CASTRA: Seamless and Unobtrusive Authentication of Users to Diverse Mobile Services

Devu Manikantan Shila and Kunal Srivastava
United Technologies Research Center, Hartford, Connecticut 06108
Email:{manikad, srivask}@utrc.utc.com

***Abstract*—This paper presents CASTRA (*Context-Aware Security Technology for Responsive and Adaptive Protection*), an “always-on” context-aware authentication and access control framework that seamlessly and unobtrusively authenticate users to mobile applications of varying sensitivity levels. CASTRA uses a continuous and multi-faceted *behavioral biometrics authentication* that passively authenticates the user in the background while the device is being in contact with the user, and a *context-aware risk assessment and access control* that provides access to applications based on the perceived threat level around the device. The behavioral authentication module is constructed by exploiting a combination of supervised and unsupervised learning techniques on raw sensor and GPS data passively gathered from the mobile device. Multiple inferences about the user (or *user behavioral traits*) such as frequently visited locations, location transition patterns, physical proximity of user with the device (e.g., device in the pocket or placed on the table), and walking patterns are automatically inferred and extracted. Analytical studies were conducted to derive optimal thresholds to fuse these multiple traits and an *adaptive trust score* is generated every user-defined time period to determine the degree to which the user is trustworthy to access the applications.**

CASTRA is implemented in a client-server mode, utilizing the Android and the Amazon Cloud computing platform. The novelty of CASTRA stems from the design and fusion of multiple behavioral biometric based authentication factors and the development and deployment of a practical end-to-end architecture that enables real-time data acquisition, automatic training and learning of user behavioral patterns, and context-aware risk assessment and access control. The performance of CASTRA was evaluated under natural settings, on 15 subjects, using different variants of the Samsung devices. Multiple realistic attack scenarios (e.g., stolen, lost and shared devices) targeting mobile devices were designed to prove the security and user-friendliness of the proposed scheme. We also present techniques to reduce energy and bandwidth consumption and ways to unobtrusively acquire data for supervised learning algorithms without requiring explicit user annotation.

I. Introduction

The wide spread use of smart phones in conjunction with the advent of smart things, Internet of Things (IoT), will enable seamless and frictionless user experience in future, i.e., ability to automatically recognize user presence and control settings based on user preferences. Mobile applications (aka IoT apps) are extensively used for controlling and managing these smart things (e.g., thermostats, IP cameras, home automation, automobiles). For example, access cards are nowadays replaced by IoT apps (called as *mobile identity badges*) in physical access control and other application domains (e.g., facilities, automobile). When these IoT apps are accessed, users rely on password, PIN or biometrics for user recognition [11]-[12],[16] and personalized experience. Nonetheless, these traditional authentication techniques make it inadequate for users to fully realize the benefits of *seamless and frictionless authentication and access to smart things*, while ensuring strong protection of sensitive apps (or data) from unauthorized access. Existing authentication techniques, as indicated by prior efforts [2],[3],[4], [14], [16], face a number of weaknesses that include: *degraded user experiences* (requiring users to authenticate multiple times when the device is used), *lack of user-specific service access rights* (e.g., only user i has access to application A), *poor security practices* (e.g., automatic sign-in, disabling passwords or setting longer timeouts), *insufficient security provided* (e.g., PINs and passwords are susceptible to brute force, smudge and social engineering attacks), and *lack of continuous authentication* (e.g., knowledge based and biometric solutions that authenticates user once and leaves device in “trusted state” till timeout). A majority of these problems can be resolved by leveraging the processing and sensing capabilities of mobile devices to create user-specific unique signatures based on behavioral biometrics that can enable *usable security* [4], [16], [9], [10], [11], [12]. Conversely, single modality behavioral biometrics have also shown to pose issues with reliability and user distinctiveness

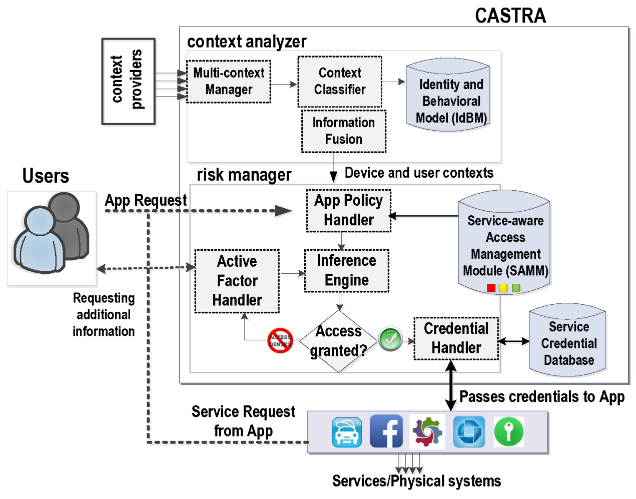


Figure 1: High level overview of CASTRA, as deployed on a mobile device

due to its probabilistic nature.

In this work, we focus on **seamless and frictionless authentication of users to physical end devices (IoT)** by presenting a continual, secure and user-friendly authentication and risk assessment technology called CASTRA (*Context-Aware Security Technology for Responsive and Adaptive Protection for mobile users*). CASTRA alleviates three main issues seen in existing techniques i.e., *security, usability and reliability*. CASTRA realizes this by *continuously and unobtrusively observing and inferring the context* around the device and by *adapting application access decisions* based on the perceived level of threats and the context around the device such as confidence in user's identity (i.e., trustworthiness of the user accessing the device), sensitivity level of the application accessed by the user, geo-fencing (location) constraints, application trustworthiness, etc. The authentication module of CASTRA is based on using a multi-factor behavioral biometrics solution that exploits various user behavioral traits such as walking patterns, frequently visited locations, mobility patterns (e.g., transition patterns of user from one location to another), and also physical proximity with the device (e.g., position of the device i.e., on the table or in contact with the user). On the top of these passive user behavioral patterns, CASTRA uses active authentication factor such as fingerprint, voice or facial detection to add an additional layer of security and to prevent lock-outs of legitimate users. The multiple behavioral traits of user are fused using optimal thresholds, every t seconds, to generate an *adaptive trust score* that determines the degree to which the user is authentic to access specific app(s) during a given period. The trust score is adaptive as a new score is generated with every positive/negative observations. The objective of CASTRA is to detect unauthorized user quickly through this adaptive score and lock access to senstive services.

A. CASTRA Architecture

Figure 1 graphically illustrates the high level CASTRA architecture, as deployed on a mobile device. CASTRA comprises of two key modules: (a) the **context analyzer** that uses machine learning and data fusion techniques to continuously and passively determine *user trustworthiness* (e.g., is the device with the right owner or not?) and *the context around the device and the apps* (e.g., is the device located in unsafe place?, are untrusted and trusted apps active at the same time?); and (b) the **risk manager** that provides adaptable application/service access decisions based on the perceived level of threats, context and the sensitivity of the service accessed or requested by the user or the app.

Within the **context analyzer**, the *multi-context manger* continuously senses and receives the information from the **context providers** of interest via a publish/subscribe architecture. The information gathered from the context providers include sensor measurements (accelerometer, gyroscope, proximity, magnetometer), application and device states, GPS (latitude and longitude with accuracy), and also user activity levels polled using the `Google Activity Recognition APIs` [32]. Relevant features from the data are extracted and utilized by *context classifier* module to infer the trustworthiness of the user carrying the device and also the context around the device (e.g., state of active applications, location constraints etc.). To deduce user trustworthiness, classifier uses the user and device specific *Identity and Behavioral Models* learned during the training phase. The *information fusion* module lastly combines multiple observations about user (i.e., confidence scores with regard to walking patterns, proximity, and location patterns) reported by the classifier module to compute the final confidence level in user's identity (called the *adaptive trust score*). This information from **context analyzer** is further used by the *Inference Engine*, within the **risk manager**, along with the application specific policies retrieved by the *App Policy Handler* from the *Service-aware Access Management Module (SAMM)* database to decide whether an identified user/app has authorization rights to access a specific resource under the given context. The SAMM database contains sensitivity level of the apps (e.g., high, medium and low) and various contextual policies (e.g., is the user allowed to access the app under the observed trust score and context? can an app be active under a given context?). For instance, assume that user assigns high and low sensitivity scores for financial and entertainment apps, respectively. Under CASTRA contextual policies, user will not be granted access to financial app unless a high trust score and a safe context is observed around the device. To prevent lock out of legitimate users, if the access to an app was denied, the CASTRA policy would allow access under stronger authentication guarantees, i.e., the *Active factor* handler will ask the user for more additional information such as perform active biometrics. The *Inference engine* will execute the policy again and if the access is granted, the *credential handler* will access the credential database belonging to the service to automatically complete the login process and provide the policy-compliant level of access. The credential database that resides on the device stores a credential for every service/resource that a user needs to access and will enable **seamless authentication of users to applications with enhanced security and user experience**.

Roadmap: The rest of the paper is organized as follows: In Section II, we present the related work in the area of implicit and context aware authentication. Section III presents the contributions of this paper. Section IV details the design of context analyzer and the performance of individual algorithms are demonstrated in Section VII-A. Section V presents the risk manager and the detailed implementation of CASTRA is discussed in Section VI. In Section VII, we present CASTRA performance. Finally Section VIII concludes this paper.

II. Background

This section summarizes related work in the field of *behavioral biometrics*, especially pertaining to single and multiple modalities. Behavioral biometrics, as opposed to knowledge based (e.g., passwords, PINs, pattern locks) and physiological biometrics, identifies human by their measurable behavior, i.e., the way they interact with the device rather than the physical attributes, enabling more secure and user-friendly solution. In spite of the availability of many solutions in this space, the specific "trait" chosen for user authentication depends on the context of the application where an activity will be highly likely to perform (e.g, speech or call patterns for authenticating caller to callee, touch/keystroke dynamics for authenticating user to the websites etc.). We call those traits as application-aware traits. Existing efforts can be further grouped into *activity-aware* and *intrusive* solutions. Activity-aware approaches such as gait, keystroke dynamics, speech patterns etc cannot identify the owner of the device, if the user is not performing any activity. On the other hand, intrusive solutions using touch/keystroke dynamics require modifications to existing app or the operating systems to understand user touch and keystroke patterns.

1) Single Modality Behavioral Biometrics: Behavioral biometrics proposed for mobile devices include touch dynamics, keystroke analysis, device placements while answering the call, proximity of device to objects (Beacons, Bluetooth proximity) or with the user, call usage, location and gait patterns [1],[4]-[5],[9]-[12],[16]-[18],[23]-[30]. Solutions like touch dynamics/gesture, keyboard dynamics and call usage patterns are activity-aware and require constant user interaction with the screen and cannot authenticate a user when the device is idle (e.g., when the device is stored in the pocket). Further, most of these efforts uses an intrusive approach to retrieve the data (touch and keyboard actions) from the device, except [23]-[24], where authors learn keystroke dynamics from underlying accelerometer and orientation sensors. While these solutions are good to authenticate users during constant interaction with the device (i.e., owner answering a call or authenticating to a website), it raises **concerns when mobile devices are used as identity badges to access physical entities** where requiring users to interact with the device for successful authentication is impractical. Device placement based techniques (or physical proximity) on the other hand only account for the physical interactions between the device and the user (e.g., whether the user is holding the device or is the device stored in the pocket or placed on the table) or between the device and the surrounding objects such as Beacons and fails to capture specific user behavioral traits (see Google Smart Lock [17]). It is not hard to see that these solutions fail to detect attack vectors such as shared or stolen devices (e.g., snatching the device from the user by an attacker), as device lacks a mechanism to infer whether the user is an attacker or owner. Efforts extracting signatures from user movements made while answering a call identifies owner only when the specific activity is performed and may be inadequate for access control solutions.

