MultiLock: Biometric-Based Graded Authentication for Mobile Devices

Shravan Aras
shravanaras@cs.arizona.edu
University of Arizona
Tucson, Arizona

Chris Gniady
gniady@cs.arizona.edu
University of Arizona
Tucson, Arizona

Hari Venugopalan
hvenugopalan@ucdavis.edu
University of California, Davis
Davis, California

ABSTRACT

While traditionally smartphones have relied on methods such as a passcode or pattern-based authentication, biometric authentication techniques are gaining popularity. However current biometric methods are heavily dependent on various environmental factors. For example, face authentication methods depend on lighting conditions, camera shake and picture framing, while fingerprint scanning relies on finger placement. All of these variables can result in these systems becoming time-consuming for the user to use. To remedy these problems, we propose MultiLock, a passive, graded authentication system, which uses face authentication as a case study to propose a system that gives users access to their devices without requiring them to manually interact with the lock screen. MultiLock allows a user to categorize applications into various security bins based on their sensitivity. By doing so MultiLock can grant users access to different sensitivity applications, based on varying degrees of sureness that the device is being used by its rightful owner. Thus, allowing the device to be used even in adverse lighting conditions without hampering user experience. In our tests, MultiLock was able to grant access to users for 88% of the interactions on average, while passively running in the background. While we use face authentication as an example to demonstrate and propose MultiLock, our system can be used with any confidence based biometric system.

CCS CONCEPTS

• **Security and privacy → Mobile platform security**.

KEYWORDS

authentication, mobile devices

ACM Reference Format:
Shravan Aras, Chris Gniady, and Hari Venugopalan. 2019. MultiLock: Biometric-Based Graded Authentication for Mobile Devices. In *MOBIQUITOUS '19: EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, November 12–14, 2019, Huston, TX* . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

MOBIQUITOUS'19, November 12–14, 2019, Huston, TX

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Smartphones have become ubiquitous in today's world. They are used everywhere, for communication, entertainment, commerce, navigation and so on. We depend on our phones for all tasks ranging from trivial time checking to complex high-security banking operations. This growing reliance on smartphones for everyday tasks has also resulted in the need for better authentication in order to secure access to the smartphone and to prevent unauthorized use of applications sensitive to privacy such as banking or communication. There are various forms of authentication available to users ranging from simple memorization of pins, passwords or patterns to biometric forms of authentication. While proper authentication is indispensable, this step requires users to perform additional tasks before they can use the phone for even trivial tasks such as weather checking.

Currently, the most popular means of securing smartphones from unauthorized access is by using a pin or password. However, many users find these methods cumbersome, and some opt to disable them as a result. In a study conducted by Breitinger and Nickel [5], where they surveyed a total of 550 participants, 40% of the participants who left their phone unprotected did so because they felt it was more convenient than setting a pin, while 3% of participants indicated that it was too difficult to memorize a pin. Owing to these inconveniences various biometric forms of authentication have been proposed [4], with face-based authentication and fingerprint scanning proving to be more widely accepted than others. Fingerprint scanning, however, requires additional hardware which has led to a slower adoption rate. Face detection, on the other hand, can make use of existing front-facing camera on smartphones.

The primary aim of a face authentication system is to confirm or deny the identity claimed by a person, thereby allowing them access to the protected resource if validated. This is in contrast to a face recognition system which attempts to establish the identity of a given person out of a closed pool of individuals. While most face authentication systems use well-established algorithms [2, 7, 12, 24], in common, they introduce a confidence parameter which determines the security of the system. Typically smartphones require the user to authenticate once, while unlocking the phone, by positioning his/her face in front of the front-facing camera until the device positively detects the user or falls back to a more traditional approach such as pin or password. The tolerance for false positive in such systems is controlled by varying the confidence threshold. Higher the confidence threshold, lower the probability of an intruder being mistaken as an authorized individual. Successful authentication is also dependent on other factors such as the time for which the user faces the camera, good lighting, proper alignment, unaltered look (such as sunglasses, hats, etc.) and so on. Thus, when the user has

to be authenticated, all these factors force the user to deviate from normal interaction with the phone.

In this paper, we propose MultiLock framework that reduces the burden of facial authentication, as outlined above, while still providing highly secure authentication for sensitive applications. MultiLock relies on a key observation that not all applications require the same level of security. In everyday scenarios, we use a variety of applications, and not all of them are perceived equally sensitive by their users. While users may deem applications related to finance or privacy like emails as sensitive, applications like games may not be thought as equally sensitive. This observation allows us to define varying levels of security for different applications, a concept referred to as graded security or multilevel security [13]. Applying this approach MultiLock allows users to quickly unlock the phone and access low sensitivity applications while monitoring user's faces and authenticating them in the background. We argue that by the time the user accesses a more sensitive application, potentially scrolling through several application pages on their phones, MultiLock will have plenty of time to observe the user's face and authenticate to the level required by the more sensitive applications. Subsequently this paper makes the following contributions: (1) We propose a graded security authentication framework using facial authentication to improve usability of the phones; (2) We show that majority of the time users do not access sensitive application, which completely eliminated any inconvenience due to authentication; (3) We show that for sensitive applications, a majority of the authentication effort can be performed in the background. It is important to note that our approach is not just limited to face recognition, but can be applied to any confidence based system, where the true user guarantee can be quantified. Upcoming technologies like iris and fingerprint scanning, which determine if the true user is using device by producing a confidence value, are other potential candidates for this approach.

2 MOTIVATION

A considerable advantage of using biometric authentication methods on smartphones is the lack of need for the user to recall passcodes or gestures to unlock the system. By using biometric authentication methods such as facial recognition, users can ideally, unlock their phones instantly by merely looking at it. However, in practice using facial recognition as an authentication method can be cumbersome for users and take considerable time to unlock the device. In a study conducted by Trewin et. al. [23] it was shown that the median user response time for using facial authentication was 5.55 seconds. The majority of the time was used to provide the system with a sample for authentication (not including processing time) as the facial recognition relies heavily on users correctly positioning the phone camera such that their complete face was visible, stabilizing the camera and having the ambient lighting that was adequate to provide an acceptable image.

