CORMORANT: On Implementing Risk-Aware Multi-Modal Biometric Cross-Device Authentication For Android

Daniel Hintze
daniel@hintze-it.de
Johannes Kepler University Linz
Linz, Austria

Matthias Füller
matthias.fueller@fhdw.de
FHDW Paderborn
Paderborn, Germany

Sebastian Scholz
sebastian.scholz@fhdw.de
FHDW Paderborn
Paderborn, Germany

Rainhard D. Findling
rainhard.findling@aalto.fi
Aalto University
Espoo, Finland

Muhammad Muaaz
muhammad.muaaz@ins.jku.at
Johannes Kepler University Linz
Linz, Austria

Philipp Kapfer
philipp@kapfer.co.uk
Johannes Kepler University Linz
Linz, Austria

Wilhelm Nüßer
wilhelm.nuesser@fhdw.de
FHDW Paderborn
Paderborn, Germany

René Mayrhofer
rene.mayrhofer@jku.at
Johannes Kepler University Linz
Linz, Austria

ABSTRACT

This paper presents the design and open source implementation of *CORMORANT*, an Android authentication framework able to increase usability and security of mobile authentication. It uses transparent behavioral and physiological biometrics like gait, face, voice, and keystrokes dynamics to continuously evaluate the user's identity without explicit interaction. Using signals like location, time of day, and nearby devices to assess the risk of unauthorized access, the required level of confidence in the user's identity is dynamically adjusted. Authentication results are shared securely, end-to-end encrypted using the Signal messaging protocol, with trusted devices to facilitate cross-device authentication for co-located devices, detected using Bluetooth low energy beacons. *CORMORANT* is able to reduce the authentication overhead by up to 97% compared to conventional knowledge-based authentication whilst increasing security at the same time. We share our perspective on some of the successes and shortcomings we encountered implementing and evaluating *CORMORANT* to hope to inform others working on similar projects.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**; **Usability in security and privacy**; *Biometrics.*

ACM Reference Format:
Daniel Hintze, Matthias Füller, Sebastian Scholz, Rainhard D. Findling, Muhammad Muaaz, Philipp Kapfer, Wilhelm Nüßer, and René Mayrhofer. 2019. CORMORANT: On Implementing Risk-Aware Multi-Modal Biometric Cross-Device Authentication For Android. In *17th International Conference on Advances in Mobile Computing & Multimedia (MoMM '19), December 2–4, 2019, Munich, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3365921.3365923

MoMM '19, December 2–4, 2019, Munich, Germany

ACM ISBN 978-1-4503-7178-0/19/12…$15.00
https://doi.org/10.1145/3365921.3365923

1 INTRODUCTION

Mobile devices like smartphones and notebooks that allow convenient access to valuable assets, information and services have long become an indispensable part of everyday life. Small and mobile, those devices have a high propensity to become lost or stolen. Strong user authentication is thus crucial to protect against the risk of unauthorized access, today commonly in form of knowledge-based mechanisms like PIN, pattern, and password. However, these authentication techniques require a significant amount of scarce user attention in proportion to the usually short usage sessions [23], which is further amplified by the inability of current approaches to scale with the ever growing number of devices used simultaneously. One out of three smartphone users thus chooses to not enable authentication, with inconvenience being cited as the primary reason as to why [1, 10, 16, 17].

Different promising approaches to reduce the burden of explicit authentication have been explored. Transparent biometrics can verify the user's identity unobtrusively [28, 46, 50] using traits like gait [39], mouth motions [52], heartbeat [56], breathing acoustics [4], voice [42], mouse movement [15], and keystroke dynamics [29]. Risk estimation can be used to dynamically adjust the security settings to apply as much security as needed but as little as possible [19, 27]. And with the number of interconnected devices owned and used by a single individual growing, extending the authentication scope in order to leverage contextual and biometric information gathered within a group of trusted devices [26, 51] is another approach to improve usability by reducing explicit authentication whilst also improving security.