Gait-based solutions (authentication based on user walking or movement patterns) enable the device to unobtrusively and continuously recognize its user based on his specific walking patterns; which seems as one of the highly probable events that a user will perform while accessing the physical entity. As reliability is a key issue for any probabilistic measures, gait based recognition algorithms are fused with other implicit factors, in this paper, to amortize the cost of false negative and false positive rates. With regard to mobile identity badge application, CASTRA uses activity-agnostic and non intrusive factors based on user location and device proximity patterns. Riva *et. al* first introduced device proximity based authentication, which looks for the proximity of device with the user after initial biometrics for continuous validation; by augmenting device proximity with user gait, adversarial threats such as stolen, lost and shared identity badges can be easily detected. Location serves as a useful factor for passively authenticating user based on the frequently visited places (e.g., home, work). We specifically use human mobility (transitions between location clusters) and temporal based location patterns as the third authentication factor. In [26] authors propose Hidden Markov Models to capture human path traces for authentication. One issue that stems from the fact of using location merely as an authentication factor is its inability to detect shared or insider attacks as both adversary and owner shares the same location.

The performance of current behavioral biometric solutions in natural environment is unknown, as user will be performing mixture of activities such as sitting, standing, running and driving, in addition to the activity of interest. For instance, user can multi-task by walking and interacting with the phone (key stroke, touch) [23] - [24] at the same time and a model designed to detect keystroke/touch/user movement patterns from underlying motion and orientation sensors will perform poor when tested on such complex data. Hence one should focus on extracting segments corresponding to the activity of interest with high confidence, by clearly identifying all possible contexts around the device. In this work, we automatically annotate and segment the walking data from such noisy sensor datasets (see Section IV)

2) Multi-factor and Context-Aware Solutions: The efforts in [11],[12],[17],[25], [28], [30] uses multi-factor behavioral biometric authentication and are closest to CASTRA. The work in [25] uses a Bayesian based approach to fuse location patterns with active factor such as PINs for optimal decision making. Their work uses simple rule based approach for learning location patterns and are inadequate to serve as a strong factor for access control domain. PersonalA [27] learns user activities and derives unique traits based on the set of activities performed by user using topic modeling. These solutions may require multiple observations to build the trust score and may not provide instantaneous authentication. ConXSense [28] uses a combination of GPS, WiFi and social interaction based contexts of interest for authentication. For insider threats or shared attacks, an attacker and actual owner may share the same contexts, leading to increased false negatives. ITSME [30] uses a fusion of slide and swipe patterns performed while unlocking the phone with voice recognition for user authentication; their approach requires interaction with the device for authentication and fails to enable frictionless authentication. CASTRA also differs from Google Smart Lock through the use of algorithms to automatically learn and authenticate based on user-safe locations and gait patterns. Existing resource access control solutions on phone employ an *all or nothing* approach where the user is either allowed to access applications (regardless of the sensitivity) installed on the device, if successfully authenticated, or rejected. There are various efforts e.g., Apex [6], Crepe [7], Kirin [8] etc., focused on designing fine grained access control policies based on the context (e.g., user locations, presence of wireless networks, time, application white listing and blacklists, battery rate etc). One major downside with these current efforts is the inability of policies to regulate application access based on the trustworthiness of the user accessing it and the sensitivity level of the apps. We resolve this issue by developing a **plug-n-play user trustworthiness aware risk manager** that can co-exist with the access control solutions provided by the systems today such as Apex [6], Crepe [7], Kirin [8].

III. Contributions

This paper presents the design and implementation of a practical behavioral biometrics based mobile authentication system that ranges from real time data acquisition with automated label annotations to training, learning and refinement of user behavioral models and context-based risk assessment and access control. The detailed design of our multi-factor authentication is discussed in our prior work [16] and is omitted here for brevity. In short, we make the following contributions:

Continuous, secure and user-friendly authentication: We use a combination of active and behavioral biometrics for seamless and continuous authentication of users to mobile applications of various sensitivity levels (low, medium and high). The behavioral traits learned includes *proximity model* (estimates the proximity between the user and the device), *gait model* (recognizes user based on walking patterns) and *location model* (authenticates user based on frequently visited locations and transition patterns from one location to another). We use supervised (e.g., Random Forest Trees) and unsupervised learning techniques (e.g., One-class SVMs and K-Means clustering with quad trees) [13] on multi-modal time series data to learn *proximity*, *gait* and *location* models, respectively. The key distinguishing factor and novelty of CASTRA stems from the implementation of efficient and non-intrusive techniques to extract multiple behavioral traits from sensor data, thereby ensuring faster and accurate training, learning, and inference. This includes: (i) optimal tuning of hyper-parameters of classifier models (e.g., One Class SVMs [13]) by exploiting a carefully chosen subset of subject's data in addition to normal users data during training phase. This in turn facilitates classifiers to fine tune the decision boundary of the normal user and make it less prone to false negatives or false positives; (ii) optimal selection of cluster centers by using quad trees on location coordinates helps to eliminate the typical drawbacks seen in K-means clustering algorithm [13]; (iii) incremental learning and update of locations, without requiring to retrain the locations from scratch.

Multi-score fusion and risk assessment: We develop an adaptive and weighted trust score computation algorithm that fuses multiple scores reported by *gait*, *proximity* and *location* models in conjunction with the historical state of the device and of the user in a continuous fashion to determine the degree to which user is authentic to access a specific service or application on the phone. Analytical studies are performed to derive the optimal weights/thresholds for each authentication score that minimizes the sum of false alarm and missed detection probabilities.

User friendly training and deployment of models: We implement **context analyzer** as Android services that runs in the background and automatically acquires data, extracts features and initiates cloud services to train, learn and deploy user-specific *gait* and *location* models. The mobile data (e.g., sensor measurements and GPS) were unobtrusively acquired from the user without requiring explicit user annotation (i.e., user activity states, semantic locations) through the use of Google APIs and unsupervised learning approaches.

Performance evaluation and analysis: We analyze machine learning algorithms and performance of CASTRA under various natural and adversarial settings. Our results show that one-class SVMs based *gait* model, learned using 20-30 minutes of multi-modal walking data (data extracted with a walking confidence score of 75% and above) can identify adversaries with a True Positive Rate (TPR) rate of 99%, when invoked at every 10 seconds with a 50% overlap window. The *proximity* model learned using Random Forest trees, when evaluated on multiple subjects, were able to identify device position i.e., is the device on the table and with the user after initial biometrics, with a TPR of 90.6% and 91.4%, respectively. The performance of our *location* models depends on the availability of training data and results show that a reasonably accurate location model can be obtained with one week worth of data (including weekdays and weekends). The performance of CASTRA were evaluated using various attack scenarios involving trusted and untrusted locations, different placement of the device (with the user or on the table), and simulating *insider, lost, stolen and shared attack scenarios*. Our evaluation on 15 subjects manifest that CASTRA can identify adversaries and lock out sensitive resources within $t - 4t$, where time period t denotes the time at which a decision is available from the **context analyzer**. The detection latency stems from the nature of the attack (i.e., the amount of data available for decision making), and the communication and computational latencies. Run-time performance analysis indicate that CASTRA incurs a maximum data usage of 8kB/s (when sensors are off) and 33kB/sec (when sensors are on), over Wi-Fi, and a battery drain of $0.5\% - 1\%$, every hour, clearly depicting its feasibility for commercialization.

Model retraining support: We present techniques and framework to retrain the machine learning models, on the fly, whenever new data is encountered. Such an architecture enables one to perform quick deployment of initial models (within 2 hours of training) and perform incremental refinement of models with every new unseen data (e., new location visits or walking patterns etc.).

A practical developed system for industry applications: We deployed a practical behavioral biometrics based authentication and access control system that performs data acquisition to context aware risk assessment and access control, and everything in between, in real-time. This enabled us to show the real challenges of deploying artificial intelligence (AI) based authentication methodologies in practice. CASTRA is currently in the process of being incorporated into one of our mobile based physical access control system (i.e., digital locks) to enable keyless and secure access of users to facilities and properties, which is the target audience of this approach. Integrating CASTRA with digital locks will allow device to automatically infer user intent ("open the door lock") and initiate communication with the end systems on identifying the user with high confidence score, thereby completely eliminating the need for any secondary password or biometrics to access the end systems.

A. CASTRA Current State

CASTRA uses a mobile cloud computing (MCC) architecture, utilizing both Google Android and Amazon Cloud services. The *context classifier* and *information fusion* modules are deployed in the Cloud, providing "*authentication as a service*" which enables one to integrate CASTRA with any mobile Operating System (e.g., IOS, Blackberry), with a minimal or no effort. The **context providers**, *multi-context manager* and the **risk manager** are implemented as Android services and runs in the background thereby providing an "always-on" context-aware authentication and access control service. These client services uses RESTful APIs to send and receive data from the Cloud. The functionality of CASTRA is extensively tested using different variants of the `Samsung` devices (e.g., `S5, S6, Note 4, Note 5`) and several normal and attack scenarios were created to evaluate the performance of CASTRA app and authentication algorithms. Current version of the app ($castra\_v\_0.26$) is available for testing on Android devices[1]. Video demonstration of CASTRA is available via GitHub[2].

IV. Design of Context Analyzer

A. Adversary Model

This work assumes *physical attacks*, where adversary has attained a lost, stolen or shared device protected with conventional or no authentication techniques. In this event, the objective of the attacker is to exploit the weaknesses in current authentication techniques and obtain access to sensitive resources (data, applications) residing in the device. Accessing insecure data from lost, stolen and decommissioned devices are one of the most prevalent mobile threats as indicated by OWASP project [21]. We particularly consider two threat vectors:

[1]The app will be provided upon contacting the primary author

[2]https://github.com/CASTRADHS/CASTRA-DEMO-DHS

Sensor	Feature
Accelerometer (Tri-axial) Gyroscope (Tri-axial)	Mean value
	Standard Deviation
	Sum of absolute deviation from mean
	Binned histograms (ten bins)
	Pairwise correlation between axes
	Mean Log Energy
	RMS value
Magnetometer (Tri-axial)	Mean value
	Standard Deviation
	Sum of absolute deviation from mean
Proximity Sensor	Binned histograms (two bins)

Table I: Extracted features from sensor streams.

(a) *Lost/stolen attacks*: Adversary attains lost/stolen devices and access sensitive resources. This may require adversary to spend ample resources to exploit weaknesses in conventional authentication schemes and gain unsanctioned access.

