to a static time frame of length equal to the duration of the individual's stride. Working outward from our representative cycle, we partition the data into blocks of that length. This results finally in parsed cycles such as those in Figure 2.

VII. Matching Methods

After pattern synchronization, the individual steps are identified and extracted from the raw accelerometer data. Then, the behavior classification procedure groups similar steps (i.e., the steps that are highly correlated) together to form a behavior set of the user. As discussed earlier, a user may exhibit different behaviors like walking and running. Thus, when the training phase is complete, the database will include multiple behavior sets, each containing the step cycles that are extracted from the corresponding behavior.

The matching phase compares unknown step cycles with the cycles stored in the training database to identify any unauthorized moves. Let $\mathcal{B} = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n\}$ denote the set formed by the behavior sets, where $\mathcal{S}_i = \{s_{i_1}, s_{i_2}, ..., s_{i_m}\}$ is the i-th ($1 \leq i \leq n$) behavior set and s_{i_j} is the j-th ($1 \leq j \leq m$) step cycle in $\mathcal{S}_i$. Further let $\mathcal{S}_u = \{s_{u_1}, s_{u_2}, ..., s_{u_l}\}$ denote the behavior set formed by the step cycles from unknown users.

A. Traditional Method: All Cycles

In this strategy, we compare all the elements in $\mathcal{S}_u$ against all the elements in each $\mathcal{S}_i \in \mathcal{B}$. Specifically, $\forall \mathcal{S}_i \in \mathcal{B}$, we compare all $s_{i_j} \in \mathcal{S}_i$ to all $s_{u_k} \in \mathcal{S}_u$ and threshold the average of the comparison results to determine if $\mathcal{S}_i \approx \mathcal{S}_u$. A positive result indicates the unknown behavior $\mathcal{S}_u$ belongs to the owner.

With n behaviors in the database, m cycles per behavior, and l cycles in the unknown behavior, the total number of comparisons is therefore nml. Sufficiently large m and l (enough training and testing cycles) should conceptually result in the best accuracy, because this gives the broadest view of the data. This does, however, require the largest number of comparison calculations that can be made between the database and unknown behavior trace. For the more high-end devices, this may be reasonable, but we offer additional methods focused on optimization, for more universal application.

B. Method 2: Random Cycle Subset

To introduce this method, we note that the step cycles classifying each particular behavior during cycle extraction were those *strongly correlated with each other* as introduced in Section III-B. Therefore, a randomly selected subset of these will continue to strongly correlate with the rest of the set, which should mean similar accuracy in matching, but with only a static number of comparisons independent of database size. Namely, $\forall \mathcal{S}_i \in \mathcal{B}$, we choose a random subset $\mathcal{S}_{i_{RS}} \subset \mathcal{S}_i$ (subscript RS stands for "random subset"). We choose a subset $\mathcal{S}_{u_{RS}} \subset \mathcal{S}_u$ as well. All random subsets are of a pre-configured size q which we derive through experimentation to be large enough to represent the original behaviors. Then, $\forall \mathcal{S}_i \in \mathcal{B}$, we compare all $s_{i_j} \in \mathcal{S}_{i_{RS}}$ to all $s_{u_k} \in \mathcal{S}_{u_{RS}}$ to determine if $\mathcal{S}_i \approx \mathcal{S}_u$, averaging and thresholding as before.

Holding n behaviors in the database, p cycles per behavior, and q cycles in the unknown behavior corresponds to npq total comparisons, compared with nml for the Traditional method. With $p \ll m$ and $q \leq l$, the processing required is orders of magnitude smaller for the typical user. Nevertheless, as with any thresholding system, some step cycles are most correlated with others, while some are consistent enough to be included in the behavior but aren't those best cycles. Ideally we would choose these "signature" cycles to represent a behavior, but if an imperfect step cycle is randomly chosen, it will play a comparatively negative role in making the final matching decision and hence reduce the overall detection accuracy.

C. Method 3: Signature Cycle Subset

To solve this challenge, we propose to identify and use for comparison these "signature" training and testing step cycles. The comparison protocol is similar to that of the random cycle subset method, choosing $\forall \mathcal{S}_i \in \mathcal{B}$ a subset $\mathcal{S}_{i_{SS}} \subset \mathcal{S}_i$ (with subscript SS referring to "signature subset") and a subset $\mathcal{S}_{u_{SS}} \subset \mathcal{S}_u$. $\forall \mathcal{S}_i \in \mathcal{B}$, we compare all $s_{i_j} \in \mathcal{S}_{i_{SS}}$ to all $s_{u_k} \in \mathcal{S}_{u_{SS}}$ to determine if $\mathcal{S}_i \approx \mathcal{S}_u$.