The concept of *CORMORANT*, a mobile authentication system that combines all three approaches, has been recently proposed by Hintze et al. [22]. It uses transparent behavioral and physiological biometrics to continuously evaluate the user's identity without explicit interaction. The risk of unauthorized access derived from

signals like location, time of day and nearby devices is used to dynamically adjust the required level of confidence in the user's identity. Risk and authentication results are shared with trusted devices to facilitate cross-device authentication for co-located devices. Conducting a large-scale agent-based simulation based on more than 720 000 days of real-world device usage and 6.7 million simulated robberies and thefts sourced from police reports, the authors found that the proposed system reduces the frequency of password entries required on smartphones by up to 97.82% whilst simultaneously reducing the risk of unauthorized access in the event of a crime by 97.72%, compared to conventional knowledge-based authentication.

In this paper, we summarize the practical implementation of the system proposed in [22] and share our experiences and lessons learned in the hope to inform and inspire others working on similar projects. Our contributions are as follows:

- We present the architecture and implementation of an open source Android framework that dynamically combines explicit and implicit authentication mechanisms with continuous risk estimation, shared securely across a group of trusted devices to enhance usability and increase security.
- We demonstrate the feasibility of authentication plugins based on gait, voice, face, and keystroke dynamics.
- We share our experience and lessons learned from designing, implementing, and evaluating a complex distributed asynchronous authentication system.

2 RELATED WORK

It has been shown previously that risk can successfully be incorporated into mobile device authentication. Hocking et al. [26] developed an *Authentication Aura* that leverages the latent security potential contained in surrounding devices and possessions in everyday life. A similar approach comes from Chow et al. [5] who propose a combination of multi-device and risk-aware authentication in which the security settings are dynamically adjusted based on the presence of other trusted devices in the vicinity. Another example is Context-Aware Scalable Authentication (CASA) [19] which dynamically adjusts the required strength of authentication based on contextual information like time and location.

Multi-modal biometric systems, i.e., systems incorporating biometric information from multiple sources, have been successfully applied to the domain of mobile devices to overcome some drawbacks of unimodal biometrics [44]. Examples are face, teeth and voice authentication on mobile devices [31, 53]. In particular transparent biometrics can be used to reduce explicit authentication. Crawford et al. [7] showed that by using keystroke dynamics and speaker verification on mobile devices, explicit authentication could be reduced by 67%. Closed to our approach comes *Itus* [30], a framework for implicit authentication on mobile devices which shares a number of goals with our approach, for instance it enables other researchers to integrate novel authentication mechanisms into the framework.

With the number of mobile devices people carry and use simultaneously, sensing and taking leverage of nearby trusted devices for the purpose of multi-device authentication becomes increasingly important. An early contribution in this direction comes from Varshavsky et al. [54], who proposed to use knowledge of the shared radio environment as proof of physical proximity to authenticate co-located devices. To avoid password challenges, Stajano [51] introduced *Pico* which unlocks a device in the presence of k-out-of-n small encrypted hardware tokens called *Picosiblings*. A preliminary architecture for a cross-device authentication system utilizing multiple biometrics as well as risk has been proposed in [20] and later expanded in [21, 24, 25]. A system for behavior-based, transparent multi-factor authentication has been patented by Gordon et al. [14]. It shares some key characteristics with our approach, for instance using different transparent biometrics across multiple devices, but differs from ours in that it relies on convolutional deep neural networks to learn subject-specific features.

The highest resemblance with our concept has the framework for progressive authentication on mobile phones by Riva et al. [43], which reportedly achieves a 42% reduction in required explicit authentication attempts. Like CORMORANT, it supports continuous multi-modal biometrics, including face and voice recognition. It takes nearby devices into account, though only rudimentary. CORMORANT is different from this work in that the set of biometrics used is dynamic at runtime, a more sophisticated notion of multi-device authentication is used, and that risk is taken into account for making security decisions.

3 ARCHITECTURE

3.1 Design Goals

We designed the architecture of CORMORANT with the following design goals in mind:

Security and Privacy: Security is an important consideration when developing any form of user authentication. In the case of CORMORANT, this means balancing the level of security achieved with the burden that explicit authentication places on the user. The goal is to dynamically provide a risk-adequate level of security whilst minimizing user interruption.

