Progressive authentication: deciding when to authenticate on mobile phones

Oriana Riva[†] *Chuan Qin*[§*] *Karin Strauss*[†] *Dimitrios Lymberopoulos*[†]
[†]*Microsoft Research, Redmond* [§]*University of South Carolina*

Abstract

Mobile users are often faced with a trade-off between security and convenience. Either users do not use any security lock and risk compromising their data, or they use security locks but then have to inconveniently authenticate every time they use the device. Rather than exploring a new authentication scheme, we address the problem of deciding *when* to surface authentication and for *which applications*. We believe reducing the number of times a user is requested to authenticate lowers the barrier of entry for users who currently do not use any security. Progressive authentication, the approach we propose, combines multiple signals (biometric, continuity, possession) to determine a level of confidence in a user's authenticity. Based on this confidence level and the degree of protection the user has configured for his applications, the system determines whether access to them requires authentication. We built a prototype running on modern phones to demonstrate progressive authentication and used it in a lab study with nine users. Compared to the state-of-the-art, the system is able to reduce the number of required authentications by 42% and still provide acceptable security guarantees, thus representing an attractive solution for users who do not use any security mechanism on their devices.

1 Introduction

Security on mobile phones is often perceived as a barrier to usability. Users weaken the security of their phones for the convenience of interacting with their applications without having to type a password every time. As a result, according to a recent study [25], more than 30% of mobile phone users do not use a PIN on their phones. On the other hand, the amount of high-value content stored on phones is rapidily increasing, with mobile payment and money transfer applications as well as enterprise data becoming available on mobile devices [22].

One approach for increasing security of mobile phones is to replace PINs and passwords with a more suitable authentication scheme. Yet, alternative schemes have their own flaws. Token-based approaches [6, 31, 35], for instance, are in general harder to attack than passwords, but they go against the desire of users to carry fewer devices. Biometric identification has gained interest in the mobile community [11], but not without performance, acceptability, and cost issues [27].

In this work, we look at the problem of mobile authentication from a different angle. Rather than exploring a new authentication scheme, we study the problem of *when* to surface authentication and for *which applications*. Unlike desktops and laptops, which users tend to use for a long, continuous period of time, users access mobile phones periodically or in response to a particular event (*e.g.*, incoming email notification). This lack of continuous interaction creates the need to authenticate with the device almost every time users wish to use it. Even though the interaction between users and mobile devices might not be continuous, the physical contact between the user and the mobile device can be. For instance, if a user places his phone in his pocket after a phone call, even though the user stops interacting with it, the authenticated owner is still "in contact" with the device. When the user pulls the phone out of his pocket, authentication should not be necessary. On the other hand, if the phone lost contact with the authenticated user (*e.g.*, left on a table), then authentication should be required. As a result, if the phone is able to accurately infer the physical interaction between the authenticated user and the device (*e.g.*, through its embedded and surrounding sensors), it can extend the validity of a user authentication event, reducing the frequency of such events.

This approach not only significantly lowers the authentication overhead on the user, but also makes the authentication effort proportional to the value of the content

[*]Work done while interning at Microsoft Research.

being accessed. If the system has strong confidence in the user's authenticity, the user will be able to access any content without explicitly authenticating. If the system has low confidence in his authenticity, he will only be able to access low-value content (*e.g.*, a weather app) and will be required to explicitly authenticate to view high-value content (*e.g.*, email, banking). By doing so, the system provides low overhead with security guarantees that can be acceptable for a large user population, particularly users who do not use any security lock on their phones – our primary target.

We propose mobile systems that *progressively authenticate* (and de-authenticate) users by constantly collecting cues about the user, such that at any point in time the user is authenticated with a certain level of confidence. Several authentication signals are used. The system can exploit the increasing availability of sensors on mobile devices to establish users' identity (*e.g.*, through voice recognition). Once the user is authenticated, the system can exploit continuity to extend the validity of a successful authentication by leveraging accelerometers and touch sensors to determine when the phone "has left" the user after a successful login. Finally, as users' personal devices are increasingly networked, it can use proximity and motion sensors to detect whether the phone is next to another device of the same user where he is currently authenticated and active.

Automatically inferring a user's authenticity from continuous sensor streams is challenging because of inherent noise in sensing data as well as energy and performance constraints. If sensor streams are naively used, the system could suffer a high number of false rejections (not recognizing the owner) and/or unauthorized accesses (recognizing others as the owner). Our solution consists of using machine learning techniques to robustly combine and cross-check weak sensor signals. Sensors and related processing create energy and performance issues. First, processing the sensor data and running the inference stack on a mobile device can be challenging. We show how to architect the system to mitigate these problems by offloading processing to the cloud or disabling computation-expensive signals. Second, continuously recording data from the sensors can be a major power hog in a mobile device. We rely on existing architectures for low power continuous sensing such as LittleRock [28, 29]. We expect in the near future all mobile devices to be equipped with such low power sensing subsystems, as it is already made possible by the latest generation of mobile processors with embedded ARM Cortex M4 based sensing subsystems [37].

In summary, the contributions of this work are threefold. First, we introduce the progressive authentication model, which relies on the continuous combination of multiple authentication signals collected through widely available sensor information. This approach makes the authentication overhead lower and proportional to the value of the content being accessed. Second, we present a Windows Phone implementation of progressive authentication, which explores the implications of applying the model to a concrete phone platform. Third, we show through experiments and a deployment of the system with 9 users how progressive authentication (within our study setup) achieves the goal of reducing the number of explicit authentications (by 42%) with acceptable security guarantees (no unauthorized accesses and only 8% of cases in which the user's authenticity is estimated higher than it should be), power consumption (325 mW) and delay (987 msec). We believe this represents an attractive solution for users who currently do not use any security lock on their phones.

The rest of the paper is organized as follows. The next section motivates our work through a user study on access control on mobile phones and sets the goals of our work. Section 3 presents the key principles of progressive authentication, and Section 4 and Section 5 demonstrate these principles through the prototype system we designed and implemented. We then evaluate our system: in Section 6 we describe how we collected sensor traces and trained the inference model, and in Section 7 we present the experimental results. Finally, we discuss related work and conclude.

2 Motivations and assumptions

Most smart phones and tablets support some locking mechanism (*e.g.*, PIN) that, if used, prevents access to all the devices applications, with the exception of a few pre-defined functions such as answering incoming calls, making emergency calls, and taking photos. Unlike desktops and laptops, users interact with their phones for short-term tasks or in response to specific events (*e.g.*, incoming SMS notification). This results in users having to authenticate almost every time they wish to use their phone.

2.1 User study

We investigated how well this model meets the access control needs of mobile users through a user study [13]. We recruited 20 participants (9M/11F, age range: 23-54) who owned both smart phones and tablets, using Microsoft's recruiting service to reach a diverse population in our region. They were primarily iPhone (9) or Android (8) phone users. 11 used a PIN on their phone and 9 did not use any. Although this study is much broader, some of its findings motivate progressive authentication.

Beyond all-or-nothing: Today's mobile devices force users to either authenticate before accessing any applica-

tion or to entirely disable device locking. According to our study, this all-or-nothing approach poorly fits users' needs. Across our participants, those who used security locks on their devices wanted about half of their applications to be unlocked and always accessible. Not coincidentally, participants who did *not* use security locks wanted to have about half of their applications locked.

Multi-level security: Even when offered the option to categorize their applications into two security levels (*i.e.*, one unprotected level and one protected by their preferred security scheme), roughly half of our participants expressed the need for at least a third category, mainly motivated by security and convenience trade-off reasons. Most participants chose to have a final configuration with one unlocked level, one weakly-protected level for private, but easily accessible applications, and one strongly-protected level for confidential content such as banking.

Convenience: Participants were asked to rank the security of different authentication approaches (PINs, passwords, security questions, and biometric authentication). More than half of the participants ranked passwords and PINs as the most secure authentication schemes, but when asked to indicate their favorite authentication mechanism, 85% chose biometric systems. Convenience was clearly the dominant factor in their decision.

These lessons highlight the value of correctly trading off security and convenience on a per-application basis. This is the cornerstone of progressive authentication.