To illustrate the above observation, we wrote a custom application on an android device, to take photos using the front-facing camera while the users went about their normal day interacting with the phone and compare it to best lighting conditions and best face positions. Figure 1.a shows an image sample taken under perfect lighting conditions, with the camera held at eye level and kept still. While Figure 1.b and Figure 1.c show image samples which were taken while users interacted with their phones. In order to test the fidelity of existing facial authentication system, we used Google's default face authentication lock screen, using a Motorola X 2014 device running Android 5.1.1. We first trained the system using a subject's face, and then held the 3 test images in Figure 1 one at a time in front of the phone camera. We observe that while the image in (a) can instantly unlock the phone both images (b) and (c) fail to unlock the phone. We also conducted these experiments with 2 other subjects, which yielded similar results. This simple example illustrated that to unlock the phone, the user needs good lighting condition and proper face position, which may be difficult or cumbersome for users.

(a) (b) (c)

Figure 1: Image samples taken under perfect lighting condition(a), and while user was interacting with his phone(b,c) to test unlocking Google's default facial authentication lock screen.

This negatively affects user experience as shown by Trewin et. al. where facial authentication resulted in 3.1% failure to acquire rate (FTA), with a system usability score (SUS) of 75% when compared to 78% for password-based authentication [23]. Failure to acquire [8] is a metric used in user studies of biometric authentication methods to measure the failure to provide a sample of sufficient quality. In case of facial authentication, it indicates the number of sample images that do not contain good enough data, due to bad lighting conditions, for the verification algorithm to work on. It is interesting to note that Trewin et. al. carried out these experiments in a laboratory setting, with fixed lighting conditions. We believe that real-life interactions would lead to a higher FTA, due to considerable variations in lighting conditions. On the other hand, System Usability Score [6] indicates how users perceive the simplicity of using a system and is calculated based on questionnaire. A higher value indicates that the system is seen as more usable. The primary reason why users preferred password over face authentication as indicated by SUS was that it was time-consuming and cumbersome to face the camera, deviating from the normal workflow of using the device.

While providing security for accessing a phone is important, it may not be necessary for all types of interactions. For example, a user checking current weather conditions on their phone has a much different security implication than user accessing banking or email applications. Dorflinger et. al. [9] have shown that different user groups (distinguished by age, gender, and profession), perceive

Users	Values			Percentage		
	High	Mod	Low	High	Mod	Low
1	8	2	56	12.1%	3%	84%
2	56	4	253	17.9%	1.2%	80%
3	21	228	200	4.7%	50%	44%
4	28	8	105	19.8%	5.6%	74%
5	59	6	190	23.1%	2.3%	74%
6	74	9	307	19%	2.3%	78.7%
7	0	231	59	0%	79.6%	20.3%
8	44	4	160	21%	2%	77%

Table 1: Distribution of interactions with various applications in traces, categorized into different security levels.

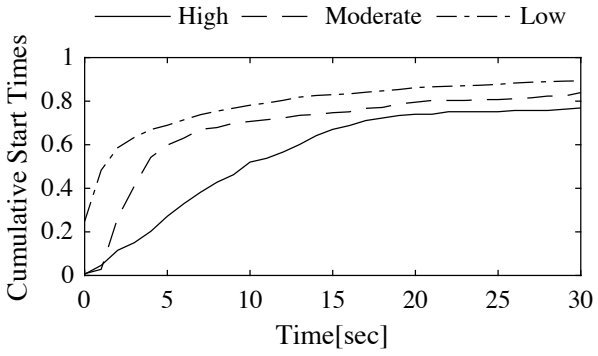


Figure 2: Cumulative distribution of application start times after screen on, categorized into various security levels as indicated by users, for all‘ users.

the sensitivity and hence the need for security of similar applications differently. In order to understand how users interact with the applications on their devices, we collected usage traces from 8 different users, recording the applications they started along with their respective timestamps. These traces were collected over a period of 2 days, with a mix of weekend and weekdays. Users were also asked to rate the applications they use into three categories (high, moderate, and low) based on how important the security of the applications are for them. Users marked applications as high if they believed an unauthorized access to those applications would lead to financial or privacy loss. This often included applications related to banking, instant messaging, social media, emails, etc. Applications considered noncritical by users were marked as low, these often included games, media players, news readers, etc. On the other hand, users often rated applications such as file explorers, sharing apps such as Dropbox, or applications which stored fitness or health data as moderate.

Table 1 shows the distribution of application interaction occurrences (number of times users interacted with different applications), marked as either high, moderate or low, as indicated by their respective users. The variation in interaction count directly relates to how often users used their devices. We observe that interactions with high sensitivity applications on average account for only 15% of total interactions. This relatively small percentage of high sensitivity application invocations suggests that users spend a considerable about of time using applications which do not require a high degree of authentication. On the other hand, we see a large number of invocations for low sensitivity applications, 78% on average for all users except user 3 and 7. In case of both user 3 and 7, it is interesting to note how they have 50%, 80% moderate sensitivity application interactions respectively, while the others have less than 6% of it. This relates back to Dorflinger et. al. [9] observation, where different users perceive security differently. In user 3's case not only did we see a large number of social media, instant messaging applications, and emails but unlike other users, he marked them as moderate level of security. Similarly, while other users marked media applications such as Spotify, Hulu, and Netflix to be low security, user 3 perceived them to be of moderate security.

It is also common for users to first interact with applications of low sensitivity before accessing high sensitivity applications. Figure 2 shows the cumulative distribution of time users took to start different sensitivity groups of applications once the screen was turned on. We observe that for all users, 80% of applications which fall into the high sensitivity category are opened after 4 seconds once the screen has been turned on and 60% are opened after 7.5 seconds. Similarly, 80% of moderate sensitivity applications are opened after 1.5 seconds after the screen has been turned on. It is interesting to note that there were several instances in which applications were opened after 30 seconds, or even after several minutes in some cases. This occurred when these applications were not the first to be opened after the screen was turned on but rather when users shifted to them after using other applications. For example, we observed that some users would open an email application after having spent several minutes watching YouTube videos. On the other hand, low sensitivity applications were opened almost immediately after the screen was turned on as users would quickly check the time or the weather.