To reduce the number of explicit authentication interactions like entering a password, we utilize behavioral and physiological biometrics, location data, and generally make use of the devices' sensors. Since those privacy sensitive data are shared in aggregated form between trusted devices using wireless communication and central servers, ensuring privacy, e.g., by using end-to-end encryption, is paramount.

Flexibility: Mobile device environments in general and the Android ecosystem in particular provide a high degree of diversity on different levels. On the hardware level, the spectrum ranges from cheap devices with outdated components to expensive, cutting edge flagship devices with ample computational resources. Available APIs depend mostly on the age of the device and the manufacturer. Form factors vary in size from smart watches to smart TVs and smart fridges. Users display a high degree of diversity in their device usage patterns with daily usage varying by more than an order of magnitude [23]. Flexibility is therefore desirable to enable the use of, for instance, biometric authentication that requires certain hardware on suitable devices without impeding the applicability of the framework on devices lacking the necessary

hardware resources. CORMORANT should also be easily customizable to account for varying needs and environments by the user.

Extensibility: With the number of available biometric authentication methods for mobile devices constantly increasing as well as the unprecedented pace with which hardware and software evolve in the mobile ecosystems, extensibility facilitates an authentication system to which independent developers and researchers can contribute novel means of authentication and risk assessment, thereby leveraging the framework infrastructure and already available means of authentication and risk assessment.

Ease of Use: Since user-friendly authentication is the topmost motivation for developing CORMORANT, usability is an important consideration when designing the framework. We consider two groups of stakeholders in this regard, namely device users and developers contributing new means of authentication or risk assessment. Regarding the device user, we strive for an user interface that is straightforward to use and has reasonable default configurations. As for developers and researchers, great importance is placed on API design, documentation and ease of implementation to make contributing to CORMORANT as easy as possible.

3.2 Android Client

We implemented CORMORANT as an Android application, though with additional effort, other platforms could be supported as well. The framework application operates on the results reported by authentication and risk plugin applications installed independently by the user as well as results reported from other trusted devices. It can lock or unlock the device or challenge the user with explicit authentication if the implicit confidence in the user's identity gained through transparent biometrics is not sufficient to grant access under the current risk level. The framework application allows the user to configure CORMORANT, including managing the group of trusted devices, learn about the location and state of other devices, and adjust various configuration parameters.

Apart from protecting general access to the device, the framework offers an interface for applications to query the current confidence in the user's identity based on available means of authentication as well as the estimated risk of unauthorized access. Moreover, applications can request to raise the confidence level in which case the framework might prompt the user with some form of explicit authentication such as pan face recognition [13]. This allows for finer access control than today's authentication models where access is largely granted on an *all-or-nothing* principle where a user either gets access to every application and service if successfully authenticated or to none otherwise. However, a calculator application arguably needs less protection than a mobile banking app. By using the CORMORANT API, an application could ensure that it only operates when certain, app-dependent confidence and risk levels are met. But one could easily go further and apply this model of access control to certain functionalities within applications or even transactions. For example, transferring 1$ using a mobile banking app might require a lower level of confidence while authorizing a transaction worth 10,000$ might require multiple authentication mechanisms to positively identify the genuine user (virtually a dynamic form of multi factor authentication).

3.3 Plugin API

To achieve the aforementioned goals of flexibility and extensibility, CORMORANT is designed as a modular system that can be extended dynamically at runtime through a plugin mechanism. Plugins come in the form of *risk plugins*, which assess the probability of unauthorized physical access to a device, and *authentication plugins* which implicitly or explicitly authenticate the user of a device. While we provide a number of biometric and knowledge-based authentication plugins as well as some risk plugins, the primary motivation behind the plugin mechanism is to allow third party developers and researchers to utilize and extend CORMORANT with novel means of risk assessment or authentication. Therefore, framework and plugins need to have independent life cycles. Plugins are hence developed and deployed as stand-alone Android applications with independent dependencies and permissions. Integration into the framework is achieved using the whiteboard pattern [41]: Plugins extend `AbstractPluginService`, which is an Android `Service`. The service implements the binding to the CORMORANT framework process as well as serializing and deserializing of parameters. Plugins are required to declare a custom Android permission to prevent applications from influencing the framework without the user's consent. Changes in risk assessment or authentication state are propagated either event-based, periodically pulled or explicitly requested upon necessity, for instance to trigger explicit authentication. The API is published as an Android library in source[1] and in the JCenter Maven repository[2]. Due to this API, creating a runnable plugin integrated into CORMORANT can be achieved with as little as implementing a single abstract method, as demonstrated in listing 1.

```
public class DemoRiskPlugin extends AbstractRiskService {
  @Override
  protected void onDataUpdateRequest() {
    publishRiskUpdate(new StatusDataRisk()
      .status(StatusDataRisk.Status.OPERATIONAL)
      .risk(0.5));
  }
}
```