(b) *Shared/Insider attacks*: This attack assumes a mobile device shared among family members or coworkers and adversary uses escalated privileges to obtain access to owners sensitive resources (e.g., banking or corporate apps) [1], [16].

B. Multi-context manager

The *multi-context manager* gathers various sensor, contextual states, event and configuration data from the mobile device to feed the *context classifier*, *information fusion* and **risk manager** modules and are described as follows:

- *Sensor*: The data acquired directly from the device hardware such as accelerometer, gyroscope, magnetometer, and proximity, at a user specified interval (e.g., the interval can be configured between 10Hz and 100Hz via CASTRA android app). We use `Google Activity Recognition API` to automatically label each time series (sensor stream) with the detected user activity type and the confidence score (e.g., `WALKING, STILL, IN-VEHICLE`). We specifically use a offline version of `API`, which does not rely on any Internet connections and produces the labels for sensor input. The acquired sensor data is further used by the *context classifier* to learn the *proximity* and *gait* models. An example raw time series sensor stream include the following fields: <*Timestamp* | *accelerometer (X,Y,Z)* | *gyroscope (X,Y,Z)* | *magnetometer (X,Y,Z)* | *proximity* | *Detected Activity with confidence score*>.
- *Context*: Information pertaining to the device and the environment, which includes user location and state of the running applications (e.g., active, inactive). While GPS data is used to learn user locations and his mobility patterns, the state of the running applications are used by **risk manager** to execute contextual policies for apps. We use `Google Play Location API` to obtain user's GPS location (e.g., latitude, longitude, GPS accuracy,

and timestamp) at a user-specified time interval (e.g., the interval can be configured between 1 and 10 minutes via CASTRA android app).

- *Event*: Various events related to sensor polling, communication errors, user authentication history (e.g., successful or failed authentication attempts) and state of the device (e.g., device in use or not) are captured. The authentication history and the device states are used by *Information fusion* module to compute the trust score.
- *Configuration*: User options such as trusted location and sensitivity level of the applications set by the user during CASTRA initialization period. This information is used by **risk manager** to enable context aware access to the applications.

C. Context classifier

This section describes the design of behavioral biometric signals that includes everything from feature extraction to design, testing and validation of data-driven and information fusion algorithms. Behavioral biometric signals identify user by measuring the abnormality of users current behavior (i.e., current observed sensor features) with his past recurrent behavior. This phase is called the *prediction* or *online* phase [13]. Users past behavior is learned by applying machine learning algorithms on the features extracted from the raw sensor and location data gathered from the mobile device. CASTRA learns three behavioral biometric signals: (a) *location and human mobility patterns* from GPS traces; (b) *gait patterns* from tri-axial accelerometer and gyroscope; and (c) *proximity patterns* from proximity, tri-axial accelerometer, gyroscope and magnetometer.

1) Feature Extractor: The dataset acquired from the smart phones will be subjected to various sensor noises. It is important to choose the right set of features from these noisy datasets that "uniquely describes the behavior" for the success of the data-driven algorithms. Let $X^{m \times n}$ denote the multi-modal time series data consisting of m sensor streams, each with n attributes (e.g., tri-axial accelerometer, gyroscope, magnetometer). We use the feature extraction algorithm outlined in Algorithm 1 to derive the features from $X^{m \times n}$ to train and learn the *gait* and *proximity* models. As shown in Algorithm 1, we will use rolling (sliding) windows with an overlap (γ) for extracting various statistical features from time-series data segments (see Table I). The number of samples within a window (d) depends on the sensor sampling frequency (s_r).

2) Learning location and mobility patterns from GPS: GPS data provide a rich source of high fidelity information which is unique to different people. Typically "the most frequent places visited by a user", "the time when a particular place is visited", and "the transition between different places " carries reliable user behavioral signature. CASTRA uses historical GPS data to automatically identify significant locations for the user. On top of these learned location patterns, the joint correspondence of which location is most likely visited during what time of the day and the transition probabilities from one location to another are learned. As there is significant change in user

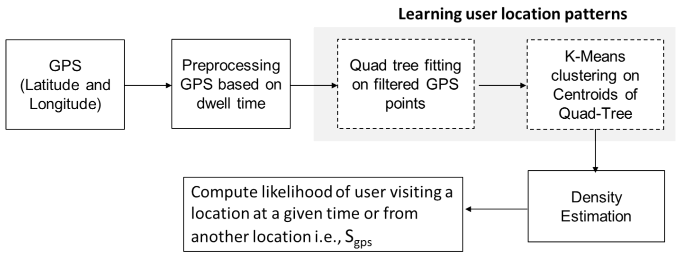

Learning user location patterns
GPS (Latitude and Longitude)
Preprocessing GPS based on dwell time
Quad tree fitting on filtered GPS points
K-Means clustering on Centroids of Quad-Tree
Density Estimation
Compute likelihood of user visiting a location at a given time or from another location i.e., S_{gps}

Figure 2: Main steps involved in learning user scores from from GPS history

Algorithm 1: Multi-modal feature extraction from time series sensor data

```
Input  : $X^{m \times n}$: $n$ sensor streams obtained during $m$
         seconds, $t_1, t_2 \cdots t_m$
Output: Extracted features: $X^{(r \times \phi(n))}$ (walking) or
         $X^{(k \times \phi(n))}$ (proximity)
Define : $\gamma$ (overlap window), $d$ (segment size), $s_r$
         (sensor sampling rate), $\tau_{walk}$ (walking
         confidence threshold), $W$ (sliding window size),
         $\tau_{still}$ (still confidence threshold)
  function PREPROCESS($X^{m \times n}$)
  if WALKING then
      if walk_threshold $> \tau_{walk}$ then
          extract ($X^{\hat{m} \times n}$) WALKING samples from $X^{m \times n}$
          goto function COMPUTE-WALKING($X^{\hat{m} \times n}$)
      else
          return -1;
  else
      goto function COMPUTE-STILL($X^{m \times n}$)
  end function
  function COMPUTE-WALKING($X^{\hat{m} \times n}$)
      segment $\hat{m}$ samples into $r$ segments consisting of $d$ samples, where $d = W.s_r$
      and $r = \begin{cases} \lfloor \frac{\hat{m}}{d} \rfloor + \lfloor \hat{m}.\gamma \rfloor - 1, & \text{if } \hat{m} = even. \\ \lfloor \frac{\hat{m}}{d} \rfloor + \lfloor \hat{m}.\gamma \rfloor, & \text{if } \hat{m} = odd \end{cases}$
      for each segment $s \in r$
      compute $\phi(n) = f_1, f_2, \cdots, f_{92}$
  return $X^{(r \times \phi(n))}$
  end function
  function COMPUTE-STILL($X^{m \times n}$)
      if still_threshold $> \tau_{still}$ then
          extract ($X^{\tilde{m} \times n}$) STILL samples from $X^{m \times n}$
          segment $\tilde{m}$ samples into $k \times d$ segments
          for each segment $s \in k$
          compute $\phi(n) = f_1, f_2, \cdots, f_{103}$
          return $X^{(k \times \phi(n))}$;
      else
          return (-1)
end function
```

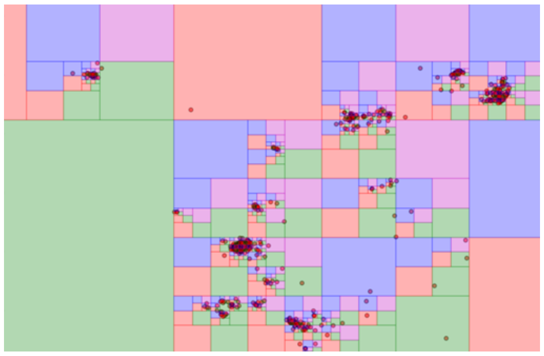


Figure 3: Illustration of the hierarchical partitioning of GPS location history using quad-trees

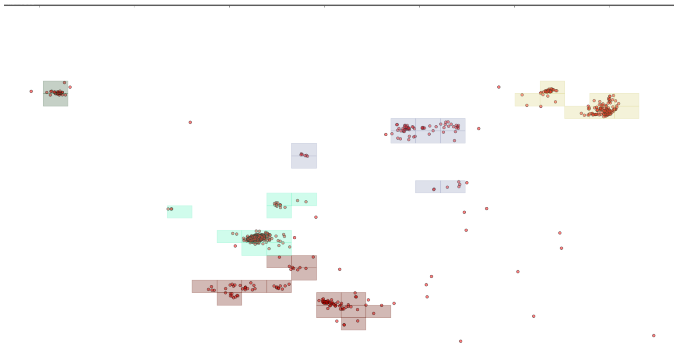


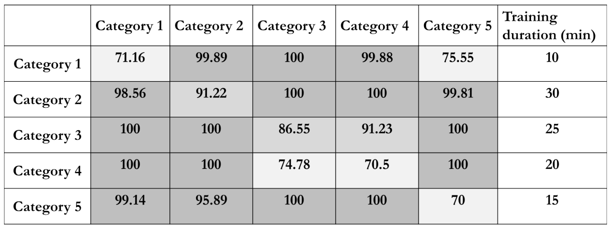
Figure 4: Clustering of leaves into semantic locations at $30m$ resolution

location behavior during a weekday vs weekend, a separate location profile is learned for weekdays and weekends.

Before delving into the temporal and transition model, we first illustrate various steps involved in identifying significant locations from historic GPS traces. Figure 2 provides an overview of the various steps involved in the learning process. The GPS-log file is structured to store latitude and longitude information along with standardized time stamps at a preset frequency. The first step is to filter the log file to disregard data points which correspond to transitory motion and are result of potential errors. This is done by filtering based on dwell time, which disregards the data point, if the user spends less than a threshold time at a particular location. The filtered GPS points are further used to fit a quad-tree data structure. We implement a modified quad-tree data structure such that the dimensions of the leaf nodes can be controlled to a preset accuracy (usually set at $10m$). Figure 3 illustrates the quad-tree data structure based clustering of GPS locations on a public smartphone dataset. A quad-tree based clustering has several benefits. This approach can be used in an incremental fashion and hence can reduce the amount of data which needs to be potentially stored. Another benefit comes from its robustness to GPS errors. The next step involves grouping the leaves into higher order clusters which correspond to locations with semantic meaning like work, home etc. We use the K-means clustering algorithm [13] on the centroids of the leaves of the quad-tree, which helped us to eliminate the limitations associated with traditional K-means clustering such as selection of initial centroids. Figure 4 shows the grouping of the leaves into various clusters which correspond to the semantic locations. In the figure, different colors correspond to different identified locations. Once the locations are clustered, the density of GPS points in the semantic locations are used to compute the likelihood model of a user visiting a certain location i.e., $P(location|user)$, which in turn is the confidence score of the user at a certain location S_{gps}.