First, however, we must detail how to select a signature subset. The cycles included should achieve the highest consistency with the others in this behavior, to best reflect the typical

motion behavior of the user. We define the most representative step cycle below:

Definition 1: (**Most Representative Step Cycle**) The step cycles extracted from the accelerometer readings form the set $\mathcal{S}$. The most representative step cycle s^* is defined as $\arg\max_{s\in\mathcal{S}} \sum_{x\in\mathcal{S}} F(s,x)$, where $F(\cdot)$ is the comparison function described in Section VII-D. Equivalently, s^* is the step cycle that results in the highest value of $\sum_{i=1}^{|\mathcal{S}|} F(s,x_i)$, where x_i is the i-th step cycle in $\mathcal{S}$.

In a simple extension, the v most representative step cycles in a behavior $\mathcal{S}$ are those resulting in the v highest values of $\sum_{i=1}^{i=|\mathcal{S}|} F(s,x_i)$. These cycles are considered the signature cycles and added to $\mathcal{S}_{SS}$. An important aspect of this process is the fact that it may be done for the training dataset once training is complete. This preprocessing trims the database to hold only the signature cycle subsets which lowers its data storage footprint significantly compared to the traditional Methods and Method 2. Additionally, the total number of comparisons is nvq, which q is the number of cycles in the unknown behavior, like Method 2.

D. Comparison Functions

A step cycle is a portion of the accelerometer reading, and so is a vector of accelerometer sample points. To compare two vectors of step cycles, we may utilize existing classification tools as mentioned before (e.g., K-S statistical test [12], Total Variation Distance [1], or statistical correlation [8]). Regardless of the type of classification tool, each returns a comparison outcome on whether or not two vectors are similar.We calculate the average of the comparison outcomes and compare it with a threshold to make a matching decision.

The proposed scheme does not impose limitations on which kind of comparison tools should be used. In this paper, we consider the statistical correlation for its simplicity and rapidity. Specifically, let $\mathbf{x} = [x_1, x_2, ..., x_m]$ and $\mathbf{y} = [y_1, y_2, ..., y_n]$ denote two vectors of length m and n respectively. The correlation between $\mathbf{x}$ and $\mathbf{y}$ is calculated by $\frac{\sum_{i=1}^{i=\min(m,n)} x_i y_i}{\min(m,n)}$. Assuming that we made a total of γ comparisons between $\mathcal{R}_T$ and $\mathcal{R}_U$, each comparison returning a correlation value, we then compare the average of the γ correlation values to a threshold. If the average is larger than the threshold, then the unknown behavior is identified.Otherwise, the unknown behavior does not match the current behavior set and it is compared to the next behavior set until a match is identified. If no matching is found after all the behavior sets are exhausted, the move is considered unauthorized.

E. Methods Discussion

We will provide a thorough examination of the effectiveness of each method in the following section on Evaluation, but compare the methods more conceptually here with regards to computational complexity. The traditional method is the slowest. In our experiment, after two weeks of training, the user converges to 22 behaviors and each behavior comprises 300 step cycles. We used 10 cycles extracted from the unknown data for comparison. Thus, the number of comparisons is $66,000$ $(22 \times 300 \times 10)$, for which we find an accuracy of 97.5% distinguishing between users and 97.8% identifying users as themselves.

Method 2 is expected to have a good performance in detection accuracy when the value of p (the number of step cycles in each training behavior) and q (the number of step cycles in the known behavior) are large enough to represent the original behaviors. So how large should p and q be? We find in experiments that $p = 100$ and $q = 10$ gives an accuracy similar to the traditional method, while the number of comparisons is $22,000$ $(22 \times 100 \times 10)$. This means that Method 2 is theoretically 3 times faster than the traditional one for the same accuracy. The performance becomes unstable when there are not enough cycles to represent the whole behavior, i.e., when p and q are small. In particular, when $p = 1$ and $q = 10$, accuracy lowers to 94.6% distinguishing between users and 86.7% identifying users as themselves.

For Method 3, we find in experiments that the proposed detection system can achieve a relatively good performance using only a single signature step cycle (the most representative one) from each training behavior. In this case, $p = 1$ and the number of comparisons becomes 220 $(22 \times 1 \times 10)$, which is theoretically 300 times faster than the traditional method. Nevertheless, the achieved accuracy is similar to the traditional method with 96.4% distinguishing between users and 97.8% identifying users as themselves.