Listing 1: Minimal example of a risk plugin implementation

3.4 Backend

In order to utilize authentication information gathered on other devices, trusted devices need to be able to efficiently and securely communicate. To maintain security and privacy, communication needs to be authenticated and encrypted. Groups of trusted devices also need to be dynamic, as users might add additional devices or remove devices sold, lost, or stolen at any time. Finally, the mobile and distributed application of CORMORANT requires the underlying communication to cope with lost messages and temporarily unresponsive devices.

To account for these requirements, our implementation relies on the Signal messaging protocol [40], a non-federated end-to-end encryption security protocol for instant messaging that has been recently adopted by popular instant messaging services such as WhatsApp [49], Facebook Messenger [47], and Google Allo [48].

[1] https://github.com/mobilesec/cormorant
[2] https://dl.bintray.com/mobilesec/maven/at/usmile/cormorant/cormorant-api

Signal offers a number of desirable security properties, including forward and future secrecy. The protocol has recently been formally analyzed with no major flaws found in its design [6].

To use the Signal protocol for asynchronous communication, a server is required to temporarily store and distribute messages between devices. The server also holds pre-send batches of ephemeral public keys to allow encrypting messages for devices currently offline and thus unavailable for direct exchange of ephemeral keys.

As basis for the *Cormorant* backend, we forked the Signal server[3] implemented by Open Whisper Systems under AGPLv3 open source license which serves as backend infrastructure for the Signal messaging application. The Signal codebase was modified to remove some external dependencies not necessary for *Cormorant*, for instance Apple Push Notifications (APN), Amazon Web Services storage, and Twillio's SMS service. Unlike the Signal application, we don't use phone numbers but universally unique identifiers (UUIDs) as device identifiers in *Cormorant*, so the server code was adapted to support UUIDs.

The *Cormorant* backend consists of four components. An Apache HTTP Server, configured as an reverse proxy that accept HTTPS connections coming from the Android clients and authenticates itself using a Let's Encrypt[4] X.509 TLS certificate. Requests are forwarded to the *Cormorant* Signal server which provides REST-APIs for creating accounts, distributing devices' public keys, sending messages as well as a WebSocket-API for retrieving messages. A PostgreSQL database serves as permanent storage for device accounts, public keys, and (encrypted) messages until they are delivered to the target device. A Redis in-memory data structure store is used as a shared cache in case multiple servers are required to handle the system load. The backend is deployed on an Ubuntu Linux 16.04 server hosted on Amazon Elastic Compute Cloud (EC2). The code is available under AGPLv3 open source license,[5] allowing curious or particularly privacy conscious users to deploy and host their own instance of the backend if desired.

3.5 Distance Estimation

To facilitate multi-device authentication, being able to accurately estimate the distance between trusted devices is crucial. To this end, we implemented coarse and fine distance estimation:

Coarse Distance Estimation: For coarse distance estimation, each device determines its own location using the Google Services Fused Location Provider. The Fused Location Provider utilizes GPS, WiFi and GSM data in order to determine the device's current location with an accuracy of 3 to 10 meters. The location in form of latitude and longitude coordinates is then periodically broadcasted to other devices within the same group.

After obtaining location information, each device uses *Location.distanceBetween()* from the Android Location Framework to compute the approximate relative distance in meters between the two devices.

Fine Distance Estimation: While the coarse distance estimation is fairly energy efficient, the accuracy is too coarse for some applications within the proposed framework. In addition, it is of limited availability in some indoor scenarios. We therefore utilize Bluetooth Low Energy Beacons to gain a finer distance estimation when devices are spaced only centimeters to few meters apart.