2.2 Assumptions and threat model

Progressive authentication builds on two main assumptions. First, we assume a *single-user model*, which is in line with current smart phones. However, such as in related systems [20, 23], adding multiple profiles (*e.g.*, family members, guests) to progressive authentication would be straightforward. Second, we assume the availability of *low-cost sensors* in the device and the environment for detecting a user's presence and identity. Indeed, the sensors used in our implementation are widely available in modern devices and office environments. As more sensors become pervasive (*e.g.*, 3D depth cameras), they can be easily folded into the system.

The main security goal of progressive authentication is to protect important applications from unauthorized use, while providing a probabilistic deterrent to use of less sensitive applications by others. In a noise-free environment where sensor signals are reliable and error-free, the same security guarantees can be provided regardless of the physical environment and the people present. In reality, these guarantees depend on several external conditions (presence of people with similar aspect, background noise, light conditions, etc.), such that the system relies on the probabilistic guarantees of a machine learning model trained to account for these variables.

Progressive authentication seeks to preserve its goal in the presence of adversaries under the following conditions. Our model inserts barriers against unauthorized use when the phone is left unattended. Attacks can be carried out by both strangers and "known non-owner" attackers such as children and relatives. Attackers can operate both in public (*e.g.*, a conference room, a restaurant) and private spaces (*e.g.*, an office, at home). Attackers may *not know* which signals progressive authentication utilizes (*e.g.*, not knowing that voice is a factor and speaking into the system) or they may *know* which signals are in use and therefore try to avoid them (*e.g.*, remaining silent). Attacks in which the adversary obtains privileged information (*e.g.*, passwords or biometrics) are orthogonal to our guarantees, *i.e.*, progressive authentication does not make these authentication mechanisms more secure, so they are not considered in our security analysis. Finally, we assume the phone operating system and the devices the user owns are trusted, and that attacks cannot be launched on wireless links used by the system.

In the presence of the legitimate user, progressive authentication allows him to implicitly authorize other users (by simply handing them the phone), as we assume the above attacks can be prevented by the owner himself. For instance, if the user authenticates with the phone by entering a PIN and, soon after that, hands it to another person, the phone will remain authenticated.

Stronger attacker models are possible, but addressing them would require more sophisticated sensors and inference algorithms, thus increasing the burden on the user (cost of sensors, phone form factor, training effort) or limiting the applicability of our approach in environments not equipped with these sensors.

2.3 Goals

Based on the assumptions and threat model described above, our system has the following goals:

Finer control over convenience versus security trade-offs: Today's models are all-or-nothing: they require users to make a single security choice for all applications. Users who do not use authentication typically make this choice for convenience. Our goal is to give these users more flexibility such that they can keep the convenience of not having to authenticate for low-value content, but to improve security for high-value content. Progressive authentication enables per-application choices.

Moderating the surfacing of authentication: Too many authentication requests can be annoying to users. In addition to per-application control, our goal is to reduce the number of times users are inconvenienced by having

to authenticate. We do this by augmenting devices with inference models that use sensor streams to probabilistically determine if the user should continue to be authenticated, avoiding the surfacing of authentication requests.

Low training overhead for users: Our focus is on convenience, so the above inference models should be easy to train and deploy. Some scenarios, such as potential attacks, are difficult to train after the phone ships, so this type of training needs to be done in advance, in a user-agnostic manner. Personalized models (*e.g.*, voice models), however, cannot be trained in advance because they depend on input from the user. In such cases, users should not be burdened with complex training routines. Our solution partitions the model into a high-level, user-agnostic model that can be trained before shipping, and low-level, personalized models that can be trained via normal user interaction with the device (*e.g.*, by recording the user's voice during phone calls).

Overhead management: Mobile phones are afflicted with battery limitations, yet their computing capabilities are limited. It should be possible to let the system transition between high-performance and energy-saving modes. We achieve this goal by offloading computation to the cloud, or moving all the computation locally and disabling certain sensor streams, at the cost of accuracy.

3 Progressive authentication model

Progressive authentication establishes the authenticity of the user by combining multiple authentication signals (*multi-modal*) and leveraging *multi-device* authentication. The goal is to keep the user authenticated while in possession of the device (*i.e.*,*continuity* since a last successful authentication is detected) or de-authenticate the user once the user lets go of it (*i.e.*, a discontinuity is detected). The confidence level in the user's authenticity is then compared to one authentication threshold for a single-level approach, or to multiple authentication thresholds for *multi-level* authentication.

Multi-modal. Possible signal types used for multi-modal authentication are the following:

Biometric signals: user appearance and sound (face and voice recognition). Fingerprinting and other hard biometric identification can also be used, but we focus on low-sensitivity signals. High-sensitivity signals may result in privacy concerns (*e.g.*, if stored on untrusted servers).

Behavioral signals: deviation of a user's current behavior from his past recurrent behavior may result in lower confidence. For example, if the phone is used at an unusual time and in a location that the user has never visited, then authentication may be surfaced. Location signals are a subset of behavioral signals.

Possession signals: nearby objects that belong to the user, such as a laptop or a PC, may increase the confidence level of the phone. This may be detected using Bluetooth signal strength or RFIDs, for example.

Secrets: PINs and passwords are still compatible with progressive authentication and actually represent some of the strongest signals it uses. They are requested from the user when the system is unable to determine the user's authenticity with high confidence.

In combining these signals several challenges must be considered. First, most signals are produced using unreliable and discrete sensor measurements. Second, certain signals may require combining readings from sources with different sampling frequencies and communication delays. As a result, most signals are not individually sufficient to determine user authenticity and when combined they may be inconsistent as well. Finally, signals vary in strength. Some signals can provide a stronger indication of authenticity than others because, for example, some may be easier to fake and some are more discriminating than others (*e.g.*, a PIN vs. the user's voice which may be recorded in a phone call). For all these reasons, signals need to be combined and cross-checked. However, drawing the correlations across these signals manually is a cumbersome job, prone to errors and inconsistencies. As we discuss in the next section, we use machine learning tools to derive a robust inference model from the signals we collect.

Continuous. Continuity is one of the cornerstones of progressive authentication. It comes from the observation that users are likely to use their phones shortly after a previous use. For example, after the user reads emails, he locks the phone to save energy. He keeps holding the phone and talking to someone else. When he tries to use the phone five minutes later, the phone is locked even if he did not let go of it. If the user has been touching the phone since the last successful authentication, the authentication level should be maintained unless "negative" signals are being received (*e.g.*, mismatching biometric signals). By "touching", we currently mean actively holding or storing the phone in a pocket. A phone's placement with respect to the user can be determined by accelerometers, touch screens, light, temperature and humidity sensors, most of which are embedded in modern phones.

Multi-device. Progressive authentication takes advantage of widespread device connectivity to gather information about the user from other devices he owns. If a user is logged in and active in another nearby device, this information represents a strong signal of the user's presence. For example, if a user enters his office, places the phone on his desk, logs into his PC and wants to use the phone, the phone already has some cues about his

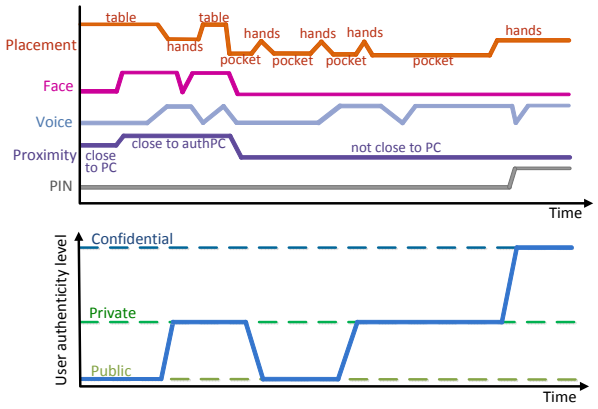


Figure 1: Progressive authentication in action. Signals are combined to infer the level of confidence in user authenticity. Based on this level, access (without PIN) to public, private or confidential content is granted.

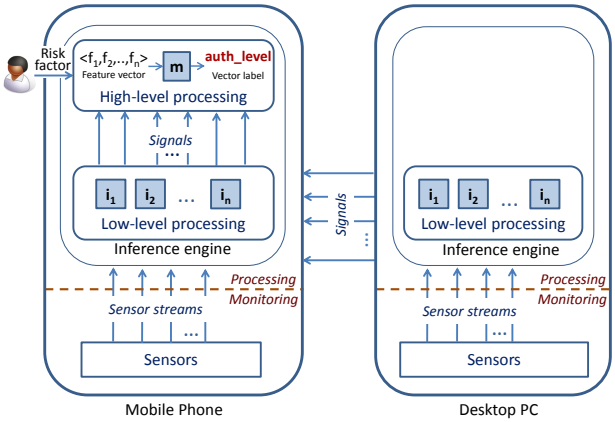


Figure 2: Progressive authentication architecture. The two-level inference engine processes incoming sensor data to estimate the level of a user's authenticity.

authenticity. Devices need to be registered once (*e.g.*, at purchase time) so the overhead for the user is limited.