From the above analysis, we draw two key insights. First, there is significant amount of applications that do not require security measures. Second, there is a considerable amount of time before users open high or moderate sensitivity applications. Those insights lead us to propose performing facial authentication in the background, while users interact with low security application. The goal of our approach is to mitigate this obtrusiveness of facial based authentication and improve user experience.

3 DESIGN OF MULILOCK

The primary design goal of MultiLock is to provide users with fast and unobtrusive access to their devices while providing desired application security. MultiLock provides graded authentication by varying the false acceptance rate (FAR) according to the application sensitivity level designated by the user. Applications labeled by the user with high sensitivity will have a lower FAR providing better user validation, while applications having lower sensitivity will have higher FAR allowing for faster and lower level of user validation. MultiLock consists of two components: Facial Authentication and Security Manager. The system components and the operation flow is shown in Figure 3. **Facial Authentication** mechanisms

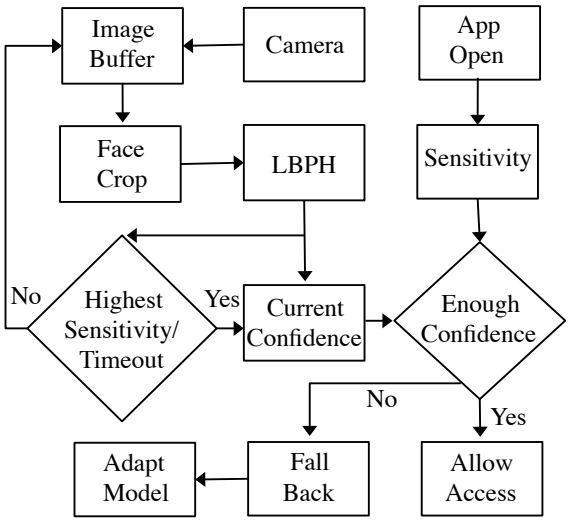


Figure 3: Flowchart of the MultiLock system.

utilize the phone's front-facing camera to search for the user's face and perform authentication until the desired confidence value is reached. **Security Manager** monitors application interactions and validates the security level from Facial Authentication phase, needed to interact with the given application.

3.1 Facial Authentication

Facial authentication in MultiLock is performed continuously from the time the user turns the screen on until the highest level of authentication is obtained. At this point, the user can interact with any application without interruptions. MultiLock captures images from the front-facing camera at a rate of 5 fps. This frame rate balances the authentication processing overhead and face image capture in case the user does not directly face the camera, in which case a face may appear only periodically in the camera view. Since user perception threshold is 50-100ms [22], which states that the user needs up to 100ms to notice the change in output, we set the frame capture period to 200ms arguing the user needs another 100ms before the information is processed and the user reacts to alter screen content. The images are stored in a circular buffer as shown in Figure 3 before being processed by the face crop phase.

The face crop phase isolates the user's face from superfluous background content in preparation for the actual face authentication step. To achieve this, the system makes use of Haar Cascade face detection technique provided by Open Computer Vision Library (OpenCV) [19], an open-source cross-platform library for performing real-time computer vision. The cropped image is then passed onto the face recognition algorithm. In our current implementation, we make use of the Local Binary Patterns Histogram (LBPH) algorithm for face recognition, however our design can easily be adapted to support other algorithms by adding them to this stage. LBPH focuses on extracting local features from images as proposed by Ahonen, T. Hadid et al in [2]. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The features extracted in this manner exhibit a low dimensionality. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. It takes a pixel as center and thresholds its neighbors against it. If the intensity of the center pixel is greater or equal to its neighbor, then it denotes it with a 1 else a 0. This generates a binary value for each pixel referred to as the Local Binary Pattern codes, and a histogram is created with these values for the entire image.

The histogram of the captured image is used to determine the likelihood that captured images belong to the authorized user of the device, by comparing the histogram of the captured image with those from a pre-trained LBPH model of the authorized user's face. More precisely it calculates the chi-square distance between the sample histogram and histograms in the model (a histogram corresponding to each image in the model), this distance is known as the *confidence value*. This resulting confidence value indicates how closely the captured image matches the pre-trained model. The lower the confidence value, the higher the likelihood of the user accessing the device is the authorized user. The initial model is trained upon first invocation of the MultiLock system, and the authorized user is asked to face the camera under good lighting conditions to capture the images which are face cropped and fed into the OpenCV LBPH model trainer. To provide alternate face angles to the trainer the user is also asked to hold the phone in a typical position they interact with it, which is usually looking down at the phone. These two scenarios account for majority of interaction positions for the user facing the camera.

3.2 Security Manager

The security manager is responsible for granting users access to various applications based on the results of facial authentication. The confidence value returned by the facial authentication indicates how closely the sample image matches the trained model. Before we can proceed with authenticating and granting access to the applications, we need to map confidence values to the corresponding application sensitivity levels - high, moderate, and low. To accomplish that, we make use of two widely used metrics in face recognition literature, false rejection rate(FRR) and false acceptance rate(FAR). For a given confidence value, FAR indicates the number of cases when the model falsely accepts an impostor image, while FRR indicates the number of instances when a true user is rejected by the model. Both FAR and FRR values are related to the corresponding confidence value. Choosing a low confidence value would decrease FAR, thus making the system more secure. However, a lower confidence value also means less tolerance to changes in lighting and facial features, and it can lead to an increase in FRR, thus making the model too conservative.