Each device publishes a Bluetooth beacon signal by repeatedly sending a static *advertisement* packet, which requires little power, and simultaneously scans for nearby beacons. A published beacon is configured with the framework specific manufacturer ID, major and minor numbers intended to identify and distinguish groups and individual beacons, and the device's UUID. While the latter is necessary to assign a discovered beacon to the corresponding trusted group member, the former is configured in order to mark *Cormorant* beacons which are filtered by the framework's beacon scanner.

Once a beacon sent by a device within the same group is discovered, the distance between the devices is computed from the received signal strength indicator (RSSI) by applying an estimation model developed by Aman et al. [2]. It calculates the estimated distance between the devices d in meters by comparing the current RSSI with the transmitting power ($txPower$) of the beacon. The transmitting power is a calibrated value, which consists of the signal level at the reference distance of one meter.

$$d = 0.89976 * (\frac{RSSI}{txPower})^{7.7095} + 0.111.$$

Security Limitation. We note that our current implementation of fine distance estimation is susceptible to unauthorized tracking by other Bluetooth receivers within range, which might compromise a user's privacy to some extent. A measure to mitigate tracking would be to apply cryptographic randomization to the broadcast identifier as proposed in [18].

Apart from tracking, Bluetooth beacons in general can be spoofed by amplifying the Bluetooth signal. In a scenario where two devices constituting a group are in different locations, an attacker could receive the beacon signal of one device, then relay it to an attacker controlled device in the vicinity of the second trusted device, which then sends out a similar signal, posing as the trusted device. To mitigate this attack, fine distance estimation is only applied when devices are believed to be in close proximity based on the coarse distance estimation. However, GPS, WiFi, and GSM signals are generally also susceptible to spoofing if a powerful attacker is equipped with the necessary hardware. If an attacker possessing the capability to spoof the distance estimation (e.g., a governmental agency) is assumed in the user's personal threat model, it is advisable to refrain from using the multi-device authentication functionality of *Cormorant* but restrain its scope to individual devices.

4 AUTHENTICATION

We implemented a number of behavioral and physiological biometric authentication plugins for *Cormorant* along with some trivial knowledge-based plugins like PIN and password as fallbacks. The following sections outline the biometric means of authentication currently available for *Cormorant*.

[3] https://github.com/WhisperSystems/Signal-Server
[4] https://letsencrypt.org
[5] https://github.com/mobilesec/cormorant-signal-server