Multi-level. Users storing high-value content on their devices want to protect this harder than less valuable information. Progressive authentication associates user authenticity with a confidence level. This enables the system to depart from the all-or-nothing paradigm and allows the user to associate different protection levels to different data and applications. For example, a weather or a game application does not reveal any sensitive data, so it can be made accessible to anybody. An email or a social networking application contains sensitive information, but perhaps not as sensitive as a financial application. Thus, the confidence level required for accessing email is lower than that for accessing the banking application. In a complete system, users may configure their applications into an appropriate security level, perhaps at installation time, may specify levels by application categories (*e.g.*, games, email, banking, etc.), or may even accept a default configuration set by their corporate's IT department. We expect users to deal with no more than three or four levels. The preferred configuration that emerged from our user study was three levels. Our system prototype adopts the same.

Putting it all together. Figure 1 illustrates the progressive authentication process. Multiple signals are aggregated into a single framework, which determines the level of confidence in user authenticity. The example considers continuity (phone placement), biometric (face and voice) and multi-device (proximity to a PC where the user is logged on/off) signals as well PIN events. Based on the confidence level, the user is allowed to access predefined data and applications in three sensitivity levels: public, which requires a very low confidence; private, which requires medium confidence; and confidential, which requires very high confidence. Note that when the user's authenticity level is too low for accessing high security applications, the system requires a PIN, which raises the score to the confidential level. Previously achieved levels are maintained by continuity as long as signals are sufficient.

4 Architecture

This section describes the architecture of progressive authentication. Note that the system is designed for phones and mobile devices with rich sensing capabilities, but could also be applied to other types of computers. The specific system we consider is a 2-device configuration including a mobile phone and another user-owned device such as a desktop PC. Progressive authentication is used to control authentication on the phone and the PC simply acts as another sensor.

As Figure 2 shows, the software running on each device consists of two levels. At the monitoring level, sensor streams are continuously collected and fed as inputs to the inference engine, the core of our system. The inference engine processes sensor information in two stages. A collection of inference modules (i_i boxes inside low-level processing) processes raw sensor data to extract the set of primitive signals described in the previous section, such as placement of the mobile device as an indication of continuity, presence of the user through biometric inference, and proximity to other known devices as an indication of device possession and safe location. A confidence level is associated to each signal. These inferred signals are not sufficiently robust in isolation to authenticate or de-authenticate the user, hence the next step (high-level processing) combines them to compute the overall confidence in the user's authenticity.

We rely on well-established machine learning techniques, particularly support vector machine (SVM) models, to do this properly. In using such models, *feature extraction* is the most critical step. Features should provide as accurate and discriminative information as possible about the user's authenticity and whether the phone has left or not the user since the last successful authentication (Table 1 in the next section lists all features we have considered in our prototype). Once the high-level processing extracts these features from the most recent signals produced by low-level processing, they are combined in a *feature vector*. This vector is passed as input to the machine learning model (*m* box inside high-level processing), which in turn associates a *label* to it, indicating the estimated confidence in the user's authenticity at that particular time. This label is then mapped to one of the system protection levels, which we currently limit to three. Named after the most restrictive type of application to which they allow access, they are: *public level*, in which only applications classified as public are accessible; *private level*, in which public and private applications are accessible; and *confidential level*, in which all applications are available.

The machine learning model is trained offline using a set of *labeled feature vectors* (Section 6 describes how they are obtained). The model is trained to minimize a *loss function*, which represents the relative seriousness of the kinds of mistakes the system can make. More specifically, it represents the loss incurred when the system decides that a user has authenticity level higher than it should be and possibly automatically authenticates him (*false authentication*), or when the system decides that he has authenticity level lower than it should be and requires explicit authentication (*false rejection*). The system exposes a so-called *risk factor* to users such that they can themselves adjust how aggressively they want to trade security for convenience. The higher the risk factor, the higher the convenience, but also the higher the security risk.

Unlike low-level processing, which is continually active as sensor data are fed into the system, high-level processing is activated only when the phone detects touchscreen activity (*i.e.*, the user is about to use an application). This model is sufficient to provide the illusion of continuous authentication without unnecessarily draining the battery of the phone and wasting communication resources. The high level keeps being invoked periodically, as long as the phone's touch screen detects activity, allowing the system to promptly adjust its authentication decision to sudden changes in ambient conditions, which may indicate threat situations.

A main challenge for this system is resource management. Sensor monitoring and processing need to minimally impact the phone's operation and battery lifetime. For this reason, we designed the architecture in a modular manner. Depending on its resource constraints, the phone can decide to offload some processing to the cloud or another device so as to reduce the computing overhead, or it can instead decide to do more processing locally in order to reduce communication. In addition, all low-level processing modules are independent and the high-level processing can work with all or some of them, *i.e.*, some modules can be disabled to save resources, in which case the corresponding features are computed based on the latest available signals. In the evaluation section, we show the impact of these resource-saving decisions on the overall system's accuracy. The system currently supports a few static configurations. Systems such as MAUI [8] and AlfredO [10] may be used to support dynamic module reconfiguration.

For this kind of authentication to be viable, it is important to keep the machine learning models as user-agnostic as possible. The main inference model can be trained independently using a broad population (before the phone ships). Some of the low-level inference algorithms, specifically face and voice recognition, are user specific. However, their training does not incur a high overhead on the user because it can be done in the background, during user interaction with the device. A user's appearance can be recorded while the user types or reads from the PC. 1 minute of image recording is sufficient for the face recognition model to work accurately. Voice can be recorded during phone calls. In particular, Speaker Sense [21], the algorithm we used in our implementation, was designed to specifically support such type of training. The voice model was trained using 2 minutes of audio recording.

5 Implementation

We built a progressive authentication prototype on Windows Phone 7.1 OS and used it in a 2-device configuration with a desktop PC running Windows. The bottom part of Figure 3 shows the detailed architecture of the system running on the phone and on the desktop PC.

Sensors. On the phone, we use accelerometers, light, temperature/humidity sensor, touch screen, login events, microphone, and Bluetooth receiver. All these sensors are widely available on commercial phones (*e.g.*, Android) today. As the platform we chose does not yet support light and temperature/humidity sensors, we augmented it by using the .NET Gadgeteer [9] sensor kit, a platform for quick prototyping of small electronic gadgets and embedded hardware devices. Figure 5 shows the final prototype. On the PC, we use external webcams, sensors for activity detection (mouse motion, key presses, login/logout events), and Bluetooth adaptors. Bluetooth is used for proximity detection between phone

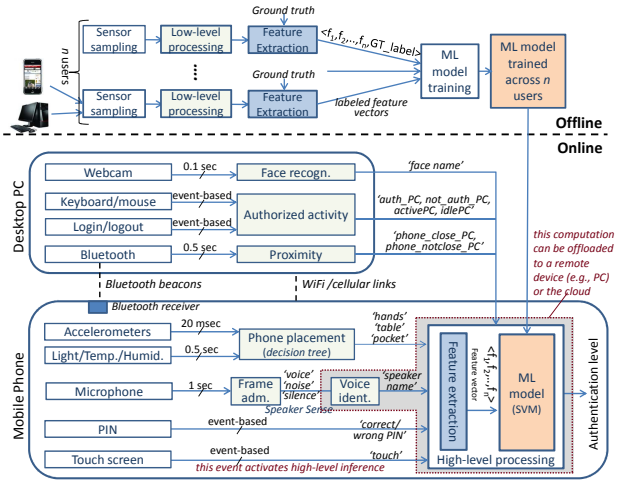


Figure 3: Detailed architecture of a 2-device progressive authentication implementation. Offline, a machine learning model is trained using traces from several users. Online, the model is loaded into the phone and invoked using the features extracted from the most-recent sensor signals. Signals' confidence values are not shown.