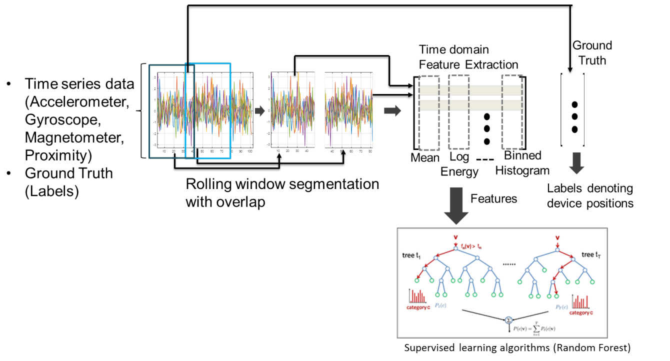
A pure location based approach can be drastically improved by considering temporal historical information. In the next stage we learn a model which is indexed by time, where 'time' abstractly refers to weekday vs weekend and a discretized time interval in each day (e.g., user i seen at location L from time 9:00 AM to 12:00 PM). One of the drawback of learning this joint model is the need for more training data from the user. A further refinement is possible by learning a separate model for each day of the week. This model could account for variability in the schedule for each day, nonetheless, will incur an even more cumbersome data requirement. While learning the joint model $P(location|(time, user))$, we discretized each day into one hour intervals and the probabilistic model learns the location distribution for the user for each hour interval in the day. We further refine our models by taking into account the transition patterns of the user, by learning a joint correspondence between the location clusters (i.e., the probability of moving from location L to location K). This is achieved by computing the number of transitions made between each location cluster, at a given time of the day, and is determined by $S_{gps} = P(location_i|location_j, time)$.

3) Learning gait *patterns from mobile sensor data:* The walking pattern (gait) of a user is an important and unique biometric marker that can be captured using the accelerometer and gyroscope sensors. Most of the efforts on gait based identification is based on supervised learning. In a supervised setting, our task is to learn a function/model from labeled training data i.e., training set consisting of data from multiple users. The function so learned will be able to classify or predict the user given the new test data. In practice, such a setting is not applicable as adversarial gait data is hard to achieve for training. One of the main challenges for gait based authentication purpose is to use unsupervised modeling techniques that learns a nominal model of a user's gait pattern based on the training data and further use the learned model to flag anomalous gait signatures of an adversary (i.e., any user different from the normal user). A significant challenge here is that in order to learn the gait based user recognition, we need a mechanism to identify durations where "walking" activity was being performed by the user. Most of the existing efforts either require user to input the ground truth or use a controlled lab setting where user is instructed to walk for certain duration. We use `Google Activity Recognition API` for automatically annotating the sensor streams with the user activity. As `Google Activity Recognition`



	Category 1	Category 2	Category 3	Category 4	Category 5	Training duration (min)
Category 1	71.16	99.89	100	99.88	75.55	10
Category 2	98.56	91.22	100	100	99.81	30
Category 3	100	100	86.55	91.23	100	25
Category 4	100	100	74.78	70.5	100	20
Category 5	99.14	95.89	100	100	70	15

Figure 5: Performance of One-class SVMs evaluated on subjects belonging to five categories. Each cell (i, j), $i, j = 1, 2, \cdots 5$ denote the average detection rate when category i user's *gait* model is evaluated on category j users. For e.g., $i \neq j$ condition implies the scenario where category i user's model is tested on adversarial data from category j users. The training period of each category are also provided.



Time series data (Accelerometer, Gyroscope, Magnetometer, Proximity)
Ground Truth (Labels)
Rolling window segmentation with overlap
Time domain Feature Extraction
Mean
Log Energy
Binned Histogram
Ground Truth
Labels denoting device positions
Features
Supervised learning algorithms (Random Forest)

Figure 6: Supervised learning pipeline for proximity model

`API` can identify a range of activities such as `STILL`, `IN-VEHICLE`, `WALKING`, during training and validation of models, we use the API to isolate time durations where "walking" activity was being performed with high confidence score ($> \tau_{walk}$). The extracted samples are further used by *context classifier* for learning and evaluation of gait based user recognition model.

As illustrated in Algorithm 1, we approach the gait learning process by extracting 92 statistical features from accelerometer and gyroscope in a moving window of duration $W = 10$ seconds with $\gamma = 50\%$ overlap. The set of features extracted are presented in Table 1. Once the features are computed, we use a one-class SVM with RBF kernel to model the nominal behavior [13]. Figure 5 summarizes the performance of the learned model for selected five categories of user. Each category i is represented by the amount of training data contributed by users within that group ($5-6$ users within a group). For instance, category 1 represents the users trained for less than 10 minutes. All the training data from users were collected in an uncontrolled fashion to account for the normal variability in gait signatures in real life scenarios. From our results, we observe that one could learn an accurate gait-based user recognition model using at least $20-30$ minutes of walking data with confidence scores $\geq 75\%$. Since the number of observations (training samples) exceeds the number of features, our classifier will have low variance and will perform well on test observations [13]. In our analysis, we observe that when adequate training data are available, one could detect an adversary based on gait with 99%. On the other hand, the model incurs a false alarm rate of 10% - 15%, which can be significantly reduced by using larger training periods and optimal hyper parameter tuning. CASTRA's reliance on the use of multiple factors to authenticate a user can be used to amortize the cost of false alarm rate.

4) Learning proximity *patterns from mobile sensor data:* In this section we discuss how proximity information can be used to extend the validity of the initial biometric authentication or high confidence score and reduce the need to re-authenticate using password or challenge questions every time a sensitive application is accessed. This module comes into action after we have achieved a high confidence in the users identity via active factor biometrics or due to the fusion of multiple authentication factors. After observing a high confidence score, we detect if the device is in close contact with the person based on sensor signatures and this observation in turn is used to extend the validity of the prior high confidence authentication until a user-specified timeout interval or when the device loses contact with the user.

CASTRA uses a supervised machine learning approach [13] to learn a *proximity* classifier which outputs the proximity of the device i.e., if the device is in contact with the person or not. We formulated experiments to collect data from fifteen subjects in a wide variety of scenarios. Users were instructed through voice and text over the course of two hours to alternatively put the device "on table" and "in pocket" at various standing and sitting stations. During this time data from accelerometer, gyroscope, magnetometer and proximity sensors were collected at 50Hz. Figure 6 illustrates the various steps involved in the supervised learning approach. In the first step the multi-dimensional time series is segmented into frames using a moving window of duration $W = 15$s, with $\gamma = 50\%$ overlap. As described in Algorithm 1, each frame is passed through a feature extraction routine which extracts various time domain features (see Table I for the list of features used). To learn *proximity* model, we extracted 103 features from each segment. Finally, these features were used along with the tagged device position label from the experiment to train a binary classifier. We used several supervised learning algorithms such as SVMs, AdaBoost and Random Forest trees to learn the *proximity* classifier. Random Forest trees with 500 estimators were selected on the basis of superior performance for the classification task. Table II shows the performance results on a test dataset. As *proximity* models are user-agnostic, these are pre-trained and deployed in the *Identity and Behavioral Models* database and are available to use when CASTRA is installed.

D. Information fusion

In previous sections, we discussed about various data-driven based models that will be used to flag an anomalous behavior

	Device on table	Device in pocket
Device on table	90.6%	9.4%
Device in pocket	8.6%	91.4%

Table II: Confusion matrix on test dataset for proximity detection.

(example behaviors include abnormal walking, device seen in unseen location, device placed on the table etc). The output of these models (anomaly scores) will be fused at every time t to generate a dynamic trust score. This section will briefly describe our approach to fuse the anomaly scores computed by various models. Algorithm 2 presents the pseudo-code of the dynamic trust score computation algorithm. Let $S_{bio}, S_{gps}, S_{gait}$ denote the anomaly scores obtained from the *proximity, location and gait* models, respectively. Let the fused dynamic trust score score be S_{fuse} which is computed by taking a weighted linear combination of the individual scores:

$$S_{fuse} = w_b S_{bio} + w_{wlk} S_{gait} + w_g S_{gps}$$

where $w_b + w_g + w_{wlk} = 1$ and the individual weights w_b, w_g, w_{wlk} are chosen to minimize the sum of false alarm and missed detection probabilities (see subsection below on the configuration of weights and optimal thresholds).

Let $x_t = (in\_use, in\_contact, wlk\_st, t_{no\_contact}, S_{bio}, S_{gps}, S_{gait})$ denote a state vector at time t. in_use is a binary variable denoting if the device is idle (0) or currently being used (1). $in\_contact$ and wlk_st are binary variables denoting if the device is in contact with the user and if walking activity is detected, respectively. $t_{no\_contact}$ records the time elapsed since the device lost contact with the user. Let us also denote the history of state vectors as $\bar{x}_t = (x_0, \ldots, x_t)$ The first stage in trust score computation is to process sensor signatures over a time duration $(t - \delta, t)$ and estimate the state vector $\hat{x}_t$. Once the state estimate is available, the final fused score is computed $S_{fuse} = F(\bar{x}_t)$. We now briefly outline the steps involved in state estimation. The binary variable $in\_contact$ is estimated using the `Google Play Services ActivityRecognition API` and the learned proximity model (`on table` or `in pocket`). At a preset frequency, the `Google ActivityRecognition API` is used to detect if the phone is `STILL` with high confidence. During this time interval, the Random Forest based proximity model is used to detect if the phone is in contact with the user or not. The scores S_{gps} and S_{gait} are computed using the *gait* and *location* models respectively. For S_{gait}, given a new set of features extracted from the raw accelerometer and gyroscope data, we compute the distance of features from the SVM decision boundary and then use a sigmoid transformation $\frac{1}{1+\exp^{-\lambda x}}$ to generate a probability score. This probability score is then transformed into S_{gait} such that a negative score corresponds to an anomalous user and a positive score corresponds to nominal user.

In instances where wlk_st is not detected (i.e., $wlk\_st = 0$), we set S_{gait} according to the following rules:

(i) At time t, $wlk\_st = 0$, $in\_contact = 1$ and in time period $t-1$, if a nominal gait score (i.e., > 0) was detected,then $S_{gait}(t)$ is computed by applying a discount factor on the gait score observed at time $t-1$.

$$S_{gait}(t) = \alpha_1 S_{gait}(t-1), \text{ where } \alpha_1 \in (0,1)$$

(ii) If the device is not in contact at time t (i.e., $in\_contact = 0$) then S_{gait} is computed as:

$$S_{gait}(t) = \alpha_2 S_{gait}(t-1), \text{ where } \alpha_2 \in (0,1) \text{ and } \alpha_1 > \alpha_2$$

(iii) A walking gait is not detected and at the previous instance an anomalous gait was flagged, then the anomalous score is maintained $S_{gait}(t) = S_{gait}(t-1)$. The proximity or biometric score S_{bio} is set to 1 whenever the user authenticates using an active factor. The S_{bio} is held high (score of one) for a user-specified timeout interval ($\approx$ one to two minutes) and then the S_{bio} will gradually decay to a lower value. The decay rate is governed by the state $in\_contact$. For e.g., the decay rate for S_{bio} is lower when the device is in contact ($in\_contact = 1$) as opposed to the state when the device is not in contact ($in\_contact = 0$).

Configuration of optimal thresholds: In this section, we discuss how to compute the optimal weights (w_b, w_g and w_{wlk}) and thresholds (τ_l) that minimize the sum of false alarm and missed detection probabilities.