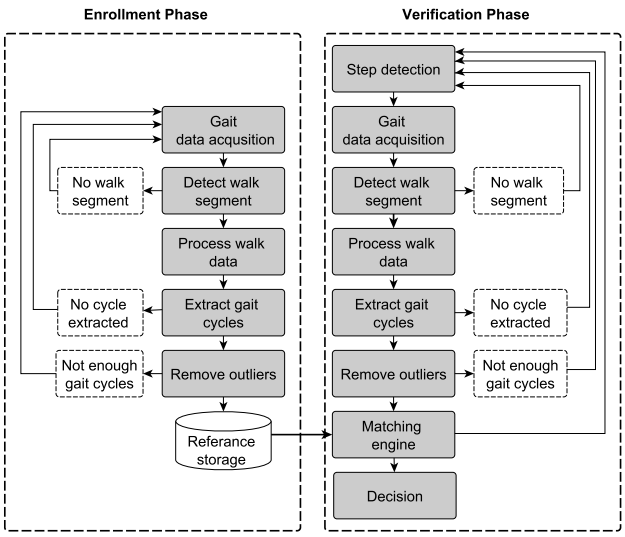

Figure 1: Gait recognition overview

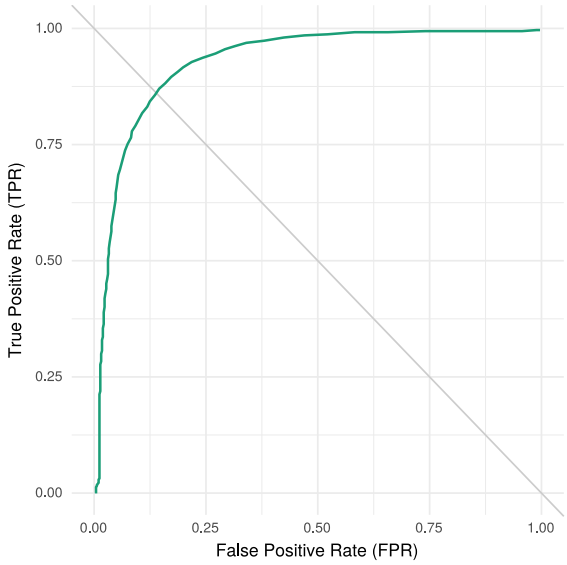

Figure 2: Gait recognition evaluation results

4.1 Gait Recognition

Gait as a biometric trait offers an unobtrusive way of recognizing individuals from their walking styles. Various studies have explored the feasibility of deploying gait authentication on off-the-shelf mobile devices [9, 32, 37, 39]. We have developed a gait plugin that implicitly processes accelerometer data and delivers authentication results to the *CORMORANT* framework. Figure 1 outlines the steps involved in the enrollment and the verification phases. To enroll, a gait template is created by walking ≈ 300 meters at normal pace, carrying the mobile phone in the trousers' front pocket. Once the gait template is generated, the plugin automatically switches to continuous verification which is similar to the enrollment phase. Details about the gait template generation process can be found in [37, 38]. Capturing acceleration data is fairly power-intensive. We therefore utilize a low-powered ever-on step detector sensor to avoid recording accelerometer data when the user is not actually walking. Once the user starts walking, the step detector triggers the plugin which in turn registers to the accelerometer sensor to start recording acceleration values. Accelerometer data are recorded for a period of fifteen seconds, after which the application checks if the user is still walking by monitoring the timestamps between the steps taken by the user, and if so, continues to record acceleration data. In parallel, previously recorded data are processed and authentication results computed by applying a matching engine that uses Dynamic Time Warping (DTW) distance to compare live gait cycles with the enrolled template.

To evaluate the performance of our gait authentication plugin, we have recorded biometric gait data from 35 participants (6 females and 29 males) using a Google Nexus Android phone. For data collection purpose, we developed an Android application which records three dimensional (X, Y, and Z axis) accelerometer data at a sampling rate of 100 Hz and writes it to a text file with timestamps. Participants were asked to wear a trousers with not-too-loose front pockets. For capturing a distinctive walking style, the phone or sensor must be placed close to the body otherwise it might pick up too much random noise. In the data recording phase, the phone was placed inside the trousers' right side pocket. Participants were asked to walk at their normal pace in a 68 meter long straight corridor (with no stairs). They were told to wait for one second at the end of the walk then turn around and wait for another second before starting their new walk. In one session, every subject walked $4 \times 68 = 272$ meters or in other words completed two rounds of the corridor. For every subject, data recording was conducted in two different sessions. An average gap between the sessions was about 25 days. Eight walks were recorded for every subject in two different sessions. The first walk was used to generate the reference template of the subjects, whereas, the remaining walks were used to generate probe templates. Once the reference and probe templates are generated they are compared against each other by using DTW to compute inter-class and intra-class distances. Using this data set we have obtained a 13% Equal Error Rate (EER) as shown in fig. 2.

4.2 Keystroke Dynamics Recognition

Every person has their own individual typing style while entering data into a computer system. By employing these so called keystroke dynamics, it is possible to recognize the user of a device. Keystroke dynamics is an implicit and unobtrusive biometric feature that can be captured without any user interaction apart from usual keyboard input. An important design decision is whether keystroke recognition should be static or dynamic. For static recognition, the system is aware of the text to be entered beforehand and can train the classifier for specific input sequences. Dynamic recognition, however, is much more difficult as any interaction of a user with the keyboard is taken into account and the system has to continuously re-evaluate whether that matches the user's general behavior. The increased difficulty is offset by the possibility to authenticate users more often. The main component of keystroke dynamics recognition is measuring the flow of key presses and extracting timing