VIII. Evaluation

Our experiment involved 45 volunteers comprising 10 women and 35 men. All are students or researchers in our university, with ages ranging from 18 to 50. During the data collection sessions, each participant was instructed to walk and run 20 meters, holding the data collection device in a variety of positions. These included in the left hand and right hand, in a bag held by the left and right hand, and in a backpack, for a total of 5 locations per movement type.

We first examine the correlation threshold for ensuring step cycles are extracted correctly. Next, we present our evaluation metrics and methodology, following this with a discussion on comparison threshold optimization for each of our matching methods. Training and testing time complete the evaluation.

A. Cycle Extraction

We cover in Section VI the technical details involved with partitioning raw data into step cycles and here ensure that the number of cycles extracted matches with our observation of the data. In particular, a cycle width is chosen which cuts the data into a series of blocks that correlate highly with each other. Through this experiment we define the term "highly."

Our observations of the raw data indicate volunteers took 40 steps on average, for each motion and device location trial. Some took more or less than that, but an average of 40 steps suggests that we need a correlation threshold that results in an average of 40 steps for each test. We test thresholds ranging from 0.6 to 0.9, and plot the results in Figure 5. This illustrates that with a correlation threshold of 0.8, an average of 40 cycles

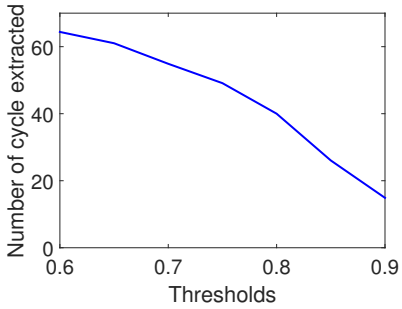

Fig. 5. Effects of threshold on cycle extraction

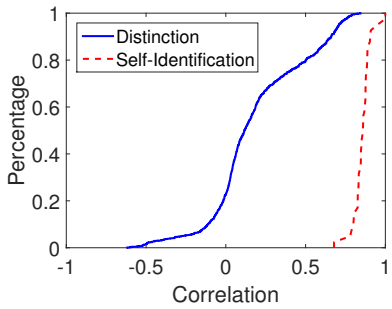

Fig. 6. CDF for thresholding Method 1

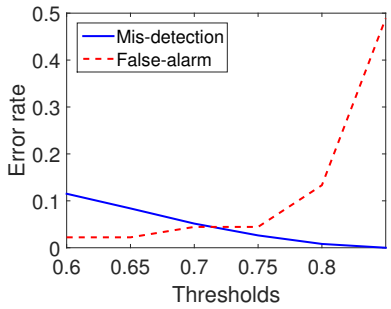

Fig. 7. Mis-detection and false-alarm for Method 1

are extracted from each trial, as desired. We use this threshold for cycle identification throughout all further evaluation.

B. Matching Methods

The metrics we use to evaluate our three matching methods in the following sections are Distinction and Self-Identification. Distinction refers to the differentiation of individuals from each other, and represents the recognition of the current device holder is a thief. Self-identification refers to the matching of users to themselves, representing the system verifying the current device holder as the owner. Naturally, we wish for these metrics to yield a high percentage and their inverse error rates to be low. The Mis-Detection error rate is the inverse of the distinction rate, showing the portion of volunteers who would be able to impersonate some others. Similarly, the False Alarm error rate indicates the number of volunteers not correctly identified as themselves, who would be wrongly flagged as thieves.

Ascertaining these error rates involves a comparison of each volunteer's data to that of every volunteer. Mis-detection is defined mathematically as the number of tests where we cannot distinguish between individuals. For 45 people, each analyzed against every other (but not themselves), this is some fraction of 1980 tests. The false alarm rate describes the amount that individuals cannot be recognized as themselves, so for 45 people compared only to themselves the false alarm rate is a portion of 45 such tests. Hence, to calculate the mis-detection rate, we use the full data sets for each volunteer, but for the self-comparison required to find the false alarm rate, we use half a user's data for training and the other half for testing.

Our evaluation testing differs slightly here from what would occur in practice, as we are attempting to match users' full databases to each other, rather than one sample behavior vector to a database. In other words, in practice we collect a single test behavior and try to find a match, but for this evaluation we are trying to match any of the behaviors in one database to any in the other. As such it should be noted that this is akin to a thief having several chances to steal a device, so the low error rates are actually an upper bound on the error.