False alarm probability: A probability of false alarm occurs when the information fusion gives low trust score i.e., $\bar{s}_t = L$ but user is authentic in fact. Let us denote adversary and normal user by A_u and N_u respectively. Thus, probability of false alarm, at a given time t, is given by:

$$P_{FA} = P(S_{fuse} \leq \tau_l \mid N_u)$$

Substituting S_{fuse} with $w_b S_{bio} + w_{wlk} S_{gait} + w_g S_{gps}$ and using Boole's inequality [22], we can rewrite the above equation as:

$$\begin{aligned} P_{FA} &= P(w_b S_{bio} + w_{wlk} S_{gait} + w_g S_{gps} \leq \tau_l \mid N_u) \\ &\leq P(w_b S_{bio} < \tau_b \mid N_u) + P(w_{wlk} S_{gait} < \tau_w \mid N_u) \\ &\quad + P(w_g S_{gps} < \tau_g \mid N_u) \text{ (using Boole's equality)} \end{aligned}$$

where $\tau_b + \tau_w + \tau_g = \tau_l$. Each probability term $P(.)$ denote the event of observing a low score for a normal user i.e., a low biometric score, a low GPS score and a low gait score. We use Markov's inequality ($P[X \geq a] \leq \frac{E[X]}{a}$) to derive the bounds for each probability term and P_{FA} is given by:

$$\begin{aligned} P_{FA} &\leq 3 - \left(\frac{E[w_b S_{bio}]}{\tau_b} + \frac{E[w_{wlk} S_{gait}]}{\tau_w} + \frac{E[w_g S_{gps}]}{\tau_g}\right) \\ &\leq 3 - \left(w_b \frac{E[S_{bio}]}{\tau_b} + w_{wlk} \frac{E[S_{gait}]}{\tau_w} + w_g \frac{E[S_{gps}]}{\tau_g}\right) \end{aligned}$$

where $E[x]$ represents conditional expected value of each authentication factor x, i.e., $E[x \mid N_u]$.

Missed detection probability: A probability of missed detection occurs when the *information fusion* module gives high or medium trust score but the device is with the adversary in fact. We denote the probability of missed detection as P_{MD} and is given by:

$$P_{MD} = P(S_{fuse} > \tau_l \mid A_u)$$

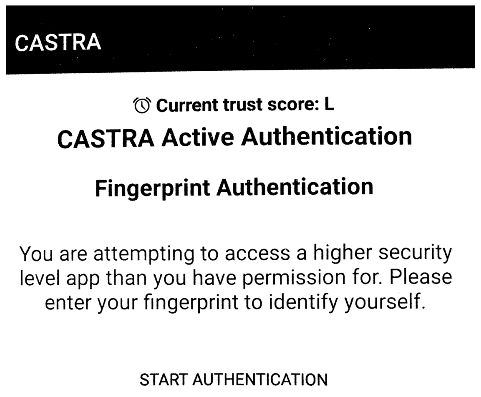
Substituting for S_{fuse} and using Boole's equality and Markov inequality, we can rewrite the above equation as:

$$\begin{aligned} P_{MD} &= P(w_b S_{bio} + w_{wlk} S_{gait} + w_g S_{gps} > \tau_l \mid A_u) \\ &\leq P(w_b S_{bio} > \tau_b \mid A_u) + P(w_{wlk} S_{gait} > \tau_w \mid A_u) \\ &+ P(w_g S_{gps} > \tau_g \mid A_u) \text{ (using Boole's inequality)} \end{aligned}$$

$$P_{MD} \leq (w_b \frac{E[S_{bio}]}{\tau_b} + w_{wlk} \frac{E[S_{gait}]}{\tau_w} + w_g \frac{E[S_{gps}]}{\tau_g})$$

where $E[x]$ represents conditional expected value of each authentication factor x, i.e., $E[x \mid A_u]$.
Based on the above equations, we observe that with an increasing threshold τ_l, the missed detection probability decreases while the false alarm probability increases. In fact, the summation of the false alarm and missed detection probabilities is a convex function of the threshold τ_l, i.e., τ_g, τ_w, τ_b. With the convexity, the optimal thresholds $\tau_g^*, \tau_b^*, \tau_w^*$ can be computed by minimizing the summation of P_{MD} and P_{FA} probabilities according to

$$\frac{d}{d\tau_b}(P_{FA} + P_{MD}) \mid_{\tau_b = \tau_b^*} = 0$$
$$\frac{d}{d\tau_w}(P_{FA} + P_{MD}) \mid_{\tau_w = \tau_w^*} = 0$$
$$\frac{d}{d\tau_g}(P_{FA} + P_{MD}) \mid_{\tau_g = \tau_g^*} = 0$$


CASTRA

Current trust score: L

CASTRA Active Authentication

Fingerprint Authentication

You are attempting to access a higher security level app than you have permission for. Please enter your fingerprint to identify yourself.

START AUTHENTICATION

Figure 7: CASTRA *Active factor* handler: requesting user to input active biometrics to access a H sensitive application.

Algorithm 2: Dynamic Trust Score Computation

```
Input  : $sen_{t-\delta:t}$ (sensor values during time interval
         $t-\delta, t$),
         $\bar{s}_{t^-} := S_{fuse}(t-1)$ (previous fused score),
         $Prox\_mdl$, $Gait\_mdl$, $GPS\_mdl$
         (learned machine learning models)
Output: Current score $\bar{s}_t$
function ESTIMATE STATE($sen_{t-\delta:t}$,$\bar{s}_{t^-}$)
    in_use $\in \{0,1\} \leftarrow$ Device in use
    (wlk_st, STILL)$\in (0,1) \leftarrow$ Google ActivityRecognition API
    if $in\_use = 0$ then
        if $wlk\_st = 1$ then
            in_contact $\leftarrow 1$
            $S_{gait}(t) \leftarrow Gait\_mdl$
        else if $STILL = 1$ then
            $in\_contact \leftarrow Prox\_mdl$
        else
            $in\_contact \leftarrow 1$
        if $S_{gait}(t^-) > 0$ then
            $S_{gait}(t) = \alpha * S_{gait}(t^-)$   ▷ varying discount factors depending on in contact
        else
            $S_{gait}(t) = S_{gait}(t^-)$
        if $in\_contact = 1$ then
            $S_{bio}(t) = \beta_c S_{bio}(t^-)$   ▷ $\beta_c$ denotes the discount factor in contact
        else
            $S_{bio}(t) = \beta_{nc} S_{bio}(t^-)$
        ▷ $\beta_{nc}$ denotes the discount factor not in contact
    else
        $\bar{s}_t = \bar{s}_{t^-}$
end function
function FUSE SCORE($\bar{s}_t, w_g, w_b, w_{wlk}$)
    $S_{fuse}(t) = w_g * S_{gps}(t) + w_b * S_{bio}(t) + w_{wlk} * S_{gait}(t)$
    return $\bar{s}_t = \begin{cases} H, & S_{fuse} \geq \tau_h \\ M, & \tau_l < S_{fuse} < \tau_h \\ L, & S_{fuse} \leq \tau_l \end{cases}$
end function
```

V. RISK MANAGER

The inputs to the **risk manager** compasses various contexts around the device which includes user trust score (H, M, L), state of the apps (active or inactive), and the current location of the user (is the user in trusted zone which in turn is chosen by the user during the CASTRA initialization stage). Two major policies are implemented by CASTRA and are called the *trust-aware* and *context-aware* policies. As the name suggests, *trust-aware policies* compares the trust score reported from **context analyzer** with the application sensitivity level (set by the user via CASTRA app or by the system admin via Cloud during CASTRA initialization stage) to determine whether the user is trustworthy to access the application. For instance, under this specific policy, user with trust score L will be denied access to M and H sensitive applications and will be allowed access to merely low sensitive applications (see Figure 7 to see how CASTRA *Active factor* handler requests user to prove

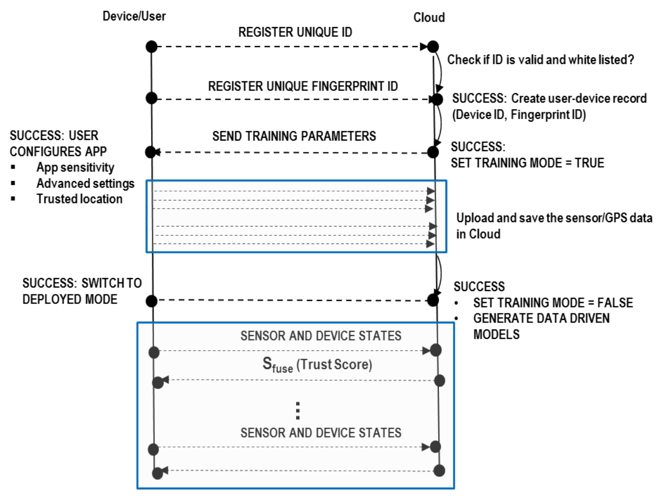

Device/User
Cloud
REGISTER UNIQUE ID
Check if ID is valid and white listed?
REGISTER UNIQUE FINGERPRINT ID
SUCCESS: Create user-device record (Device ID, Fingerprint ID)
SEND TRAINING PARAMETERS
SUCCESS: USER CONFIGURES APP
App sensitivity
Advanced settings
Trusted location
SUCCESS: SET TRAINING MODE = TRUE
Upload and save the sensor/GPS data in Cloud
SUCCESS: SWITCH TO DEPLOYED MODE
SUCCESS
SET TRAINING MODE = FALSE
GENERATE DATA DRIVEN MODELS
SENSOR AND DEVICE STATES
S_fuse (Trust Score)
SENSOR AND DEVICE STATES

Figure 8: Overview of CASTRA message sequences

his ownership in order to access H sensitive application).

On the other hand, *context-aware policies* ensure that an application can be accessed only under safe contexts (defined by location, time of the day, presence of untrusted apps etc.). The **risk manager** is also designed to continuously monitor for contextual changes (e.g., user moving to an unsafe location or changes of user confidence score) and perform the actions suggested by the policies (e.g., perform biometrics, killing applications etc.) on detection of such contextual changes.