We use the MUCT [16] facial database, a public domain research database of faces from the University of Cape Town, and split the images into various groups as outlined in the Laussanne [15] protocol to calculate confidence values corresponding to the security levels. The MUCT database contains a total of 3755 images, obtained from 276 individuals, capturing different angles of the subjects face using 5 cameras and under varying lighting conditions. We split

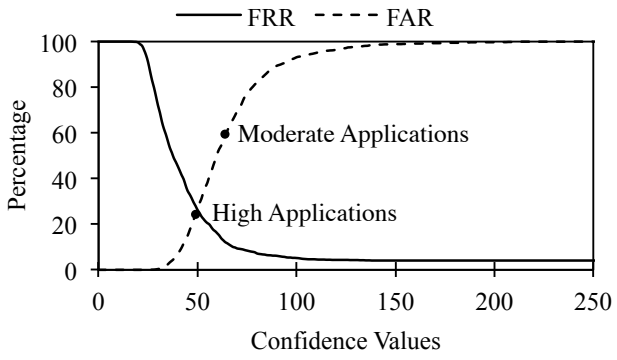


Figure 4: FRR and FAR values for different confidence values, from running LBPH on MUCT database, annotated with confidence values we use for different application sensitivity.

the images into 2 groups - set of images corresponding to 83 individuals into an impostor group, whereas images of the remaining 193 individuals were added to the client group. The client group served as a source for generating the training model, and also for providing sample images to obtain the FRR values. On the other hand, the impostor set was used to extract images for calculating FAR values. We then generated client training models for each of the 193 individuals using images from a single front-facing camera (1 of the five camera angles in the MUCT database) and their corresponding images under various lighting conditions. The remaining images (4 camera angles and their respective lighting condition variations) from the client set where used as test images against the trained model to obtain FRR values. Figure 4 shows how FAR and FRR values for our test setup differ for various confidence values.

Ideally, we would want critical applications to have zero tolerance against importers gaining access to them, i.e. we would choose a confidence value for critical applications where FAR=0. However, this would make the model too conservative and would reject all true images, as seen from Figure 4. Thus, we set our confidence threshold for critical applications to a value where FAR=FRR, as shown in the Figure 4. This gives us FAR value of 25% and a corresponding confidence value of 50. However, unlike high-security applications, determining a confidence value for moderate applications is debatable and comes down to user preference. While some users might perceive the security of moderate applications to be closer to high applications, others might be content with much looser security guarantees, allowing them to gain quicker access to the applications. Thus, as shown in Figure 4 this value can range from 50 to 150 where 150 offers the least security with 100% FAR.

For our current implementation, we pick the default value as the midpoint between 25% and 100% FAR, i.e. a confidence value corresponding to 62% FAR. The corresponding confidence value of 66 will; therefore, give equal importance to both user preferences. Finally, low sensitivity applications are trivial. These often consist of applications which require no security to unlock, for instance clock, weather, news, etc. We don't set a predefined confidence value for these and allow them to be opened by any user. These predefined confidence values are then used by MultiLock to grant or deny access to various applications with different sensitivity levels that were designated by the user.

3.3 MultiLock Operation

We start MultiLock as soon as the user turns on the screen. As Facial Authentication continues to perform the image acquisition steps in the background, it generates a steady stream of confidence values. Whenever the user opens an application, as indicated in Figure 3, MultiLock first searches for that applications sensitivity in its database. If the application is not found, i.e. the user has opened this application for the first time, MultiLock asks the user to mark it either as high, moderate or low. Users can also change this at a later stage by accessing the setting menu inside MultiLock. MultiLock then determines the required confidence value for the application based on its sensitivity as outlined in Section 3.2. If this confidence value is equal to or greater than the confidence value returned by the image recognition phase, the user is granted access to the application else the user is requested to authenticate either by better face position or through a secondary authentication method such as passcode.

While our simplified assumption so far was that MultiLock runs continuously until highest confidence level is reached, we need to take a closer look at the termination requirements to prevent it from negatively impacting system performance and battery life. Subsequently, MultiLock has three terminating conditions for Facial Authentication. First, as we described so far, the Face Authentication phase is terminated when a confidence value of 50 or lower is obtained, since a confidence value of 50 indicates that MultiLock can grant users access to applications with the highest sensitivity, as shown in Figure 4. Second, the Face Authentication phase is terminated when a user requests to open an application which requires a lower confidence value (higher sensitivity) than what Face Authentication has been able to acquire. In this scenario, MultiLock prompts the user for a better face position or lighting, or falls back to a secondary authentication, resulting in the highest authentication level. Third, there may arise a scenario when neither the first or second conditions are met. For example, when a user interacting with low sensitivity application under poor lighting condition, the Face Authentication may attempt to run as long as the user continues interacting with it, degrading the battery life. To avoid such situation, MultiLock relies on a timeout period to shut down Facial Authentication. We select the timeout period based on the distribution of application start times, as observed in our user studies shown in Figure 2. We observe that there is an 80% chance that users will start high or moderate sensitivity applications before 30 seconds from the time the screen is turned on. This scenario depicts how users would normally interact with their phone, and we have used a 30-second timeout value in our current implementation.

3.4 Dynamic Training Optimization

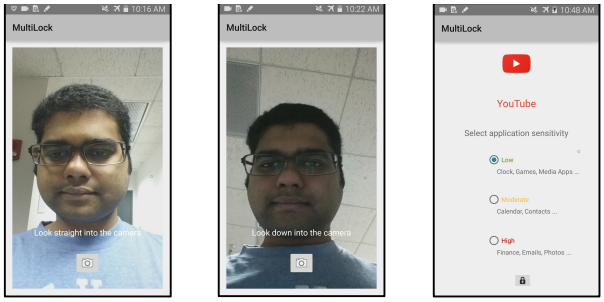
The adaptability of MultiLock to changing user condition or face position angles with respect to the camera is critical in providing a robust system that will minimize the fallback on secondary authentication. We augment MultiLock with dynamic retraining to update the authorized user's trained face model to better cope with varied lighting conditions or positions that users use their devices

in. Dynamic retraining occurs when MultiLock fails to provide authentication as described in the second termination case above. Once users authenticate themselves using the alternate method, the images captured just before the secondary authentication method is used to update the model. Not all lighting conditions can result in beneficial retraining and MultiLock only performs retraining if the Haars Cascade method is able to detect a face in the set of the collected image captures. Since the LBPH model maintains histograms corresponding to the images inside the model, adding another model image only requires the appending a new histogram, a fast and efficient operation that does not require the complete rebuilding of the model. Further, we must also deal with the case when the true user unlocks the phone using secondary authentication (due to Mutlilock failing), and then hands the device to another person. In order to prevent Multilock from retraining using the wrong face we empirically defined a retraining threshold. We calculated this retraining threshold by running user models against true user sample images and then against advisory images (where the face does not match) and then clustered these values. We found that when an advisory face was presented, the LBPH values were in the range of 110 - 171. On the other hand when the model was presented with the true user's face, the values where much lower in the range of 38-47. We thus set our retraining threshold at 47 based on our observations.