1) Method 1: All Cycles: To refresh, our preliminary matching method uses all available cycles to inform its decision, comparing cycle in one data set to every cycle in the other, and finding the average of these correlations. This averaging reduces the impact of noise to better reflect the situation as a whole, so this method should be the most accurate. Figure 6 depicts a cumulative distribution function (CDF) view of our partitioning efforts, suggesting a correlation threshold somewhere around 0.75 is best. Most self-identification tests find correlations larger than this value, and most distinction tests arrive at a lower value.

With this realization we optimize the error rates by varying the correlation threshold between 0.6 and 0.85, with results visible in Figure 7. We find an intersection of the error rates

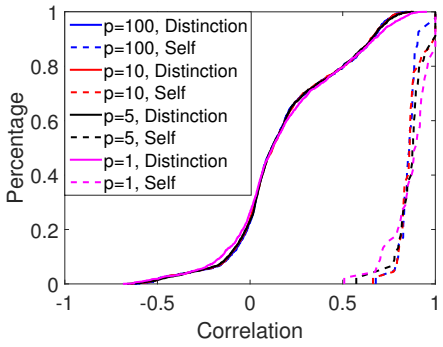

Fig. 8. CDF for thresholding Method 2

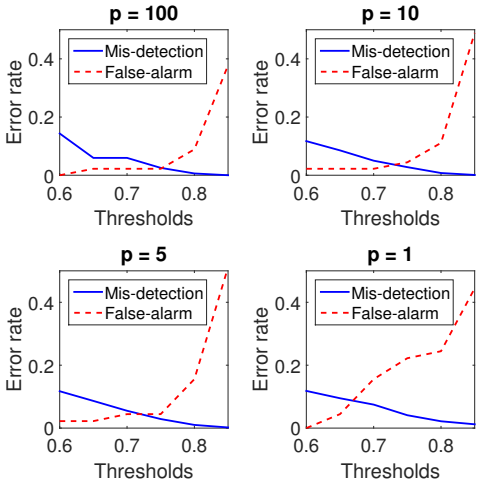

Fig. 9. Mis-detection and false alarm rates for Method 2

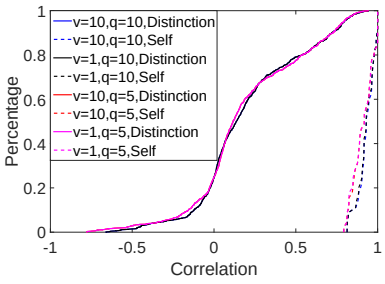

Fig. 10. CDF for thresholding Method 3

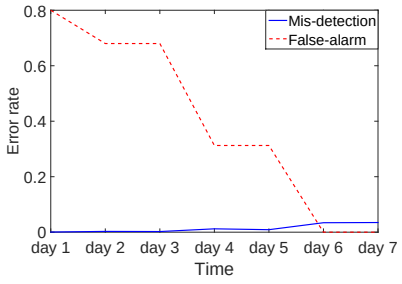

Fig. 11. Effects of varying training time

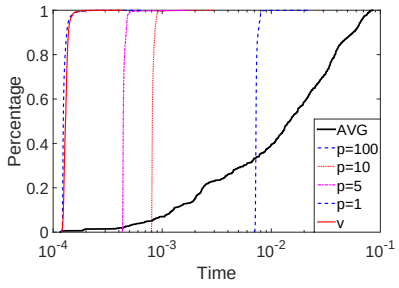

Fig. 12. CDF of running time with multi-methods

around a threshold of 0.75, which allows only 2.5% mis-detection and 2.2% false alarm rates. This corresponds to 97.5% distinction and 97.8% self-identification rates, which is indeed nicely accurate.

2) Method 2: Random Cycle Subset: This method revisits the use of correlation average but only for a static number of cycles, in an effort to optimize processing time. Choosing a number p of cycles randomly from each behavior, we compare each from one behavior to every one from the other behavior, and thereby limit to a fixed number of computations. A threshold is then applied to the average correlation between this subset of cycle pairs. In Figure 8, the CDF reports distinction and self-identification curves for various p. While the distinction curves are fairly similar for different p, there is some separation in the self-identification curves. A higher p, of 100 cycles, is more desirable than the low p of 1 cycle.