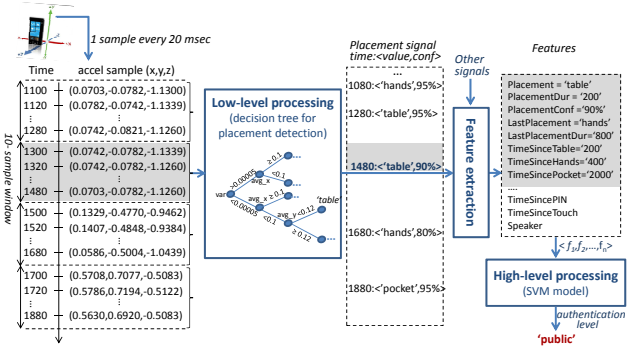


Figure 4: Processing chain from accelerometer data (sampled every 20 msec) to placement signals and to continuity features which are fed as input to the machine learning model to estimate the user's authentication level.

and PC. The system employs Bluetooth, WiFi or cellular links for communication. The channel can be encrypted.

Low-level processing. It currently produces these signals: proximity to known devices (detected using Bluetooth beacons), presence of activity on phone or PC (detected using touch screen and activity detection sensors, respectively), speaker identification, face recognition, and phone placement.

Voice recognition relies on Speaker Sense [21], which is based on a Gaussian Mixture Model (GMM) classifier [30]. To make the recognition process more efficient, in a first stage, called Frame Admission, sound recorded by the phone's microphone is processed to identify voiced speech frames and discard silence frames or unvoiced speech frames. Voiced speech frames are admitted to the second stage, called Voice Identification, where speaker recognition occurs and produces an identifier with associated confidence.

Face recognition is based on a proprietary algorithm. Pictures are collected using a PC's webcam and fed into the face recognition model. This returns an identifier of the recognized person along with a confidence.

High-level processing. The primitive authentication signals and associated confidence levels generated by the low-level processing are combined at this stage to extract the features used by the machine learning model. Figure 4 gives a detailed example of the entire process, where accelerometer data are aggregated every 200 msec and processed by the placement decision tree to extract placement signals. Continuity features are extracted and combined in a feature vector with all other features (biometric, possession, continuity and secret-derived features). Feature vectors are fed as input to the machine learning (ML) model and a label indicating the authenticity of the user is produced.

Table 1 lists all features currently used. These features cover all categories of signals described in Section 3 with the exception of behavioral signals. For these, we plan to first collect longer-term traces of users interacting with our prototype, and then extract behavioral patterns for each user. The current implementation relies on SVM models, but we experimented also with Decision Tree and Linear Regression models (see Section 7).

Finally, the high-level processing and some of the low-level processing can be too computationally intensive for mobile phones, so we support a *minimum_power configuration* where Voice Identification and high-level processing (gray box in Figure 3) are offloaded to the cloud (Windows Azure) or a remote device.

User interface. Figure 5 shows our current user interface. We do not claim this to be the most appropriate interface. For instance, users may want to configure the level of details returned in the feedback. However, we used it during our user study [13] and it was well received by our participants. When the phone is picked up, the user can see which applications are accessible and which are locked (not shown in the figure). When trying to access a locked application, the user is prompted to enter a PIN and receives a visual explanation of why access is denied. Icons representing the authentication signals are lit up (the corresponding signal is present) or grayed out (the corresponding signal is absent). This allows users to understand why a PIN is required.

Table 1: Machine learning features used in the high-level processing.

Category	Features	Description
Cont.	Placement, PlacementDuration, PlacementConf	Current placement of the phone, how long it has lasted, and associated confidence
Cont.	LastPlacement, LastPlacementDuration	Last placement of the phone, and how long it lasted
Cont.	TimeSinceTable, TimeSinceHands, TimeSincePocket	Time elapsed since the last time the phone was on the table, in the user's hands, or pocket
Cont./Secrets	TimeSincePIN, TimeSinceTouch	Time since last login event and time since the phone's screen was last touched
Biom.	Speaker, SpeakerConf	Whether a human voice was identified (owner, other, no-voice) and associated confidence
Biom.	TimeSinceOwnerVoice, TimeSinceNonOwnerVoice	Time since (any) voice was identified
Biom.	TimeSinceSound	Time since any sound (either voice or noise) was detected
Poss./Biom.	ProxAuthDev, ProxAuthDevConf	Proximity of phone to a device where the user is logged in and active, and confidence
Poss./Biom.	TimeSinceProx	Time elapsed since the proximity status last changed

Figure 5: System prototype running on a WP Samsung Focus augmented with Gadgeteer sensors (white box).

6 Data collection and offline training

We collected traces from 9 users (4F, 5M) to evaluate our system prototype. The participants were researchers (4), engineers (3) and admins (2) working in our group. During the lab study, they used the system for a little longer than an hour. The study consisted of two parts. The first part focused on collecting data for evaluating the accuracy of the low-level processing models. It lasted for roughly 30 minutes and it was guided by a researcher. The data collected was used to train voice identification and face recognition models, which were plugged into the system for use in the second part.

The second part of the study was conducted in two user study rooms that had a participant and an observer side, which allowed us to collect information about the activities in which the participant was engaged without disturbing. This information was used to generate the *ground truth* necessary for training the high-level processing model (more details below). Each study session was also video-recorded such that it could be used later for the same purpose. Participants were asked to perform a series of tasks by following a script. The script consisted of ordinary tasks in an office setting, involving a phone and a desktop PC. The user was asked to check his phone from time to time, work on the PC, answer incoming phone calls, chat with a colleague, and walk to a "meeting room" with his phone to meet a colleague. The script also included security attacks. Overall, the second part of the study lasted 40 minutes. The data collection ran continuously and collected sensor measurements from phone and PC, as well as types and time of applications invoked on the phone.

Each participant interacted with a total of five phone applications categorized as public (1 application), private (3) or confidential (1). This distribution reflects the results of our preliminary user study where across their most frequently used applications (*i.e.*, more than 10 times a day) participants wanted 22% of them to be always available (public) and the rest protected by some security mechanism (private or confidential). We then assumed a similar frequency of invocation across these 5 applications: across all participants, the average ratio at which public, private, and confidential applications were invoked was 19%:57%:24%, respectively.

6.1 Attack scenarios

In the script, we inserted 3 attack sessions for a total of 26 attack attempts (12 to private applications and 14 to confidential applications) covering the threat scenarios described in Section 2.2. In two sessions the user leaves the office and the phone on the desk, and soon after that an attacker (*i.e.*, a user unknown to the system) enters the room and tries to use the phone. These attacks take place in a private place (*i.e.*, office). We simulate both an attacker that knows which signals the system is using and one that does not know. In the first case, the attacker's strategies are staying out of the camera's range, being silent and entering the office soon after the user leaves and locks his PC. In the second case, the attacker is in the camera's range and speaks three times for about 10 sec-

onds. In each of the two sessions we have both types of attacks. The third type of attack occurs in a public place. The participant forgets his phone in a meeting room. The participant is asked to interact with the phone just until leaving the meeting room. The attacker enters the room soon after the user leaves, picks the phone up and tries to access the phone applications. These scenarios cover a small range of situations, but in the office setup we selected for the study they simulate all types of attacker strategies we considered in our threat model.

6.2 Training and testing the models

Low-level processing models were trained and tested using the data collected in the first part of the study. The inferred model for placement detection is the same across all users, while the face and voice recognition models are user specific (more details in Section 7.5). To train the high-level inference model, which is user-agnostic, we extracted feature vectors from each user's trace, and used WEKA [14], a popular suite of machine learning software, to train three models: decision tree, support vector machine with a non-linear model, and linear regression. As the top part of Figure 3 shows, the model is trained offline, across users. 8 users' traces are used for training and the remaining user's trace for testing. A feature vector is generated for each line in the user trace and then labeled with a ground truth label – the state the model is trained to recognize under different conditions. The ground truth labels are extracted by the study's observer based on a set of predetermined definitions. These definitions are used to provide an objective, quantifiable, and implementation-independent way of defining the ground truth labels:

Public Label: The legitimate owner is not present OR Other people are in contact with the phone OR The legitimate owner is present, but not in contact with the phone and other people are present.

Private Label: The legitimate owner has been in contact with the phone since the last private-level authentication OR The legitimate owner is present and is not in contact with the phone and no one else is present.