Finally, to prevent the model from growing in size indefinitely, we limit the model size to 16 MB and employ an LRU(least recently used) eviction policy for histograms in the model. This memory overhead is small and less than 1% of the overall memory in a typical smartphone, which is 2 GB in size or larger. We implement LRU by associating a timestamp with each histogram in the model. Whenever LBPH returns a confidence value, it also outputs the histogram inside the trained model which was the closest match to the sample image and we associate the current timestamp with this histogram. The histogram with the oldest timestamp is evicted once the memory limit is reached. Given the small memory footprint, we are able to store up to 54 image histograms, for images corresponding to 480p inside the model. A significant number before we need to evict old unused models. Subsequently, we selected the timestamp implementation instead of a typical stack implementation (as in case of LRU), since evictions are infrequent and stack manipulations would be slightly costlier. Finally, we refer to this variant as MultiLock Dynamic, whereas the one without retraining is referred to as MultiLock Static in subsequent sections.

3.5 Implementation Details

We decided to implemented MultiLock for testing as an android application, thus allowing MultiLock to run on non-rooted devices and without any OS modifications. We used a Samsung Note3 running Android 5.0.0 for testing MultiLock implementation. Both the image acquisition phase and the security manager run as independent services, using Android Binder to communicate. The image acquisition service uses an image size of 480p to take images in the background. To detect when an application comes to the foreground, we make use of Android accessibility features. Each application produces various accessibility events, one of them being *TYPE_WINDOW_CONTENT_CHANGED* along with the package



(a) (b) (c)

Figure 5: Screenshots of the MultiLock application. (a) and (b) capturing training samples, (c) prompts the user to select application sensitivity.

name that produced it. We simply check for the package name in our database and open an activity requesting the user to rate the application sensitivity, if we cannot find it. If on the other hand, the application exists in the database, we either allow activity with the fallback authentication mechanism, if security manager determines the application is too sensitive to grant access or give user access to the application. Figure 5 shows screenshots of MultiLock in action. In order to carry out face recognition and cropping tasks, we integrated the OpenCV binaries into our application using the OpenCV SDK for Android [18]. In order to increase user security, we automatically switch to the home screen when the phone's display is turned off. This has no performance impact on the user, since their current application is pushed onto the system memory in pause state and not killed. Both Figure 5.a and Figure 5.b show one of our users taking images for training the model. The application instructs them to take images while looking straight into the camera, in Figure 5.a and another one while looking down at it Figure 5.b. Figure 5.c shows the interface when an application is invoked, who's sensitivity MultiLock is not aware of. Along with the 3 options of low, moderate and high, we also provide footnotes describing example type of applications for those categories. This makes it easier for users to choose the most appropriate one. Finally, as MultiLock runs in the background as a service, we show a small camera-shaped icon in the notification bar, whenever MultiLock is running, to notify the users about it.

3.6 Threat Model

We use a threat model to analyze the security of Multilock. This helps us identify, quantify, and address the security risks associated with Multilock. The threat model is divided into 3 sections - External Entities, Stride Thread List, and Countermeasures List.

External Entities. We refer to all components which are external to the code/implementation of Multilock and can pose a threat to it. These components are not fully under the control of Multilock execution environment.

- The user currently interacting with the mobile device.
- The OpenCV Haar cascades are used for face detection. The Haar descriptions are read from an XML file.

- On training, the user model is stored in a private file. The model is updated for Multilock dynamic.
- The security preferences of the user are loaded from an application-specific private database. The database stores the security levels designated by the user for each application on the device.

STRIDE Threat List. We categorize the threats using STRIDE - Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege. We also specify the set of security controls that could prevent these threats from harming the system.

- Type: Spoofing
  Description: An attacker could try to tamper with the trained user model file or the Haarcascade. This could result in false positives as well as false negatives.
  Security Control: Authentication.
  Countermeasures: Store trained model file and Haarcascade in Android application-specific private storage. We rely on Android's sandbox security model to safeguard our files from unauthorized access by other applications.
- Type: Elevation of Privileges
  Description: An attacker could change the security levels of different applications. The attacker could lower the security level of a sensitive application and try to obtain access with lesser confidence.
  Security Control: Authorization.
  Countermeasures: We store the security levels in Android application Sqlite database. Android ensures that this database content can only be accessed by the same application that created it. Thus preventing unauthorized access from other applications.

4 EXPERIMENTAL RESULTS

In order to validate MultiLock, we collected interaction traces from 8 users over a period of 2 days. We choose 8 users to include a mix of graduate students and professors. These users were asked to install our custom application which recorded the user's face using the front-facing camera when they interacted with their phone. We also recorded the applications that users opened during this session. Both of these events were triggered after the users turned on ftheir screen and were augmented with timestamps to match interactions videos to exact application openings. We also gave a brief explanation to each of the users about how our application worked and provided them with an option to disable video and application monitoring if needed. This setup allowed us to collect real-life interaction traces from users throughout the day, thus giving us a realistic set of traces to validate MultiLock. Finally, we provided each of the users with their respective list of applications, and asked them to categorize them into - high, moderate, and low sensitivity applications.

Figure 6 shows the profiling system for power measurements of MultiLock. We connected Samsung Note 3 directly to the power supply and removed the battery to eliminate variability of the battery voltage as it discharged. We used National Instruments PCI-6230 digital acquisition board to measure voltage drop across the resistor and calculate the resulting power. To measure energy consumed by

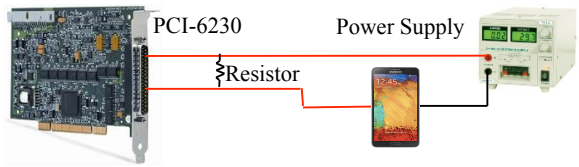


Figure 6: Profiling Hardware.