VI. CASTRA Implementation

A. CASTRA Initialization

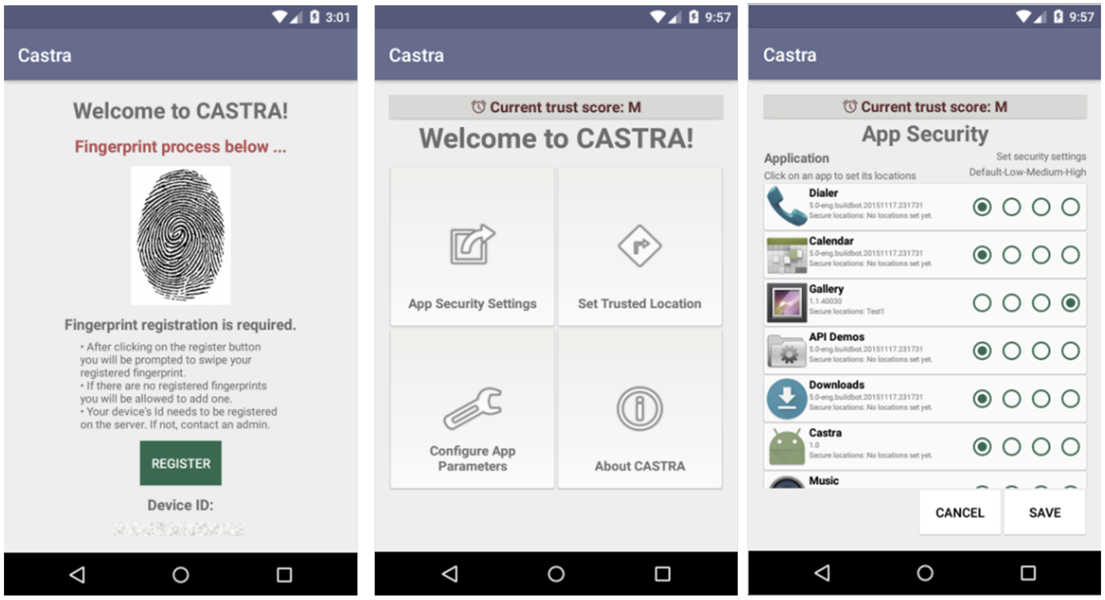
Figure 8 provides an overview of communication between the device and the Cloud and Figure 9 presents CASTRA application. We use a two-step approach to securely enable CASTRA authentication and access control services: (i) Enrolls user device with the Cloud by utilizing a white-list of device identifiers; (ii) Register user fingerprint (hash of user fingerprint) with the Cloud, once the device is validated and authenticated. On successful registration, the device is set to training mode and various configuration and training parameters (e.g., training time) are pushed to the end device. During this mode, user can further configure CASTRA app which includes `app sensitivity, advanced settings` (configuration parameters for sending and training CASTRA algorithms e.g., server upload frequency, sliding window, walk and still thresholds, sensor sampling rate, GPS radius etc) and `trusted locations` (e.g., user can manually choose trusted locations where **risk manager** constantly assumes H trust score at these locations and no sensors are polled to mitigate the battery drain and bandwidth issues). The `app sensitivity level` and `trusted locations` are further used by **risk manager** to provide *trust-aware* and *context-aware* access control decisions. During the training period, the device also uploads various user data (sensor data, GPS, error logs etc) to the Cloud. To amortize the cost of data rate consumption, CASTRA uploads training data when Wi-Fi connections are available. At the worst case, CASTRA can securely store one week worth of user data on the device. Automated agents are deployed at the Cloud that switches user to deployment phase when training period ends and also adequate data from the user is received at the Cloud. At every $t_a = 5$ minutes, the automated agents will monitor for completed devices, set training mode to `false` for the trained devices and generate user-specific *gait* and *location* models. During the deployment phase, the *multi-context manager* pulls several data from **context providers** and invokes *information fusion* with the user information. This is performed at every server upload frequency interval set by the user on the app (default interval is 1 min). As frequent communication with the Cloud impacts battery and bandwidth, we implemented a context-aware invocation of Cloud Services, where the *information fusion* module is invoked only at specific contexts i.e., when user enters a specific geo-fence or when the user is in close proximity of the physical device of interest. Fig 9 presents the CASTRA app upon successful installation and registration of user with the Cloud.

B. CASTRA model retraining

We have deployed automated agents to retrain the *Identity and Behavioral Models* when *information fusion* frequently raises false alarms. To deal with such scenarios, CASTRA logs those false alarm events where a L/M trust score is obtained from *information fusion* albeit the user is “genuine" (user is determined as genuine when accurate biometrics are entered by user on request from *Active factor handler*). CASTRA specifically logs the sensor and the location information corresponding to the false predictions. At the end of system-defined time period (24 hour period), the logged data is pushed to the Cloud where the *IdBM* models are retrained and deployed. This in turn helps CASTRA to adapt to new sensor and location data.

C. CASTRA Services

CASTRA employs a server-centric architecture and comprises of two key modules: CASTRA client services and CASTRA Cloud services. While the CASTRA client is implemented on Android, the CASTRA Cloud services are implemented using the Amazon Cloud Computing platform.
CASTRA client services: CASTRA client is implemented on the application layer of the Android Software Stack. When CASTRA application is launched and the device owner is successfully registered with the Cloud, various Android services are initialized at the device including the *multi-context manager*, **context analyzer** and **risk manager** services. These services operate in the background of the device and uses an `Alarm Manager` to monitor these services every 5 minutes in order to relaunch the services in the event of the Android OS killing them. The services are also called “sticky” which will request the Android OS to restart them if stopped. The *multi-context manager* service is responsible for obtaining user specific data from various device sensors (**context providers**) and runs in a separate process from the CASTRA Application.


3:01

Castra

Welcome to CASTRA!

Fingerprint process below ...

Fingerprint registration is required.

- After clicking on the register button you will be prompted to swipe your registered fingerprint.
- If there are no registered fingerprints you will be allowed to add one.
- Your device's Id needs to be registered on the server. If not, contact an admin.

REGISTER

Device ID:

[redacted]

9:57

Castra

Current trust score: M

Welcome to CASTRA!

App Security Settings | Set Trusted Location | Configure App Parameters | About CASTRA

9:57

Castra

Current trust score: M

App Security

Application — Click on an app to set its locations | Set security settings Default-Low-Medium-High

Application	Default	Low	Medium	High
Dialer 5.0-eng.buildbot.20151117.231731 Secure locations: No locations set yet.	●	○	○	○
Calendar 5.0-eng.buildbot.20151117.231731 Secure locations: No locations set yet.	●	○	○	○
Gallery 1.1.40030 Secure locations: Test1	○	○	○	●
API Demos 5.0-eng.buildbot.20151117.231731 Secure locations: No locations set yet.	●	○	○	○
Downloads 5.0-eng.buildbot.20151117.231731 Secure locations: No locations set yet.	●	○	○	○
Castra 1.0 Secure locations: No locations set yet.	●	○	○	○
Music				

CANCEL SAVE

Figure 9: CASTRA application: The three graphics display CASTRA registration page, CASTRA main page and app security settings, respectively.

It uses various android schedulers to continuously acquire the data from the underlying hardware sensors and GPS units. The *multi-context manager* also uses the user activity recognition service which is implemented as an intent service that obtains "user activity" from the Google API and broadcasts the information to *multi-context manager* whenever the value changes. Similarly, **context analyzer** service also relies on android schedulers to continuously run feature extractions and further store those features on the device SQLite database. The **risk manager** service is an android service that runs in a separate process and is responsible for running and looping in the background to prevent a user from accessing the applications on the Android device when the context around the device is untrusted or unsafe. In our implementations, **risk manager** verifies the context around the device and of the user every 5 seconds. When contexts are detected to be unsafe or policy rules are violated, it either kills the application or launches fingerprint verification screen requesting the user to input his biometrics to boost the trust score. One of the limitations of **risk manager** is the inability to control access to the system level applications as CASTRA Client runs at the application layer of the software stack. CASTRA is designed with modularity in mind and one can easily integrate CASTRA Client with existing access control modules provided by Apex [6], Crepe [7], Kirin [8] etc. The **risk manager** monitors for contextual changes by storing all the running applications on the device and monitoring for changes in the running list of applications or their associated policies.

CASTRA Cloud services: The CASTRA Cloud service is responsible for obtaining the data from CASTRA client application and invoking the **context classifiers** and *information fusion* modules for classification and trust score computation, respectively. The server code that handles the client requests are implemented in Python utilizing the Django framework. The server code is hosted live on Amazon Web Services (AWS) Elastic Beanstalk platform and uses t2.larger instance server with 2 cores and 8GB RAM. We use Relational Database Service (RDS) with a MySQL database, with 5Gbs capacity for acquiring the CASTRA generated user data and the analytical models. Separate databases for logging user generated features, user events (walking, still, biometrics), user trust score, and application policies are implemented. The communication between the client and the Cloud are realized using Resful APIs utilizing HTTPS POST.

Privacy: As various information related to user are passively gathered from the device, privacy is always a concern for behavioral biometrics. This specifically holds true for behavioral solutions based on application usage patterns that collects data from geographic locations, phone calls, text messages, and website URLs. In CASTRA, we address privacy of personal identifiable information (PII) by associating each user/device with an identity randomly derived from the device identity number and the registration key. Alternatively, we represent each user with a randomly generated identity. However, our approach is still vulnerable to statistical inference attacks on location and sensor data history, where attacker exploits sensor data to derive latent user behavior such as health, stress levels [29]. One of our future direction in behavioral biometrics is to enable differential privacy (adding noise to the sensor data) to prevent leaking of user sensitive information.

VII. CASTRA Results

This section discusses various experimental settings used by CASTRA and the run time performance of CASTRA under normal and adversarial settings.

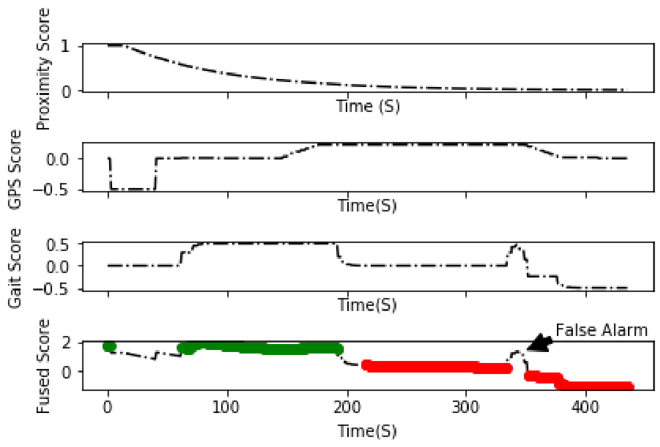


Figure 10: Stolen attack: Illustration of S_{fuse} and each authentication factor ($S_{bio}, S_{gait}, S_{gps}$) under stolen attack in a trusted location ($S_{gps} > 0.5$). The red and green dots denote low (L) and high (H) fused trust score, respectively. The dotted lines indicate medium (M) trust score. We set the weights such as $w_{wlk} > w_g > w_b$

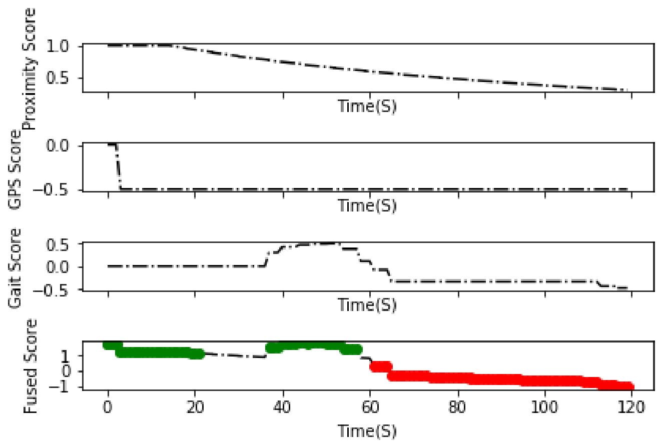


Figure 11: Shared attack: Illustration of S_{fuse} and each authentication factor ($S_{bio}, S_{gait}, S_{gps}$) under shared attack in an untrusted location ($S_{gps} < 0$).