Confidential Label: The legitimate owner has been in contact with the phone since the last confidential-level authentication.

Across all participants, the distribution of ground truth labels was such that 55.0% of the labels were of type public (2379 labels), 42.6% were of type private (1843 labels), and the rest were of type confidential (2.4% or 105 labels). The distribution of the ground truth labels should not be confused with the distribution of application invocations and the type of application invoked. For instance, despite the participants invoked the confidential application as frequently as the public application or one of the three private applications, only 2.4% of the labels were confidential. This means that only in few cases the signals the system could collect were sufficient to automatically authenticate the user at the confidential level. Hence, confidential applications required a PIN most of the time.

In the testing phase, the model was invoked using a user's trace. For each touch event which had been generated less than 0.5 seconds earlier, a feature vector was generated and fed as input to the model. To assess the model's accuracy, the output of the model was then compared against the ground truth label. We also tried training the high-level model on a per-user basis, but the accuracy of the resulting model was lower than that of the generalized model, so we did not pursue this strategy (more details in Section 7).

7 Experimental evaluation

We verify that our prototype meets the following goals: *(1)* Lowers authentication overhead on mobile phones; *(2)* Allows users to trade off stronger protection and more convenience; *(3)* Achieves reasonable accuracy in estimating the level of user authenticity; and *(4)* Provides acceptable execution time and power consumption.

7.1 Authentication overhead

The main goal of progressive authentication is to reduce the authentication overhead on mobile phones and become a viable (more secure) solution for users who currently do not use security locks on their devices. We measure how many times the participants executing the tasks of our script had to type a PIN and how many times they would have had to type a PIN without progressive authentication. We also count the number of unauthorized authentications (UAs) that occurred during the attack scenarios – cases of false authentication in which a non-legitimate user tried to unlock the phone and succeeded. For progressive authentication, we use an SVM model. As baseline schemes, we assume a system that locks the user's phone after 1 minute of inactivity (PhoneWithPIN) and one that never locks the phone (PhoneWithoutPIN). We choose a 1-minute timeout for the baseline case because the script was designed to allow for frequent application invocations in a limited time window. In real-life such invocations would be spread over longer time. Table 2 shows that, on average, compared to a state-of-the-art phone with PIN, progressive authentication reduces the number of times a user is requested to enter a PIN by 42% and provides the same security guarantees (*i.e.*, 0 UAs). Our claim is that such a system would be acceptable by users who currently do not use PINs on their phones. If for some of these users

Table 2: Reduction in the authentication overhead with progressive authentication (using an SVM model without loss function) compared to the two baseline cases available on today's mobile phones. The table reports the number of times the user was required to enter a PIN (Num of PINs) and how many unauthorized authentications (Perc of UAs) occurred. Average and standard deviation are reported.

Avg [Stdev]	PhoneWithoutPIN	PhoneWithPIN	ProgAuth
Num of PINs	0.0 [0.0]	19.2 [0.6]	11.2 [0.4]
Perc of UAs	100% [0.0]	0.0% [0.0]	0.0% [0.0]

this reduction in overhead is not sufficient, they can decrease it even more by tuning the system's risk factor, which we evaluate next.

7.2 Convenience and security trade-offs

As with password-based systems, for which users can set easy or hard passwords, with progressive authentication users can set a high or low risk factor (R). If R is high, the inference model is optimized for convenience – it reduces the number of false rejections (FRs), but it can possibly increase the number of false authentications (FAs). If R is low, the inference model is optimized for security. Recall that FAs are cases in which the system overestimates the level of the user authenticity and grants the user automatic access instead of requesting a PIN. Unauthorized accesses (UAs), reported in the previous test, represent a subset of the total number of FAs, as UAs only refer to cases in which non-legitimate users got access to the system. FAs refer to any user, legitimate or non-legitimate ones. FRs are cases in which the system underestimates the level of user authenticity and unnecessarily requires a password.

We configure 4 additional SVM models with increasing risk factors and compare their rates of FAs and FRs for private and confidential applications against a baseline without loss function (this is the same model used in Section 7.1, which uses $R = 1$). Table 3 shows that as R increases the percentage of FAs for private applications (FA Priv) increases from 4.9% to 16.1% and there are no FAs for confidential applications (FA Conf). Conversely, FRs decrease. With $R = 20$, the system reduces FR Priv to only 34.4% (*i.e.*, the user is required to enter a PIN for private applications 1 out of 3 times), but it still requires PINs for confidential applications most of the time (96.8%). This happens because the loss function used for optimizing the model always penalizes more strongly FAs for confidential applications (*i.e.*, if the penalty for accessing private applications is P, that for confidential applications is P^2). This makes it very hard for the system to allow access to confidential applications without a PIN.

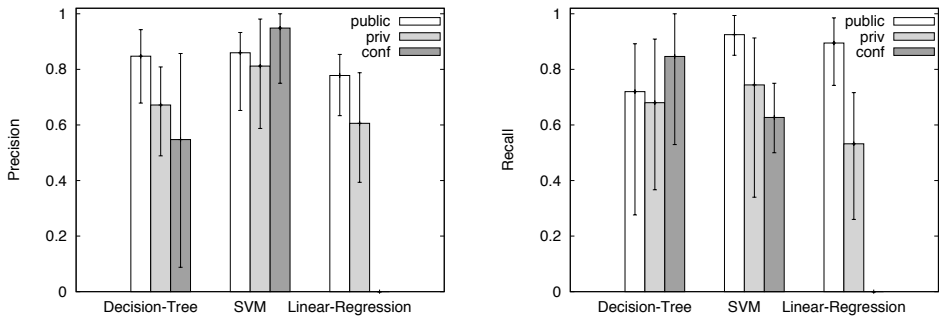
Table 3: Comparison of 5 SVM models, each with a different risk factor (R). One default model does not use any loss function ($R = 1$), while the other 4 are optimized either for convenience ($R = 5$ and $R = 20$) or for security ($R = 0.05$ and $R = 0.2$). The table reports percentage of false authentications and false rejections for private (FA Priv and FR Priv) and confidential (FA Conf and FR Conf) applications.

Risk factor	%FA Priv	%FA Conf	%FR Priv	%FR Conf
0.05	3.3	0.0	57.7	100.0
0.2	3.6	0.0	55.8	100.0
1	4.9	0.0	53.5	98.4
5	5.8	0.0	39.9	96.8
20	16.1	0.0	34.4	96.8

7.3 High-level processing accuracy

We have so far used an SVM model. We now evaluate the accuracy of high-level processing in more detail and compare SVM against two other popular modeling techniques: a decision tree with maximum depth of five and linear regression. All models are trained and cross-validated across users (*i.e.*, successively trained on every eight users and tested on the ninth user). None of the models use a loss function. Unlike the previous tests where we measured the accuracy of the model in estimating the level of user authenticity only for the first touch event leading to an application invocation, here we evaluate its accuracy in estimating the ground truth labels for all touch events extracted from the trace. This means that this analysis not only reports whether access to an application correctly required or did not require a PIN, but also whether while the application was in use the system was able to maintain the correct level of authentication or to de-authenticate the user as expected.

Precision and recall. We start by reporting the precision and recall for each model. Precision is defined as the fraction of correct predictions across all testing samples that resulted in the same prediction, while recall is defined as the fraction of correct predictions across all testing samples with the same ground truth label. As Figure 6(b) shows, SVM and decision tree outperform linear regression, which instead presents many incorrect predictions when the ground truth is private or confidential. This could lead to a disproportionate number of both FRs and FAs, so we discard it. We choose to use SVM



(a) Precision (b) Recall

Figure 6: Precision and recall of different machine learning models when trained with 8 user traces and tested with the remaining one. The average and standard deviation is across the 9 users. No loss function is used.

Table 4: Confusion matrix for the SVM model.

	Rec. as Public	Rec. as Priv.	Rec. as Conf.
Public	92.50%	7.44 %	0.06%
Private	25.02%	74.42%	0.56%
Confidential	37.31%	0.00 %	62.69%

as the default inference model in our system because it is the most secure: it is able to recognize 92.5% of the public labels (high recall) which implies very few false authentications, and shows precision higher than 81% for all labels (Figure 6(a)). These graphs consider inferences done when both the authentic user and the attacker used the system. Since we saw in Table 2 that no unauthorized accesses occurred (*i.e.*, cases of false authentications when an attacker tried to access the system), all false authentications of SVM happened for authentic users and were due to incorrectly binning a user's access as private/confidential instead of public. Decision tree is a more aggressive model which presents fewers FRs for confidential (higher recall for confidential), but generates more FAs (lower recall for public).