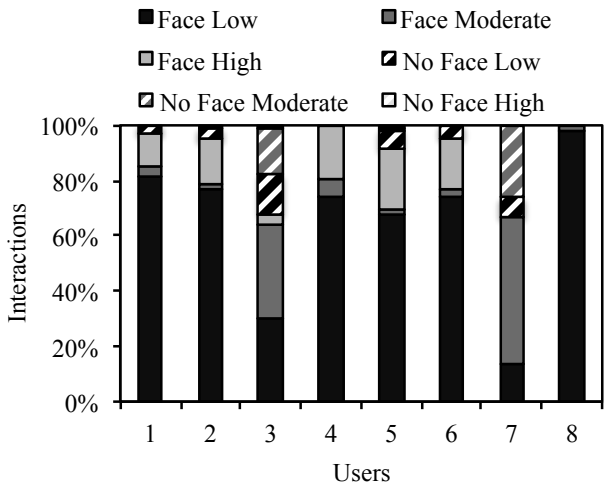


Figure 7: Distribution of application security levels in interactions with and without faces.

MultiLock, we first measured the baseline power of the phone. To do so we put the device in airplane mode, thus disabling all radios; disable screen auto-brightness; and interact with the phone as one would normally do after unlocking it (open apps, swipe through home screen, pull down notifications bar). Next, we performed the exact same steps, however with MultiLock running in the background. We found that the baseline power was 438mW while the power when MultiLock was running was 479mW. Subtracting these two, gave us 41mW which was the power demanded by MultiLock while it ran in the background. To put this number into perceptive, an LTE radio, which has become a standard component of modern smartphones and is heavily used demands 1.6W of active power and 70mW when idle. Thus we argue that a power demand of 41mW is not significant enough to have an adverse effect on the smartphones battery life.

4.1 Interaction Characteristics

As MultiLock's ability to unlock devices depends on the camera capturing images of a user's faces while interacting with the device, we split the traces into 2 categories - those which contained a face, and those which did not. A trace is deemed to not contain a face if the Haar-Cascade mechanism we used for face detection fails to find one in the interaction session. We found that most prominent reason for the absence of a face in a trace is device usage in complete darkness, for example in a dark room or outdoors at night. We summarize our results in Figure 7 which shows the distribution of user

interactions with different levels of application sensitivity (High, Moderate, and Low), with and without faces (Face, No Face). We observe that for the majority of the user interactions, except user 3 and 7, there is a face present 95% of times on average. Furthermore, the sessions which do not contain a face, are largely or entirely dominated by low sensitivity application interactions. As MultiLock does not require low sensitivity applications to pass through the face authentication phase, users would be able to easily open these without having to fall back upon time-consuming secondary authentication methods. For instance, while user 3 has relatively fewer interaction sessions with a face, 65% when compared to 95% on average for other users, 14% of their non-face sessions invoked only low sensitivity applications.

We argue that for traditional face authentication system, user 3 and 7 would have to fall back on secondary authentication for all interactions without a face, which amounts to 31% and 33% respectively. MultiLock, on the other hand, would take advantage of its graded security design, and only require the user to use secondary authentication for interactions with higher security levels, which are only 17% and 26% respectively of the overall interactions. Thus MultiLock is highly beneficial even in scenarios with adverse lighting condition as seen in the case of user 3 and 7. Subsequently, in the next few sections, we focus solely on interactions which contain faces in them to analyze how MultiLock performs.

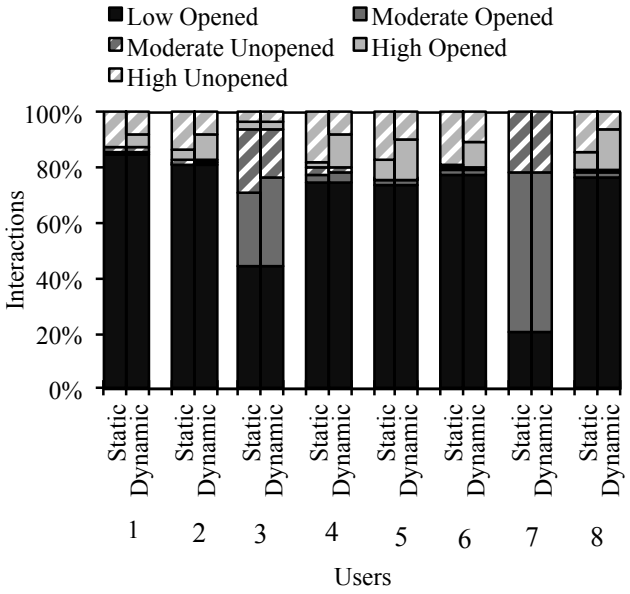


Figure 8: Distribution of application sensitivity MultiLock Static and MultiLock Dynamic can open.

4.2 Granting Application Access

In order to analyze the degree to which MultiLock is able to grant users access to applications of various sensitivities, we run it on the interaction traces obtained from users. Further, we divide our analysis into MultiLock Static, when no dynamic retraining is performed, and MultiLock Dynamic, which performs retraining as outlined in Section 3.4.

Figure 8 shows the distribution of interactions for each user that MultiLock was able to open, as well as those that required MultiLock to fall back on secondary authentication. In the case of both MultiLock static and MultiLock dynamic, low sensitivity applications are opened without the need for facial authentication. We observe that in case of MultiLock static, as shown in Figure 8 we were able to server 81.1% of user interactions on average across all users without requiring to fall back on secondary authentication. Out of these, on an average 2.8% and 11.5% of the interactions correspond to high and moderate sensitivity applications respectively, while the remaining 66.7% of interactions involve low sensitivity applications. Further, unlike most of the users, a large number of interactions for user 7 are related to moderate sensitivity applications. It is interesting to note that even in the case of user 7, for which 79.4% of the interactions where for moderate sensitivity applications, MultiLock static is still able to open 58% of those interactions without any user intervention.