Reviewing Figure 9 and the error rates for four p values depicted there, the degradation of accuracy with lower p is visible. We find that while an p of 1 causes undesirable error rates, an p of 100 results in a high accuracy, which is the same as the accuracy of the Method 1. The error rates for p equal to 20, 10, and 5 are 2.6%, 2.8%, and 3.9%, respectively, for mis-detection, and 4.4% in all cases for false alarms. This method finds that $p = 100$ is large enough to represent the original behaviors and achieve comparable accuracy to Method 1.

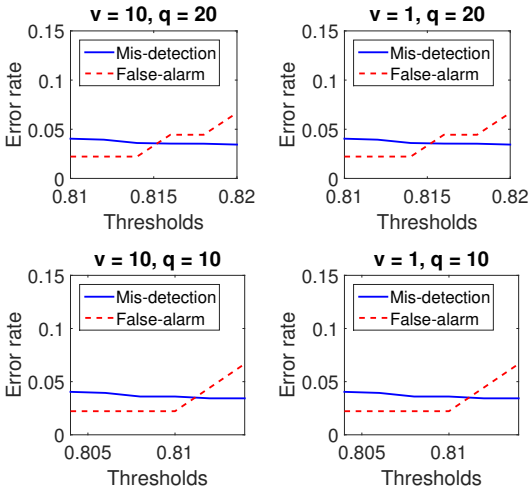

Fig. 13. Mis-detection and false alarm for Method 2

3) Method 3: Signature Cycle Subset: The final matching method builds from the random cycle subset method and attempts to decrease the training database storage footprint, through identifying behavior representative cycles after training and storing only those. Again we must choose a number v of cycles to use from the training behavior sets, but unlike the previous method we only store these cycles. We alter testing behavior set length independently to compare with these v cycles per training behavior. Specifically, we try testing set sizes of 10 cycles (20 steps) and 5 cycles (10 steps), and compare them with 10 signature cycles and 1 signature cycle. Each cycle from the database's behavior sets is compared with all cycles from the test behavior, and their correlations averaged as before.

Figure 10 indicates that the accuracy is rather similar among these options, meaning that a user can opt for slightly lower accuracy in order to achieve a faster detection time. That said, Figure 13 indicates that with a correlation threshold of 0.814, a training size v of 10 cycles and a testing size of 10 cycles (20 steps) provides the best accuracy with a mis-detection rate of 3.6% and false alarm rate of 2.2%. Although it is not as good as the Method 1, this method saves a lot of comparison time, one of the important goals in our paper.

C. Training Time

A good anti-theft system should not be intensive in its initial setup. To ensure that our system satisfies this requirement, we analyze the effects of varying training times on the resulting accuracy. Specifically, we asked one volunteer to continue collecting data every day for two weeks. We used the first half of this data as the training set and the second as the testing set. Then we included increasing portions of the training set and measured the accuracy against the full testing set. The accuracy relative to number of training days included is shown in Figure 11. With more training data available, the false alarm error rate, otherwise described as the difficulty in identifying the user as that individual, decreases sharply.

D. Detection Time

Our application also requires the quick detection, so that a thief may be identified before eluding the owner. Detection time is measured over two components, data collection and data analysis. The former refers to the number of steps that

the attacker must take before an accurate decision can be made, while the latter refers to the processing time of that decision.

For data collection, we reiterate the success of the cycle subset matching method using ten randomly chosen step cycles from training and testing datasets, as well as the success of the signature subset method using five or ten cycles. We conclude then that 10 to 20 steps are all that is necessary to provide a testing dataset with our stated accuracy of 96.4% to 97.5% distinction and 97.8% self-identification.

Data analysis time is the time required for parsing the collected data into step cycles and then behavior data, followed by comparison with the database. This time varies with the matching method used for comparison, so a CDF of processing time stemming from different methods appears in Figure 12. "AVG" refers to the traditional method of averaging all data, and the lines prefixed "p=" correspond to different values of p for the random cycle subset method. "V" is the average case for the signature cycle subset method, the variants of which all performed roughly the same. The traditional method is clearly slowest, while the random subset method has varying runtimes based on the subset size. The signature cycle subset method with all tested parameters is very close to the random subset method with $v = 1$, confirming it as the most cost-efficient method. This method provides a match decision in fractions of a millisecond, still with the above-mentioned high accuracy.