Table 4 further characterizes the results for SVM. It provides a complete breakdown of test outputs, including which states were incorrectly inferred when an incorrect inference was made (*i.e.*, how the difference between 100% and the height of the bar in Figure6(b) is broken down). We observe that most of the errors are false rejections for the confidential state: when the ground truth is confidential (last row in the table), the model labels it as public 37% of the time (first column) and it never labels it as private (second column). When inferring private states, it labels private states as public 25% of the time and almost never as confidential. The higher accuracy in inferring private states is an artifact of our traces and the reliability of the sensor signals our implementation uses: the number of private ground truth labels (42.6%) was much higher than the confidential ones (2.4%), thus making the model better trained for inferring private labels.

From these results two observations follow. First, we do expect users to place commonly used applications in the private application bin more often than in the confidential one, hence requiring higher convenience for private and higher security for confidential. Second, one could argue that for the confidential level, the system should simply adopt password-based authentication and avoid making any inference. Our model does not exclude this option, but in an implementation with a larger number of more reliable signals, the system would be more likely to automatically detect the confidential level thus making the machine learning approach still useful at this level. Overall, these results show high potential in using machine learning models such as SVM in this context. False authentications are less than 8% and restricted to authentic users, and the system is able to infer the private/confidential states about 70% of the time.

Feature Importance. Next, we evaluate which features more strongly contributed in training the inference model. Table 5 shows the relative importance of features for SVM (in WEKA, this parameter is called GainRatioAttributeEvaluator: "it evaluates the worth of an attribute by measuring the gain ratio with respect to the class"). Features are shown in rank order (second column), along with their respective gain ratio (third column). The feature rank shows the importance of the corresponding authentication signals: possession (ProxAuthDev and ProxAuthDevConf), continuity (LastPlacementDuration), secrets (TimeSincePIN), and biometric (TimeSinceOwnerVoice). All types of signals envisioned for progressive authentication contributed to the inference model. Particularly, these results confirm the im-

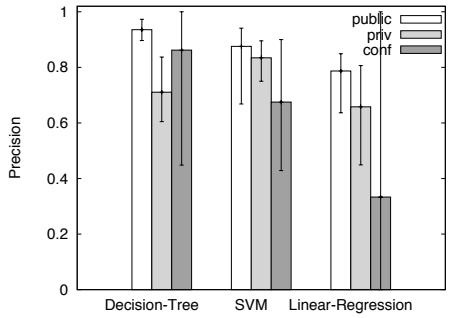
Table 5: Relative feature importance for SVM

Feature rank	Feature name	Gain ratio
1	ProxAuthDev	0.16105
2	LastPlacementDuration	0.09785
3	TimeSincePIN	0.04879
4	ProxAuthDevConf	0.04584
5	TimeSinceOwnerVoice	0.04554
6	TimeSinceProx	0.03919
7	TimeSinceTouch	0.02802
8	TimeSinceSound	0.02618
9	LastPlacement	0.02529
10	TimeSinceTable	0.02264
11	Placement	0.01849
12	TimeSinceHands	0.0174
13	TimeSinceNonOwnerVoice	0.01505
14	TimeSincePocket	0.01456
15	Speaker	0.00983
16	SpeakerConf	0.00907
17	PlacementDuration	0.00884
18	PlacementConf	0.00000

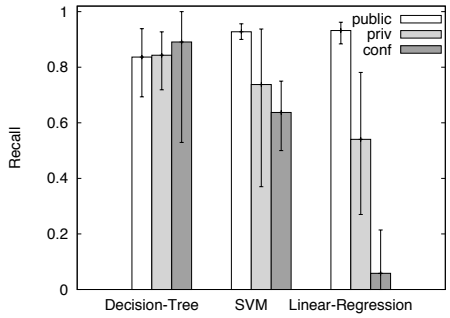
portance of recency and continuity of significant events that indicate user authenticity, such as being close to a known PC, having been in contact with the phone recently, having a PIN entered, or "hearing" the owner's voice. However, the fact that in this specific implementation of progressive authentication ProxAuthDev is the top ranked feature does not mean that progressive authentication works only in scenarios where the user is nearby a PC. Instead, not surprisingly, the features ranked first are those derived from the most reliable sensor signals. For instance, ProxAuthDev is derived from BT-based proximity, activity detection, and PIN events.

7.4 Model personalization

We have so far evaluated high-level processing models trained across users because these are the models we expect to ship with the phones. However, it is possible to collect additional data from specific users to further tailor the model to their behavior. To assess the benefits of personalization, we evaluate the models when trained with data of a single user and tested through leave-one-out cross-validation. Figure 7 reports average precision and recall of these "personalized" models. Compared to the models trained across users (see Figure 6), SVM's recall remains the same while the precision is slightly worse. Decision tree and linear regression show slightly better performance, but these modest improvements do not justify the overhead of training personal models. Perhaps with much longer traces for individual users we could see



(a) Precision



(b) Recall

Figure 7: Precision and recall of different models when tested through leave-one-out cross-validation. Average and standard deviation across users are shown. No loss function is used.

higher improvements, but otherwise these results confirm that building accurate generalized models that can be shipped with the phone is feasible.

7.5 Low-level processing accuracy

The low-level processing models include placement detection, face recognition and voice identification. Figure 8 reports the average accuracy (across 9 users) and variance for each recognition model. The accuracy is computed as the ratio of correct recognitions out of the total number of model invocations.

In the current prototype, **placement** can be in one of three possible states: "hands", "table", or "pocket". Although they do not cover all possible states, we started with three states that are fairly common and sufficient to cover the scenarios in the script. To evaluate the placement detector accuracy, each participant performed a series of actions that put the phone in each of these states multiple times. In the meantime, sensor measurements were recorded: accelerometer samples were taken every 20 milliseconds, temperature and humidity every second, and light every 100 milliseconds. The placement recog-

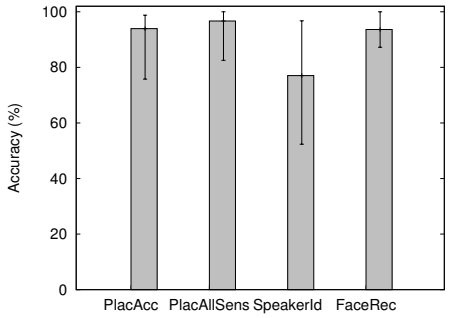


Figure 8: Average accuracy of placement, voice and face recognition models. Two versions of the placement model are tested: PlacAllSens relies on external Gadgeteer sensors and PlacAcc does not.

Table 6: Confusion matrix for the PlacAllSens model.

	Recogn. as Hand	Recogn. as Table	Recogn. as Pocket
Hand	98.78%	1.22%	0.003%
Table	0.01%	98.43%	1.56%
Pocket	0.12%	5.82%	94.06%

Table 7: Confusion matrix for voice identification.

	Rec. as Owner's voice	Rec. as Other's voice or Unknown
Owner's voice	77.0%	23.0%
Other's voice	0.4%	99.6%

nition model was invoked every half second. We measured the accuracy of two decision tree models, one that uses only accelerometer data (PlacAcc), and another that is augmented with temperature/humidity and light sensors (PlacAllSens).

Figure 8 shows that the model accuracy varies from 83% to 100% when all sensors are used, and from 76% to 99% if only accelerometers are used. Without the added Gadgeteer sensors, the model still provides high accuracy. As Table 6 shows, the most common error consists of confusing a pocket placement with a table placement. Due to the prototype bulky form factor, the light sensor was sometimes left outside of the pocket, biasing the system towards a "table" state. The "hands" state was rarely confused with other states because of the high reliability of the temperature/humidity sensor (used to detect a human hand).