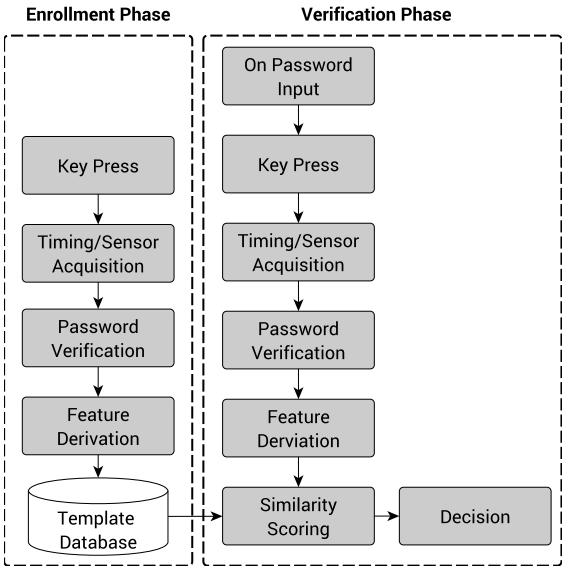

Figure 3: Keystroke dynamics authentication overview

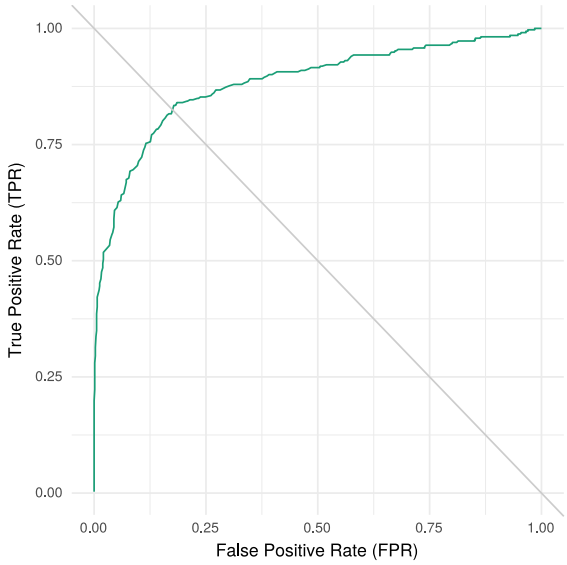

Figure 4: Keystroke dynamics authentication results

information. Commonly used metrics are digraphs (press-to-press time between one and the consecutive key) and dwell time (time a key is held down). These timings can be gathered using any type of keyboard, but mobile devices contain a vast array of sensors available for enhancing performance. In addition, a touch screen tells much more about a user's interaction with the keyboard than other devices. This can be the pressure applied to the screen for each keystroke, angle and size of the finger on the screen, or the relative position of the finger to the pressed key.

The *Cormorant* keystroke dynamics plugin is a static recognition system for password inputs. These usually stay the same for a long time, have relatively few characters and are therefore quicker to verify than whole texts. Also, they distinguish different typists more easily as the genuine user has more training. We extended the standard keyboard contained in the Android AOSP with functionality to capture keystroke behavior and an API to indicate successful logins. Classification is performed using a Greedy Maximum Match Scores (GMMS) measure with modification to minimize instead of maximizing [35]. The keystroke timings are augmented by the aforementioned touch metrics, gravity and accelerometer sensor data. These are captured every time a keystroke event (pressing or releasing a key) occurs.

For evaluation, we developed an app using a randomly generated six-character password with one number at start and end as login. We asked three male owners of identical Google Nexus 4 phones and a male owning a Samsung Galaxy S4 to enter this password over three months and a total of 100 times each. They were asked to distribute acquisitions over various situations during the day. This resulted in 300 samples for training and evaluating the algorithm and 100 samples for testing whether the trained model is portable between devices. For evaluation, the enrollment phase ended after the first ten acquisitions, the rest were used for verification. Picking the optimum data sources as explained above, we could obtain a 17.19% EER as shown in fig. 4.