A. Offline Analysis of Context Analyzer Algorithms

To demonstrate the performance of data-driven algorithms, the efficiency of using multiple authentication factors, and correspondingly its ability to lock sensitive resources in the event of detecting unauthorized access, we constructed several attack scenarios involving stolen, lost and shared threats. Our prior work [16] demonstrates several attack and normal scenarios and in this paper, we illustrate two new attack scenarios, *stolen attack in a trusted location* and a *shared (or insider) attack in an untrusted location*. While we detect *stolen* attack using both *proximity* and *gait* models, the *shared* attack is detected using *gait* and *location* models. In both cases, unauthorized access to device was detected in less than 10 seconds, when data-driven modules and correspondingly, the *information fusion* module are invoked every 10 seconds. In our experiments, sensors were sampled every 50Hz with $\gamma = 50\%$ and GPS locations were polled every 60 seconds. As opposed to sensors, locations rarely change within 60 seconds and using a lower sampling rate (one GPS sample in every minute) enables us to amortize the cost of battery consumption. Video demonstration of the attacks can be accessed in GitHub (see footnote 3).

B. Stolen attack

In reference to Figure 10, we assume a device owner who performs his initial biometrics and places his phone on the pocket ($S_{bio} = 1$). The owner starts at an untrusted location ($S_{gps<0}$) and during the course of experiment, he enter the trusted zone ($S_{gps} > 0$), where the device is stolen and taken away by an adversary. Due to initial biometrics, the trust score for the initial time period (0s, 20s) is H, denoted by green markers. As S_{bio} decreases after 20s due to the $in\_contact = 1$ discounting factor (see Algorithm 2), the trust score drops to M. As he walks towards the trusted location, the score gradually boosts to H which in turn is contributed by escalating S_{gps} and S_{gait} scores. The red dots indicate the scenarios where the owner places the phone on the desk (225s, 325s) (indicated by $S_{gait} = 0$ and later taken away by the adversary (350s, 450s). During the time period (350, 450), S_{gait} drops below 0 due to the detection of adversarial gait. In our experiments, as we used higher weight for gait score ($w_{wlk} > 0.6$), the trust score (S_{fuse}) is very sensitive to the changes in gait signals and hence, can quickly react to anomalous gait patterns. The GPS score remains high, as both adversary and user shares the same location, which in turn shows the downside of using location merely as an authentication factor as in [26]. Device placement techniques such as proximity [1] also fails to detect the attack, as it lacks the ability to identify user-specific behavioral traits.

C. Shared attack

In reference to Figure 11, we assume a device owner who handover the phone to his colleague at time 60s and the adversary (colleague in this case) tries to access sensitive applications. As the experiment is performed in an untrusted zone, S_{gps} is negative ($= -0.5$). Simialr to experiment 1, device owner performs his initial biometrics, and places his phone on the pocket which initially leads to H trust score i.e., due to high proximity score at time (0, 20) seconds. As user walks, gait score gradually increases and boosts the trust score from M to H during the time interval (40s, 59s). At time period 60, the phone is handed over which in turn is recognized by CASTRA through abnormal gait patterns ($S_{gait} < 0$), leading to L trust score. The lower trust score in turn prevents adversary from accessing medium or highly sensitive applications.

D. Experimental Settings

In Table III, we present various thresholds and parameters used for validation and evaluation of CASTRA performance. In particular, we evaluated CASTRA on 15 subjects (age groups from 20 to 40) using several variants of Samsung

Parameters	Values
Number of Subjects	15
Devices	Samsung (S5/6, Note 4/5)
Training time [c]	1 to 10 hours
Upload frequency (1 to 10 mins)	1 min
Overlap % (γ)	10%
Walk threshold (τ_{walk}) ($10\% - 100\%$)	$75\% - 90\%$
Still threshold (τ_{still}) ($10\% - 100\%$)	90%
Sampling rate (s_r) ($10Hz - 100Hz$)	$10Hz$
GPS sampling rate	5 min
Biometric validation ($1 - 10$ min)	1 min
τ_h (High threshold) [c]	≥ 0.8
τ_l (Low threshold) [c]	≤ 0.02
w_{wlk} (Gait weight) [c]	$(0.5, 0.8)$
w_g (GPS weight) [c]	$(0.1, 0.4)$
w_b (Proximity weight) [c]	$(0.1, 0.3)$
β_c (In contact discount) [c]	$(0.5, 0.9)$
β_{nc} (Not contact discount) [c]	$(0.2, 0.3)$

Table III: CASTRA Experimental Settings. [c] indicates the thresholds set via Cloud respectively

devices. We asked the participants to carry the device for multiple days depending on the duration of training period. Once the user is moved to the deployment phase, we designed several scenarios (normal and adversarial) to test its efficiency in terms of latency, computational overhead and detection accuracy. Each of the performance measure is discussed below with results.

E. Algorithm performance

We investigate the performance of the trust score computation algorithm under safe, risky and adversarial settings. Figure 12 presents an overview of the computed trust score (S_{fuse}). In this particular case, user performed the training in one location (Loc 1) (3/29) for two hours and tested its performance in two locations, Loc 1 (3/29) and Loc 2 (4/2). In the graph, we denote Loc 1 as trusted location and Loc 2 as untrusted location (location where the user is not seen before). Due to the lack of adequate location information, caused by limited hours of training, the GPS score for trusted zone ranges between 0.2 and 0.3. On 3/29 from 10:40 through 10:53, user tested in Loc 1 i.e., $S_{gps} = 0.2, S_{gait=0}, S_{bio} = 0$ leading to M trust score. On 4/2 (18:32) user entered his biometrics and realized a H trust score. Since $S_{gps} < 0$ (untrusted location), but the phone is in close proximity with the user after initial biometrics, the trust score drops to M (18:33, 18:41). On placing the device on the table $in\_contact = 0, S_{gps} = -0.5$, the trust score drops to L. At 18:47, when the phone is taken

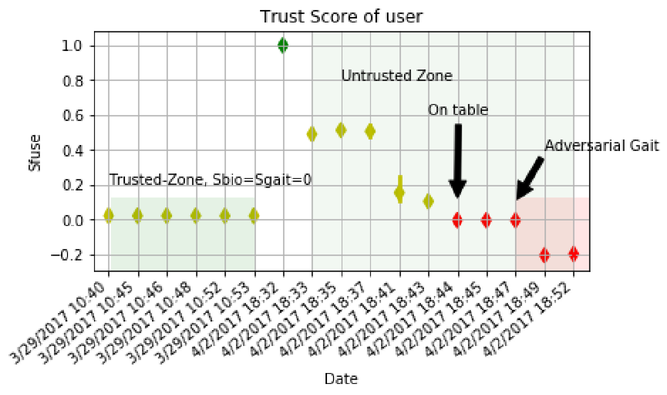


Figure 12: Trust score of user in trusted and untrusted zones after two hours of training. The green, yellow and red dots denote H, M, L trust scores.

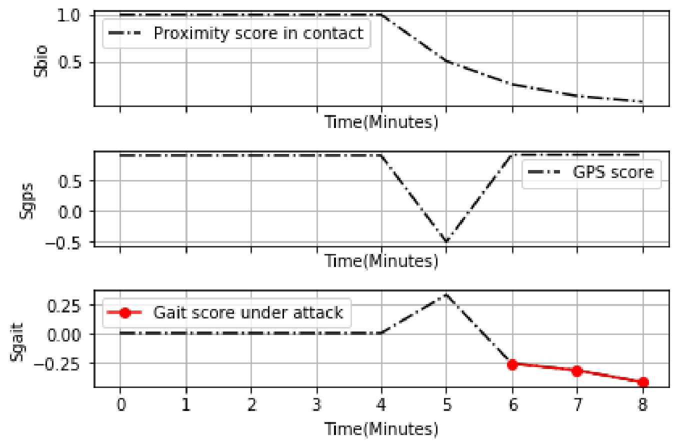
by an unknown, S_{gait} drops to negative which subsequently reduces the trust score to L ($S_{fuse} = -0.2$).

F. Optimal thresholds

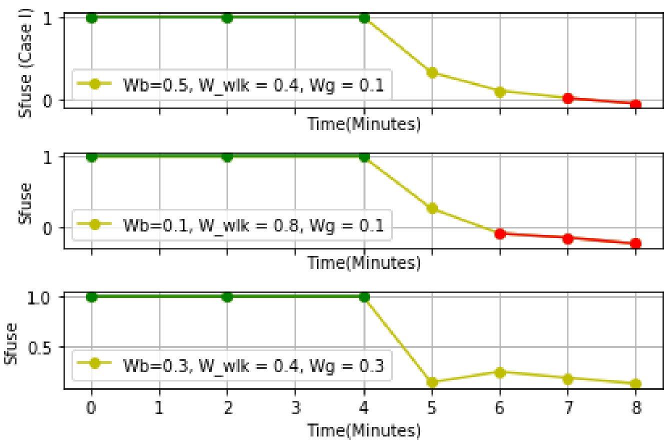
Figures 13a and 13b demonstrate the performance of CASTRA *information fusion* algorithm for various weights under stolen attack, where adversary snatches and walks aways with the device from normal user. As opposed to GPS and proximity, since gait can capture user distinctiveness and correspondingly adversarial gait, leveraging a higher weight for gait (w_{wlk}) helps CASTRA to rapidly detect gait-based attacks. From the fused scores, setting $w_{wlk} = 0.8$ enables CASTRA to detect attack and push the trust score to low rapidly as opposed to cases where $w_{wlk} = 0.4$. On the other hand, since attacks are "rare" events, providing low thresholds for GPS and proximity can affect the user friendliness of the algorithm. Using the analytical results from Section III.D, the optimal thresholds were computed to be $S_{gait} = 0.5, S_{bio} = 0.4$ and $S_{gps} = 0.1$. One may note the thresholds may vary depending on the user (due to $E[.]|N_u$ and $E[.]|A_u$) and deriving a user-specific threshold is one of our future goal. Figure 14 also depicts S_{fuse} when the device is in close contact with the user and when the device is placed on the table. As expected, when the device is in close proximity with the user, S_{fuse} is high. The difference in S_{fuse} in "in-contact" and "on table" is guided by the *in_contact* thresholds β_c and β_{nc} and w_b.

G. Run time performance

Figure 15 presents the runtime performance of CASTRA, averaged across 15 subjects, clearly demonstrating its feasibility to integrate with commercial applications.
Latency: The latency (L) is defined as the sum of *computational latency* (CL) and access latency (AL). CL is defined by the time expended from acquiring the device data to receiving a trust score from the Cloud and is computed as $2T_c + T_p$, where T_c and T_p denote the communication latency from device to Cloud and vice versa and the processing latency. AL is defined by the time expended by the **risk manager** to



(a) Individual Scores of user at a trusted zone.



(b) Fused Scores under various weights for gait, GPS and proximity

Figure 13: Detection latency of *information fusion* depending on various weights for gait, proximity and GPS.