We can see further improvements when dynamic retraining is added to MultiLock, as shown by MultiLock Dynamic in figure 8. As MultiLock Dynamic, updates the model every-time a user is forced to fall back on secondary authentication provided the conditions outlined in Section 3.4 are met, it adapts to various adverse lighting conditions thus giving us an overall better performance. This improvement is most notable for high sensitivity applications. Where MultiLock static opened 18.1% of the total high sensitivity application interactions on average, MultiLock Dynamic is able to open 47.5% of the total high sensitivity applications. This improvement is more than double that of MultiLock Static. This improvement is most notably seen in user 4, who had a large number interactions both indoors and outdoors, thus giving rise to some rather unique lighting conditions. In this case, MultiLock Dynamic is able to open 2.6 times more high sensitivity applications automatically, when compared to its static counterpart. Overall, MultiLock Dynamic allows the user to open more high and moderate sensitivity applications under adverse lighting conditions without the need to fall back on secondary authentication. Finally MultiLock Dynamic had an average success rate of 88% of overall interactions without requiring users to change their normal use flow to face the camera directly as is the case with traditional face authentication methods.

4.3 Background Execution

MultiLock runs as a background process, while users interact with their device, and ideally, we would like it to run for as short duration as possible to reduce any potential impact on battery life and performance. Figure 9 presents background execution time of MultiLock Dynamic. The runtime is capped by our timeout period of 30 seconds. While the runtime is affected by several factors such as lighting conditions, types and time of applications that users interacted with, 50% of interactions across all users required the algorithm to run only 8.2 seconds on an average. We also observe that for both user 1 and 3, 80% of their interactions required the algorithm to run for 11 seconds or less. In case of user 1, the interactions were relatively short, accessing the phone to interact with low sensitivity applications and then immediately turning off the screen. As for user 3, most of the interactions were for moderate sensitivity

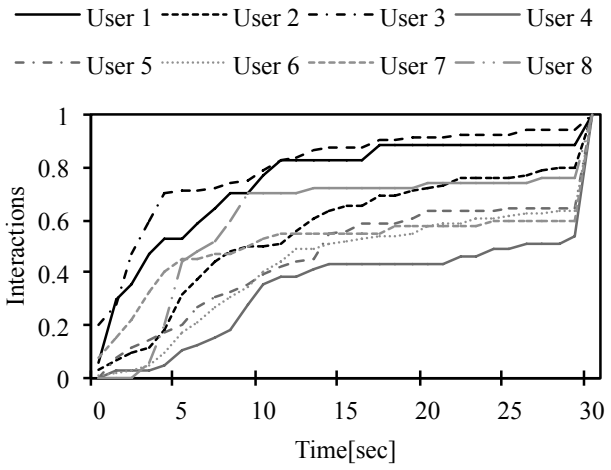


Figure 9: Cummulative distribution of how long MultiLock runs in the background for each user.

and MultiLock was able to establish a confidence threshold well below the threshold for high sensitivity applications (one of the termination conditions). In case of the remaining users, MultiLock was required to run longer in the background, due to poor lighting conditions when users interacted with low sensitivity applications. Subsequently, we argue that in majority situations MultiLock maintains a low overhead by terminating early by authenticating the user, while being bound by our timeout interval, preventing it to run indefinitely under unfavorable lighting conditions.

5 RELATED WORK

There has been a considerable amount of research in using graded security on mobile devices. Ben-Asher et.al. [3] surveyed 465 participants to understand how users perceived the security of various applications and researched the use of various biometric methods for graded authentication. Similarly Dorflinger et.al. [9] conducted a survey using focus groups to determine user willingness to use graded security. They found that different users perceive the security of same resources differently. Overall, users stated it would be inconvenient to use different biometric methods to protect different resources. MultiLock; however, uses only facial recognition to achieve graded security, thus not impacting user convenience. The idea of using varying levels of authentication for protecting different resources on a system is fairly old and well studied. Traditionally, systems like UNIX [20] relied on explicitly setting user permissions, delegating users to groups and setting permissions on files to offer a level of graded security access to the complete system.

Researchers in the past have also proposed various passive sensing techniques to authenticate users and provided graded access to various applications, based on their sensitivity. Zhu et.al. proposed SenSec [26], a system which would constantly run in the background, collecting data from phone sensors, and authenticate the users based on how they use their phone, via a gesture model. Similarly Feng et.al. [11] make use of touch input (pinching, zooming, etc.) as the user interacts with the touch screen to continuously authenticate the user in the background. Their approach; however, requires a custom sensor glove to complement the data from smartphone. Xu et.al. [25] also proposed a system that monitored user interactions in the background using touch interactions, training a SVM [14] and using it to authenticate users. Other background authentication methods proposed such as SenGaurd [21], also make use location and accelerometer data to perform passive authentication. Methods like KeySens [10] have also been proposed which authenticate users automatically based on their typing pattern, using the on-screen keyboard. Niinuma et.al. [17] proposed a posture invariant system to passively authenticate users using a laptop camera, as they worked on it. However, unlike laptops, which are stationary on the desk or lap, mobile phones suffer from constant camera shake as users interact with them. This makes the problem of authentication even more challenging for MultiLock due to lack of focus in most frames. Additionally, recording video continuously in the background would also negatively affect battery life of a handheld device.

Multilock is agnostic to the underlying face recognition algorithm used. We currently use the popular Local Binary Pattern Histogram (LBPH) for face recognition, by Ahonen et. al. [1], primarily due to its computation simplicity as our system is designed to run on mobile devices. An algorithm similar to the LBPH that was implemented is the Active Appearance Model (AAM) [7]. An Active Appearance Model (AAM) is an integrated statistical model which combines a model of shape variation with a model of the appearance variations in a shape-normalized frame. An AAM contains a statistical model if the shape and gray-level appearance of the object of interest which can generalize to almost any valid example. Matching to an image involves finding model parameters which minimize the difference between the image and a synthesized model example projected into the image. However, this method is more computationally expensive than LBPH for cellular devices.

Other popular face recognition algorithms that can be employed include Eigenfaces method and the Fischerfaces method. The Eigenfaces described in [24] uses Principal Component Analysis (PCA) to reduce the high dimensional image into lower dimensions along which there is maximum variance. The Eigenfaces method does not take class labels into account, and hence if variance is generated from an external source, the axes with maximum variance need not necessarily contain any discriminative information, and classification becomes impossible. The Fischerfaces method as proposed by Sir R. A. Fischer in [12] takes class labels into account as well and identifies the lower subspace with maximum variance across each class.