For **face recognition**, a webcam mounted on a desktop PC's screen collected pictures of the participants. First, users were asked to look at the center of the screen (4 pictures) and then to look at 8 different points around the perimeter of the screen (2 pictures for each point). We trained a user-specific model using these 20 photos. For testing, a video was recorded while users read text on the screen and watched a ball moving in a figure of eight pattern across the entire screen. Roughly 1,000 frames were extracted from the video and used for testing. The whole recording was performed indoors and in the same lighting conditions (since it was from a desktop webcam). Figure 8 shows a 94% accuracy and a small variance across users. In scenarios with changing light conditions it is likely that the face recognition accuracy would decrease, but for office environments where lighting conditions are more stable this was not a problem. Nevertheless, when face recognition did not work we observed a small impact on the overall accuracy of our system because the face recognition signals are cross-checked against other presence signals collected using activity detection sensors (*e.g.*, login/logout events, typing on keyboard or moving a mouse) which can be estimated with much higher reliability.

Finally, we evaluated the accuracy of the **speaker identification** model [21]. In the study, the participants' voice was recorded for roughly 10 minutes using the phone's microphone at 16kHz, 16bit mono format. All samples were collected in an office environment where the user's voice was the dominant signal and there was modest background noise such as people talking in the hall, keyboard typing, or air conditioning. The phone was placed within 1 to 2 meters from the participant. The participant was asked to speak or read some text (4 of them only read text, 1 only spoke, and 4 alternated between reading and speaking). This model requires multiple voices for training, so we created a single model with the samples we collected. The model was trained using 2 minutes from each of the users and the voice of all 9 participants was tested against the model (*i.e.*, a user's voice may be recognized as somebody's else voice or as unknown). Compared to the other models in Figure 8, speaker identification varied the most across users. There were 3 participants for whom the average accuracy was roughly 59.3%, while for all the other users the model achieved at least 83.7% accuracy. However, Table 7 shows that the system rarely recognized another voice as the owner's voice (0.4%) – false positives were rare. Most of the errors were due to the system not recognizing the user (23%) – false negatives. In the actual study (the second part), these errors were unfortunately amplified, perhaps due to the variety of conditions in which the phone was used (the speaker was at a variable distance from the user, sometimes the phone was in a pocket while the user spoke, different background noise, etc.).

Table 8: Power consumption and execution time on a Samsung Focus WP 7.1 for 4 different power configurations.

Conf	Sensing (mW)	Comput (mW)	Comm (mW)	TotalPower (mW)	ExTime (sec)
LocalMin	<1	41	0	42	0.20
Local	≈160	447	44	651	0.23
LocalRemote	≈160	71	94	325	0.99
Remote	≈160	49	98	307	1.50-2.81

7.6 Latency and power consumption

We evaluate latency and power overhead using a 2-device configuration including a WP 7.1 Samsung Focus using WiFi for communication and a Windows PC (2.66GHz Intel Xeon W3520 CPU with 6GB of RAM). We consider four device configurations:

- *LocalMin*: low-level and high-level processing runs on the phone, however power-consuming tasks (BT-based proximity detection and voice identification/recognition) are disabled.
- *Local*: all low-level and high-level processing tasks depicted in Figure 3 run on the phone.
- *LocalRemote*: computation-light low-level processing runs on the phone while voice identification and high-level processing run on the PC (*i.e.*, gray-colored modules in Figure 3 are offloaded).
- *Remote*: both low-level and high-level processing runs on the PC. The phone only runs the sensors and sends raw measurements to the PC.

We measured the power drained from the phone battery by connecting it to a Monsoon Power Monitor [26], specifically designed for Windows phones. The Monsoon meter supplies a stable voltage to the phone and samples the power consumption at a rate of 5KHz. During the measurements, the phone had the display and WiFi on, which corresponds to an idle power consumption of 896 mW. Table 8 shows power consumption and execution time results for all four configurations. LocalMin is the best configuration from both a power consumption and latency point of view: it consumes only 42 mW and has an average delay of 200 msec. However, as this configuration disables features derived from proximity and voice recognition, its model may provide lower accuracy compared to the other 3 configurations, which allow for the complete set of features. Specifically, LocalMin's SVM model is able to reduce the number of PINs by 53%, but it presents 70% UAs to private applications. On the other hand, the model still provides 0% UAs to confidential applications. Although less accurate, this model still provides advantages compared to an unprotected system, thus being an option for users who currently do not use a PIN on their phone.

Among the other 3 configurations, the best compromise from a power-latency point of view is Local-Remote – this is also the default configuration of our system. Its delay is less than 1 second and it consumes about 325 mW, which may be acceptable for modern phones. The reason for such a reduction in power consumption compared to Local is that this configuration offloads voice identification to the PC thus significantly reducing the power drained by the phone's CPU. Basically, the phone uploads voice recordings only if voice is detected (*i.e.*, it does not upload raw data if no voice is present). Local represents the fastest full configuration with an average execution time of 225 msec per inference, including feature extraction (≈ 90 msec) and SVM invocation (≈ 115 msec). This configuration may be convenient for a phone at home or in an office, with full power or currently connected to a power charger.

We have shown a range of options. Executing on the client is faster, using the server for remote execution yields lower power costs. Temporarily disabling computation-intensive features can also significantly lower power consumption at the cost of accuracy. By switching between these configurations and possibly even temporarily disabling progressive authentication, we can deliver a system with lower authentication overhead and acceptable power and latency requirements. In general, users who currently do not use PINs on their phones can have a much more protected system without the need to worry about power consumption.

8 Related work

Other researchers have explored work related to progressive authentication in the following areas: multi-level authentication systems, context-based and automatic authentication, and mobile device authentication in general.

8.1 Multi-level authentication

Multi-level authentication has been considered before. As in progressive authentication, data and applications are categorized in different levels of authorization, variously called "hats" [34], "usage profiles" [19], "spheres" [32], "security levels" [4], or "sensitive files" [36]. With the exception of TreasurePhone [32] and MULE [36], most of this work has been conceptual, with no actual implementation. TreasurePhone divides applications into multiple access spheres and switches from one sphere to another using the user's location, a personal token, or physical "actions" (*e.g.*, lock-

ing the home door would switch from the "Home" to the "Closed" sphere). However, these sphere switching criteria have flaws. First, location is rather unreliable and inaccurate, and when used in isolation, it is difficult to choose the appropriate sphere (*e.g.*, being alone at home is different than being at home during a party). Second, the concept of personal tokens requires users to carry more devices. Third, monitoring physical "actions" assumes that the device can sense changes in the physical infrastructure, something that is not yet viable. Conversely, progressive authentication enables automatic switching among the multiple levels of authentication by relying on higher-accuracy, simpler and more widely available multi-modal sensory information. MULE proposes to encrypt sensitive files stored in laptops based on their location: if the laptop is not at work or at home, these files are encrypted. Location information is provided by a trusted location device that is contacted by the laptop in the process of regenerating decryption keys. Progressive authentication protects applications, not files, and it uses multiple authentication factors, unlike MULE, which uses location exclusively.

8.2 Automatic authentication

Other forms of automatic authentication use a single authentication factor such as proximity [6, 7, 18], behavioral patterns [33], and biometrics, such as typing patterns [1,24], hand motion and button presses [3]. Most of these techniques are limited to desktop computers, laptops or specific devices (*e.g.*, televisions [3]). The closest to our work is Implicit Authentication [33], which records a user's routine tasks such as going to work or calling friends, and builds a profile for each user. Whenever deviations from the profile are detected, the user is required to explicitly authenticate. Progressive authentication differs from this work in that it uses more sensory information to enable real-time, finer granularity modeling of the device's authentication state. On the other hand, any of those proximity, behavioral and biometric patterns could be plugged into our system. Transient authentication [6, 7] requires the user to wear a small token and authenticate with it from time to time. This token is used as a proximity cue to automate laptop authentication. This approach requires the user to carry and authenticate with an extra token, but its proximity-based approach is relevant to our work in that it also leverages nearby user-owned devices (*i.e.*, the tokens) as authentication signals.

8.3 Mobile device authentication

The design of more intuitive and less cumbersome authentication schemes has been a popular research topic. Current approaches can be roughly classified into knowledge-based, multi-factor, and biometric authentication techniques. All three are orthogonal to progressive authentication. Our goal is not to provide a new "explicit" authentication mechanism, but instead to increase the usability of current mechanisms by reducing the frequency at which the user must authenticate. When explicit authentication is required, any of these techniques can be used.