TABLE I
POWER CONSUMPTION

State	Initial	20 Mins	40 Mins	60 Mins
Motion state	54%	52%	50%	47%
Non-walking	47%	47%	46%	46%

E. Power Consumption

Mobile device battery life is precious, and, as such, we must verify that our system incurs sufficiently low overhead to maintain usability. As mentioned in Section IV, our approach only processes data when the device is in the actual motion state. The rest of the time, no computation is necessary aside from cursory supervision of the accelerometer, so battery life will be less affected. We monitor the power usage of our approach in different states for a one hour duration using an iPhone 6 plus; table I shows the result. The initial power of the device is 54%, and after one hour in the walking state, 47% of power remains, for a 7% power consumption per hour of walking state. Likewise, with an initial battery status of 47%, 46% of power remains after one hour of processing in the non-walking state for a consumption of roughly 1% per hour while monitoring for state changes. To better quantify these results, consider a two-day study by The New York Times in 2003 tracking walking habits of 1,136 adults around the United States. The study found that Americans take about 30 to 40 minutes of walking per day on average [13]. With this duration of walking state processing and the remainder of 24 hours time monitoring for walking state change, our approach requires at most 3.5% to 4.7% of the battery each day.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under grants 1527144 and 1553304, and the Army Research Office under grant W911NF-14-1-0324.

IX. CONCLUSION

We proposed a fast anti-theft system that can detect authorized movement of mobile devices. Simply speaking, we perform authentication whenever the device is moved via walking. To design such a system, we created motion synchronization techniques that can extract step cycles from the raw accelerometer data to enable comparisons between individuals. We also created a representative matching algorithm to compare the "signature" step cycles for a behavior instead of comparing all possible data, for improved accuracy and reduced comparisons. We performed extensive experimental evaluation using the accelerometer data collected from 45 volunteers. Our experiment results show that the proposed system can successfully detect an unauthorized move within 10 to 20 steps by a detection accuracy of 96.4% to 97.5%, while also distinguishing the current move as by the owner 97.8% of the time, and requiring at most 4.7% battery overhead for the typical user.

REFERENCES

[1] J. A. Adell and P. Jodrá. Exact kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, page 64307, 2006.

[2] A. Annadhorai, E. Guenterberg, J. Barnes, K. Haraga, and R. Jafari. Human identification by gait analysis. In *Proceedings of the 2rd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, pages 11:1–11:3, 2008.

[3] S. Argyropoulos, D. Tzovaras, D. Ioannidis, and M. G. Strintzis. A channel coding approach for human authentication from gait sequences. *Information Forensics and Security, IEEE Transactions on*, 4(3):428–440, 2009.

[4] G. Bajrami, M.O. Derawi, and P. Bours. Towards an automatic gait recognition system using activity recognition (wearable based). In *Proceedings of the Security and Communication Networks, 2011 Third International Workshop on*, pages 23–30, 2011.

[5] N. Cristianini and J. S. Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[6] D. Gafurov, K. Helkala, and T. Sondrol. Biometric gait authentication using accelerometer sensor. *Journal of computers*, 1(7):51–59, 2006.

[7] Y. Gao, C. Zhou, and D. Shang. A smart phone anti-theft solution based on locking card of mobile phone. In *Proceedings of the 2011 International Conference on Computational and Information Sciences*, pages 971–974, 2011.

[8] M. Hazewinkel. *Correlation (in statistics), Encyclopedia of Mathematics*. Springer, 2001.

[9] ActiveTrak Inc. Gadgettrak ios security. http://www.gadgettrak.com/products/iphone/. [Online; accessed Feb.-2015].

[10] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[11] L. Li, X. Zhao, and G. Xue. Unobservable re-authentication for smartphones. In *Proceedings of the 20th Annual Network and Distributed System Security Symposium (NDSS'13)*, 2013.

[12] P. Olofsson. *Probability, Statistics, and Stochastic Processes 2nd edition*. John Wiley, 2012.

[13] T. Parker-Pope. The pedometer test: Americans take fewer steps. http://well.blogs.nytimes.com/2010/10/19/the-pedometer-test-americans-take-fewer\-steps/?_r=0. [Online; accessed Mar-2015].

[14] M. Shahzad, A. X. Liu, and A. Samuel. Secure unlocking of mobile touch screen devices by simple gestures: You can see it but you can not do it. In *Proceedings of the 19th annual international conference on Mobile computing and networking (MobiCom'13)*, pages 39–50, 2013.

[15] D. Tapellini. Smart phone thefts rose to 3.1 million last year, consumer reports finds. http://www.consumerreports.org/cro/news/2014/04/smart-phone-thefts-rose-to-3-1\\-million-last-year/index.htm. [Online; accessed May-2014].