While a plethora of research has been conducted in the field of face recognition, passive user authentication, and methods to provide graded security, to the best of our knowledge using facial authentication in the background to provide graded security for users has not been done before. With the approach proposed in MultiLock, smartphones can authenticate their users automatically, varying the severity by different application types, using well-proven and tested facial authentication algorithms.

6 CONCLUSIONS

Biometric authentication mechanisms such as face-based authentication have always been cumbersome for users to use since they required users to deviate from their normal workflow and stare at the camera. The problem has been exasperated due to variable lighting conditions while users use their devices, which prevents the system from obtaining a satisfactory sample for face recognition algorithms. In this paper, we have analyzed how users interacted with their devices after turning on the screen and proposed to apply a graded security mechanism dividing the applications up into low, moderate and high sensitivity. Further we observed that users interacted with high sensitivity applications for only 15% of their interactions on average. We used this observation to propose the design of MultiLock, a passive, graded authentication mechanism which runs in the background using face recognition, thus freeing users of the burden of manually unlocking the device. With MultiLock we allow users to access different sensitivity applications by varying the sureness that the device is being used by its true user. The lower the application sensitivity the less sure MultiLock is during authentication that the device is being used by a true user. Even though MultiLock is not tied to a particular face recognition algorithm, we implemented it as an android application, and used the Local Binary Pattern Histogram method for face recognition, due to its computation simplicity, allowing it to run on mobile devices with minimum overhead. In our trace-driven analysis using MultiLock, we found that 88% of all the interaction traces across users could be opened based on the application sensitivity without any user intervention. We also measured MultiLock's power consumption to be only 41mW when running in the background, allowing it to run comfortably in the background without draining device battery. Finally, we believe that MultiLock design described in this work provides users with a hassle-free passive authentication system, based on well researched and ever-improving field of face-based authentication techniques.

REFERENCES

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. 2006. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28, 12 (2006), 2037–2041.

[2] Hadid A. Ahonen T. and Pietikainen. 2004. Face Recognition with Local Binary Patterns. Computer Vision. (2004).

[3] Noam Ben-Asher, Niklas Kirschnick, Hanul Sieger, Joachim Meyer, Asaf Ben-Oved, and Sebastian Möller. 2011. On the need for different security methods on mobile phones. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 465–473.

[4] Debnath Bhattacharyya, Rahul Ranjan, Farkhod Alisherov, and Minkyu Choi. 2009. Biometric authentication: A review. *International Journal of u-and e-Service, Science and Technology* 2, 3 (2009), 13–28.

[5] Frank Breitinger and Claudia Nickel. 2010. User Survey on Phone Security and Usage.. In *BIOSIG*. 139–144.

[6] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[7] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. 2001. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2001), 681–685.

[8] Lorrie Faith Cranor and Simson Garfinkel. 2005. *Security and usability: designing secure systems that people can use.* " O'Reilly Media, Inc.".

[9] Tim Dörflinger, Anna Voth, Juliane Krämer, and Ronald Fromm. 2010. "My smartphone is a safe!" The user's point of view regarding novel authentication methods and gradual security levels on smartphones. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*. IEEE, 1–10.

[10] Benjamin Draffin, Jiang Zhu, and Joy Zhang. 2013. Keysens: Passive user authentication through micro-behavior modeling of soft keyboard interaction. In *Mobile Computing, Applications, and Services*. Springer, 184–201.

[11] Tao Feng, Ziyi Liu, Kyeong-An Kwon, Weidong Shi, Bogdan Carbunar, Yifei Jiang, and Ngac Ky Nguyen. 2012. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 451–456.

[12] R. A. Fischer. 1936. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* 7, 2 (1936), 179–188.

[13] Dieter Gollmann. 2010. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 5 (2010), 544–554.

[14] Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *Intelligent Systems and their Applications, IEEE* 13, 4 (1998), 18–28.

[15] Jiri Matas, Miroslav Hamouz, Kenneth Jonsson, Josef Kittler, Yongping Li, Constantine Kotropoulos, Anastasios Tefas, Ioannis Pitas, Teewoon Tan, Hong Yan, et al. 2000. Comparison of face verification results on the XM2VTFS database. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, Vol. 4. IEEE, 858–863.

[16] Stephen Milborrow, John Morkel, and Fred Nicolls. 2010. The MUCT landmarked face database. *Pattern Recognition Association of South Africa* 201, 0 (2010).

[17] Koichiro Niinuma and Anil K Jain. 2010. Continuous user authentication using temporal information. In *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 76670L–76670L.

[18] opencv dev team. 2018. Open Computer Vision Android SDK. http://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html. (2018).

[19] opencv dev team. 2018. Open Computer Vision Library. http://opencv.org/. (2018).

[20] Dennis M Ritchie and Ken Thompson. 1974. The UNIX time-sharing system. *Commun. ACM* 17, 7 (1974), 365–375.

[21] Weidong Shi, Feng Yang, Yifei Jiang, Feng Yang, and Yingen Xiong. 2011. Senguard: Passive user identification on smartphones using multiple sensors. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*. IEEE, 141–148.

[22] Ben Shneiderman and Catherine Plaisant. 1987. Designing the user interface: Strategies for effective human-computer interaction. (1987).

[23] Shari Trewin, Cal Swart, Larry Koved, Jacquelyn Martino, Kapil Singh, and Shay Ben-David. 2012. Biometric authentication on a mobile device: a study of user effort, error and task disruption. In *Proceedings of the 28th Annual Computer Security Applications Conference*. ACM, 159–168.

[24] M. Turk and A. Pentland. 1991. Eigenfaces for Face Detection/Recognition. (1991).

[25] Hui Xu, Yangfan Zhou, and Michael R Lyu. 2014. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *Symposium On Usable Privacy and Security (SOUPS 2014)*. 187–198.

[26] Jiang Zhu, Pang Wu, Xiao Wang, and Juyong Zhang. 2013. Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 1128–1133.