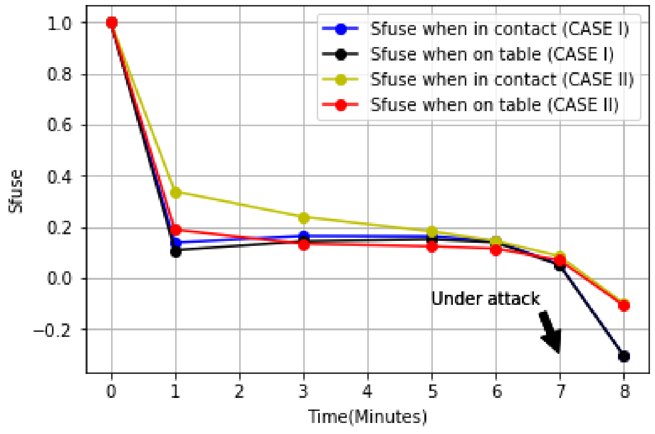


Figure 14: Fused scores when in contact and on table for various thresholds. CASE I and II denote the scenarios with w_b = 0.1 and 0.5, respectively

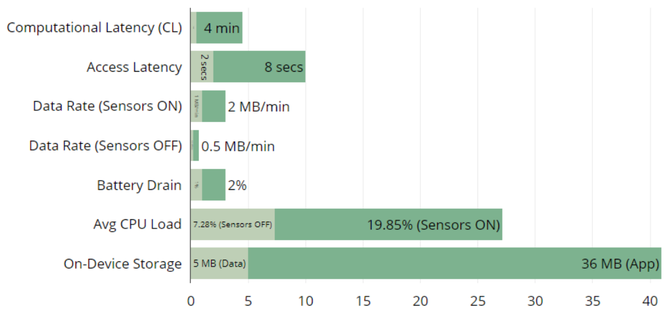


Figure 15: Average runtime performance of CASTRA

grant or deny access to an application based on the detected context. In our experiments, we observed a minimum latency of 0.5/min and a maximum latency of 4/min. This maximum latency is observed in scenarios where users had intermittent connectivity to Cloud, especially, when users are around the outskirts of work and home locations. The access latency is driven by the number of active applications in the background, which in turn forces **risk manager** to retrieve and verify a number of trust-aware and context-aware policies to ensure continuous protection. The access latency is at most 2 seconds when a minimal set of apps are active in the background and a maximum latency of 8 seconds is observed when a larger number of apps are activated.

Data Rate: The data rate (D) is defined as the amount of bandwidth consumed in sending and receiving the data to and from the Cloud. We observed a huge consumption in bandwidth when data from sensors, GPS and client events are continuously polled and sent to the Cloud ($2MB$/min). In the event of switching off the sensors, the data rate consumption is reduced substantially ($> 1/4$), which in turn supports our argument for the need of context-aware invocation of sensors. It is intuitive to see that when sensors are on, the user will be in `STILL` or in `WALKING` mode, then the number of samples sent to the server during every upload frequency $freq$ is given by $r \times freq \times \phi(n)$, where r is the number of segments extracted from each sliding window and $\phi(n)$ is the number of features extracted when the user is `STILL` or `WALKING`. This also explains the increased CPU load when sensors are on, as the computational engine has to extract features, say every 10 seconds in our implementation.

Battery Rate: The battery rate is defined as the additional amount of battery consumed while CASTRA is enabled. We used several design optimizations to reduce the cost of battery and data rate consumption which include leveraging Wi-Fi during training phase (our experiments observed increased battery drain when data is sent and received using 3G), enabling creation of trusted zones (in trusted zones, all the sensors are automatically switched off) and context-aware invocation of sensors to avoid continuous polling of sensors (sensors are off until user reaches the region of interest). These optimizations helped us to realize a 1%-2% battery drain per hour.

We observed minimal on device storage needs for CASTRA as most of the data and heavy computations resides in the Cloud. The data stored on the device mostly includes user configurations and event logs. The overhead of CASTRA mainly stems from data rate and by relying on Wi-Fi, one could significantly amortize the cost of bandwidth.

VIII. Conclusion

We present CASTRA a continuous, secure and user-friendly authentication and access control technology for mobile devices. Our results manifest that CASTRA can quickly identify stolen/lost/shared attacks (99% detection rate) and protect sensitive resources from unauthorized access. The minimal runtime overhead (e.g., data rate, battery consumption, memory, CPU load) in conjunction with its modular architecture enables one to integrate CASTRA with existing mobile based Internet of Things (IoT) to enable a seamless and secure access to the end services. In future, we will deploy the **context analyzer** on device and will investigate privacy aware schemes to protect leaking of sensitive information. In short, CASTRA is a technology that passively learns individual behavior and uses the learned behavioral traits to detect unauthorized access. We tested and evaluated CASTRA for a limited number of subjects and in our next stage, a larger scale (1000+) deployment is planned to evaluate its potential.

Acknowledgment

This project is the result of funding provided by the S&T Directorate of the United States Department of Homeland Security under contract number D15PC00155.

We thank Steve Uhl (CSC), Kermel Ruperto (CSC), and Paul Bryzek for supporting the development of CASTRA. We also thank Emeka Eyisi (UTRC), Teems Lovett (UTRC), Kishore Reddy (UTRC), Paul O'Neill (UTRC) for their useful discussion and ideas and all anonymous subjects who provided us data to perform the experiments and validate CASTRA. We also acknowledge Vincent Sritapan (DHS S&T) and Anil John (DHS S&T) for their useful discussions and thoughts. We also thank all anonymous reviewers for their insightful suggestions.

References

[1] O. Riva, C. Qin, K. Strauss, D. Lymberopoulos. *Progressive authentication: deciding when to authenticate on mobile phones*,In the Proceedings of USENIX Security Symposium, 2012.

[2] http://www.kaspersky.com/about/news/press/2013/one-in-every-six-userssuffer- loss-or-theft-of-mobile-devices, accessed 21 December 2016

[3] www.sophos.com/en-us/press-office/pressreleases/ 2011/08/67-percent-of-consumers-donot- have-password-protection-on-their-mobilephones.aspx

[4] C. Bhagavatula, K. Lacovino, S Kywe, L. Cranor, B. Ur. *Usability Analysis of Biometric Authentication Systems on Mobile Phones*, In the proceeding of SOUPS, 2014.

[5] C. Bo, L. Zhang, X. Li, Q. Huang, and Y.Wang. *Silentsense: silent user identification via touch and movement behavioral biometrics*, In the Proceedings of MobiCom, 2013.

[6] M. Nauman, S. Khan, X. Zhang, *Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints*, ASIACCS, China, April 13-16, 2010.

[7] M. Conti, V.T.N. Nguyen, B. Crispo, *Crepe: context-related policy enforcement for android*, in Proceedings of ISC 10, Springer-Verlag, Berlin, Heidelberg; pp. 331- 45

[8] W. Enck, M. Ongtang, P. McDaniel, *On lightweight mobile phone application certification*, in Proceedings of CCS 09, ACM, New York, NY, USA

[9] N. L. Clarke and S. M. Furnell. *Authenticating mobile phone users using keystroke analysis.* International Journal of Information Security, 6(1), 1-14, 2007.

[10] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D. Song. *Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication.* IEEE Trans. Inf. Forensics Security, 8(1):136 - 148, Jan. 2013.

[11] M. Jakobsson, E. Shi, P. Golle, R. CHow, *Implicit Authentication for Mobile Devices*, https://ssltest.cs.umd.edu/ elaine/docs/implauth.pdf

[12] L. Fridman, S. Weber, R. Greenstadt and M. Kam, *Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location*, arXiv preprint arXiv: 1503, 08479, 2015, pp. 1- 10

[13] T. Hastie, R. Tibshirani, J. Friedman. *The elements of statistical learning: data mining, inference and prediction.* second edition, Springer series in statistics, 2011.

[14] G.X. Ye, T.Z. Tang, D.Y. Fang, X.J. Chen, K.I. Kim, B. Taylor and Z. Wang, *Cracking Android pattern lock in five attempts*, In the Network and Distributed System Security Symposium 2017 (NDSS 17).

[15] M. Conti, I. Zlatea, B. Crispo, *mind how you answer me: transparently authenticating the user of a smartphone when answering or placing a cal*, ASIACCS 2011, China.

[16] D. M. Shila, K. Srivastava, P. O'Neill, K. Reddy, V. Sritapan, *A multi-faceted approach to user authentication for mobile devices âĂŤ Using human movement, usage, and location patterns*, IEEE HST, 10-11 May 2016, MA, USA.

[17] Smart lock, https://www.digitaltrends.com/mobile/how-to-use-android-smart-lock/.

[18] Y. Ren, Y. Chen, M. Chuah, J. Yang, *Smartphone Based User Verification Leveraging Gait Recognition for Mobile Healthcare Systems*, IEEE SECON 2013.

[19] D.M. Shila, K. Srivastava, P. O' Neill, *System and method of mobile based user authentication for an access controlled environment*, 62/492, 610, 2017.

[20] https://www.theverge.com/2016/5/2/11540962/iphone-samsung-fingerprint-duplicate-hack-security

[21] https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

[22] Sheldon Ross, *Introduction to Probability Models*, Eight edition, 2003

[23] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, *I Sensed It Was You: Authenticating Mobile Users with Sensor-enhanced Keystroke Dynamics*, 11th Conference on DIMVA, UK, July 10-11, 2014.

[24] V Stanciu, R. Spolaor, M. Conti, C. Giuffrida, *On the Effectiveness of Sensor-enhanced Keystroke Dynamics Against Statistical Attacks* , CODASPY, 2016.

[25] E. Hayashi, S. Das, S. Amini, J. Hong, I. Oakley, *CASA: context aware scalable authentication*, Symposium on Usable Privacy and Security (SOUPS) 2013, July 24 - 26, 2013, Newcastle, UK.

[26] U. Mahbub, R. Chellapa, *PATH: Person Authentication using Trace Histories*, arXiv:1610.07935v1 [cs.CV] 25 Oct 2016

[27] Y. Yang, J. Sun, L. Guo, *PersonaIA: A Lightweight Implicit Authentication System based on Customized User Behavior Selection*, IEEE TPDS, 2016.

[28] M. Miettinen, S. Heuser, W. Kronz, A. Sadhegi, N. Asokan, *ConXsense: Automated Context Classification for Context-Aware Access Control* , ASIA CCS14, June 2014, Kyoto, Japan.

[29] S. Eberz, K. Rasmussen, V. Lenders, Ivan Martinovic, *Evaluating Behavioral Biometrics for Continuous Authentication: Challenges and Metrics*, ASIA CCS 17, April 2017, Abu Dhabi, United Arab Emirates.

[30] B. Attaullah, B. Crispo, F. Del Frari, K. Wrona, *ITSME: Multi-modal and Unobtrusive Behavioural User Authentication for Smartphones*, In Proc of Technology and Practice of Passwords: 9th International Conference, PASSWORDS 2015, Cambridge, UK, December 2015.

[31] Weidong Shi, Jun Yang, Yifei Jiang, Feng Yang, Yingen Xiong, *SenGuard: Passive user identification on smartphones using multiple sensors*, IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2011.

[32] Google Activity Recognition API. https://developers/google/android/gms/location/ActivityRecognitionApi