Knowledge-based approaches assume that a secret (*e.g.*, a PIN) is shared between the user and the device, and must be provided every time the device is used. Due to the limited size of phone screens and on-screen keyboards, this can be a tedious process [5], especially when it is repeated multiple times per day. In multi-factor authentication, more than one type of evidence is required. For instance, two-factor authentication [2,31,35] requires a PIN and secured element such as a credit card or USB dongle. This practice presents major usability issues, as the need to carry a token such as SecurID [31] goes against the user's desire to carry fewer devices. Biometric schemes [5, 16, 27] leverage biometrics [17] or their combinations [12, 15], such as face recognition and fingerprints, to authenticate the user with high accuracy. Even though very secure, biometric identification comes with acceptability, cost and privacy concerns [27], and is especially cumbersome on small devices.

9 Conclusions

We presented a novel approach to progressively authenticate (and de-authenticate) users on mobile phones. Our key insight is to combine multiple authentication signals to determine the user's level of authenticity, and surface authentication only when this level is too low for the content being requested. We have built a system prototype that uses machine learning models to implement this approach. We used the system in a lab study with nine users and showed how we could reduce the number of explicit authentications by 42%. We believe our results should make this approach attractive to many mobile users who do not use security locks today. Overall, progressive authentication offers a new point in the design of mobile authentication and provides users with more options in balancing the security and convenience of their devices.

10 Acknowledgments

We thank the authors of Speaker Sense for making it available to us. Special thanks to Eiji Hayashi for many insightful discussions on the architecture of progressive authentication and for designing its user interface. Thanks to Asela Gunawardana for helping us with the

machine learning models used by the system, and to Tom Blank, Jeff Herron, Colin Miller and Nicolas Villar for helping us instrumenting the WP with additional sensors. We also thank David Molnar and Bryan Parno for their comments on an earlier draft of this paper, and the anonymous reviewers for their insightful feedback. Finally, we are especially grateful to all users who participated in our user studies.

References

[1] Bergadano, F., Gunetti, D., and Picardi, C. User authentication through keystroke dynamics. *ACM Trans. Inf. Syst. Secur. 5* (November 2002), 367–397.

[2] C.G.Hocking, S.M.Furnell, N.L.Clarke, and P.L.Reynolds. A distributed and cooperative user authentication framework. In *Proc. of IAS '10* (August 2010), pp. 304–310.

[3] Chang, K.-H., Hightower, J., and Kveton, B. Inferring identity using accelerometers in television remote controls. In *Proc. of Pervasive '09* (2009), pp. 151–167.

[4] Clarke, N., Karatzouni, S., and Furnell, S. Towards a Flexible, Multi-Level Security Framework for Mobile Devices. In *Proc. of the 10th Security Conference* (May 4–6 2011).

[5] Clarke, N. L., and Furnell, S. M. Authentication of users on mobile telephones - A survey of attitudes and practices. *Computers and Security 24*, 7 (Oct. 2005), 519–527.

[6] Corner, M. D., and Noble, B. Protecting applications with transient authentication. In *Proc. of MobiSys '03* (2003), USENIX.

[7] Corner, M. D., and Noble, B. D. Zero-interaction authentication. In *Proc. of MobiCom '02* (2002), ACM, pp. 1–11.

[8] Cuervo, E., Balasubramanian, A., Cho, D.-k., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. MAUI: making smartphones last longer with code offload. In *Proc. of MobiSys '10* (2010), ACM, pp. 49–62.

[9] Gadgeteer. http://netmf.com/gadgeteer/.

[10] Giurgiu, I., Riva, O., Juric, D., Krivulev, I., and Alonso, G. Calling the cloud: Enabling mobile phones as interfaces to cloud applications. In *Proc. of Middleware'09* (November 30 - December 4 2009), Springer.

[11] How Apple and Google will kill the password. http://www.computerworld.com/s/article/9206998/How_Apple_and_Google_will_kill_the_password_.

[12] Greenstadt, R., and Beal, J. Cognitive security for personal devices. In *Proc. of the 1st ACM workshop on AISec* (2008), ACM, pp. 27–30.

[13] Hayashi, E., Riva, O., Brush, A., Strauss, K., and Schechter, S. Goldilocks and the Two Mobile Devices: Going Beyond All-Or-Nothing Access to a Device's Applications. In *Proc. of SOUPS '12* (July 11-13 2012), ACM.

[14] Holmes, G., Donkin, A., and Witten, I. Weka: A machine learning workbench. In *Proc. of the 2nd Australia and New Zealand Conference on Intelligent Information Systems* (December 1994), pp. 357–361.

[15] Hong, L., and Jain, A. Integrating faces and fingerprints for personal identification. *IEEE Trans. Pattern Anal. Mach. Intell. 20* (December 1998), 1295–1307.

[16] Jain, A., Bolle, R., and Pankanti, S. *Biometrics: Personal Identification in a Networked Society*. Kluwer Academic Publ., 1999.

[17] Jain, A., Hong, L., and Pankanti, S. Biometric identification. *Commun. ACM 43* (February 2000), 90–98.

[18] Kalamandeen, A., Scannell, A., de Lara, E., Sheth, A., and LaMarca, A. Ensemble: cooperative proximity-based authentication. In *Proc. of MobiSys '10* (2010), pp. 331–344.

[19] Karlson, A. K., Brush, A. B., and Schechter, S. Can I borrow your phone?: Understanding concerns when sharing mobile phones. In *Proc. of CHI '09* (2009), ACM, pp. 1647–1650.

[20] Liu, Y., Rahmati, A., Huang, Y., Jang, H., Zhong, L., Zhang, Y., and Zhang, S. xShare: supporting impromptu sharing of mobile phones. In *Proc. of MobiSys '09* (2009), ACM, pp. 15–28.

[21] Lu, H., Brush, A. J. B., Priyantha, B., Karlson, A. K., and Liu, J. SpeakerSense: Energy Efficient Unobtrusive Speaker Identification on Mobile Phones. In *Proc. of Pervasive 2011* (June 12-15 2011), pp. 188–205.

[22] Mobile wallet offered to UK shoppers. http://www.bbc.co.uk/news/technology-13457071.

[23] Ni, X., Yang, Z., Bai, X., Champion, A. C., and Xuan, D. Diffuser: Differentiated user access control on smartphone. In *Proc. of MASS '09* (12-15 October 2009), IEEE, pp. 1012–1017.

[24] Nisenson, M., Yariv, I., El-Yaniv, R., and Meir, R. Towards behaviometric security systems: Learning to identify a typist. In *Proc. of PKDD '03* (2003), Springer, pp. 363–374.

[25] 24% of mobile users bank from a phone. Yet most don't have security measures in place. http://www.bullguard.com/news/latest-press-releases/press-release-archive/2011-06-21.aspx.

[26] Monsoon Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[27] Prabhakar, S., Pankanti, S., and Jain, A. K. Biometric recognition: Security and privacy concerns. *IEEE Security and Privacy 1* (2003), 33–42.

[28] Priyantha, B., Lymberopoulos, D., and Liu, J. LittleRock: Enabing Energy Efficient Continuous Sensing on Moble Phones. Tech. Rep. MSR-TR-2010-14, Microsoft Research, February 18 2010.

[29] Priyantha, B., Lymberopoulos, D., and Liu, J. LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones. *IEEE Pervasive Computing 10* (2011), 12–15.

[30] Reynolds, D. A. An overview of automatic speaker recognition technology. In *Proc. of ICASSP '02* (2002), vol. 4, pp. IV–4072–IV–4075.

[31] RSA SecurID. http://www.rsa.com/node.aspx?id=1156.

[32] Seifert, J., De Luca, A., Conradi, B., and Hussmann, H. TreasurePhone: Context-Sensitive User Data Protection on Mobile Phones. In *Proc. of Pervasive '10*. 2010, pp. 130–137.

[33] Shi, E., Niu, Y., Jakobsson, M., and Chow, R. Implicit authentication through learning user behavior. In *Proc. of ISC '10* (October 2010), pp. 99–113.

[34] Stajano, F. One user, many hats; and, sometimes, no hat – towards a secure yet usable PDA. In *In Proc. of Security Protocols Workshop* (2004).

[35] Stajano, F. Pico: No more passwords! In *Proc. of Security Protocols Workshop* (March 28–30 2011).

[36] Studer, A., and Perrig, A. Mobile user location-specific encryption (MULE): using your office as your password. In *Proc. of WiSec '10* (2010), ACM, pp. 151–162.

[37] Texas Instruments. OMAP^{TM} 5 mobile applications platform, 13 July 2011. Product Bulletin.