4.3 Speaker Recognition

Speaker recognition is a technique that allows to identify individuals from their voice samples. Speaker recognition systems can be divided into two types: *text-dependent* and *text-independent* [3]. In *text-dependent* speaker recognition systems users use the same utterance for enrollment and verification phase. Whereas, in a *text-independent* system, users are not bound to use the same utterance for enrollment and verification process. Moreover, a speaker recognition system on smartphones can be used in explicit and implicit fashion. Considering the use cases and types of speaker recognition systems, we have designed a speaker verification plugin that uses the Gaussian Mixture Model-Universal Background Model (GMM-UBM) verification framework, and it can be used in both implicit and explicit scenarios. Briefly, a GMM-UBM approach consists of three main steps. Firstly, a UBM is trained offline by using Expectation Maximization (EM), and the training data for UBM is obtained by pooling the feature vectors extracted from the speech data of lots of subjects. In general, a UBM is intended to represent a subject independent distribution of acoustic features. Secondly, user specific models are adapted from the UBM by using the Maximum-A-Posteriori (MAP) adaptation, and finally the feature vectors extracted from the test data are evaluated against the UBM and user specific model and a likelihood ratio test is carried out to accept or reject genuine and impostor users. Details about the GMM-UBM approach can be found in [42]. Figure 5 shows various steps of the enrollment and the verification process. In the enrollment phase, first, users are required to download pre-trained UBM models from a server as per their use case. This is due to the reason that training a UBM is a computation intensive task, and it will also be cumbersome for users to collect data from a lot of subjects. Thereafter, users are asked to record speech samples which are used to adapt speaker specific models. These speaker specific models are stored in the database. In the verification phase,

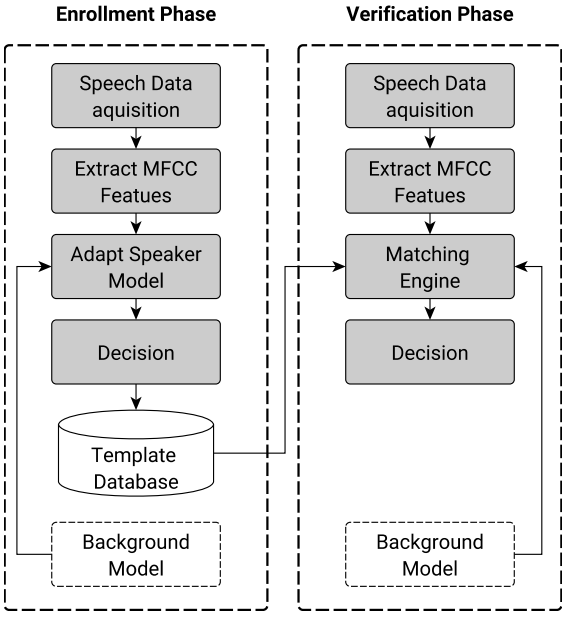

Figure 5: Voice authentication overview

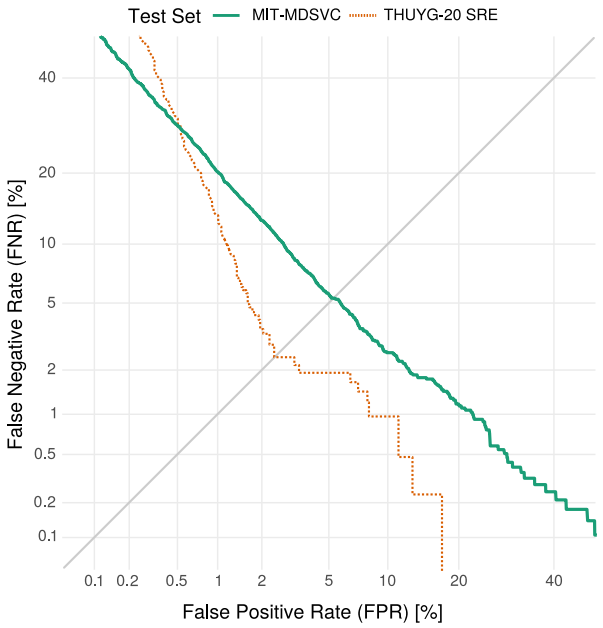

Figure 6: Voice authentication evaluation results

speech data collected (implicitly or explicitly) from the user undergoes feature extraction. These features are evaluated against the UBM and speaker specific models, and authentication results are delivered back to the *CORMORANT* framework.

We have used THUYG-20 SRE and MIT-MDSVC data sets to evaluate the performance of our text-independent and text-dependent speaker recognition plugin, respectively. THUYG-20 SRE is a text-independent data set, consisting of more than 20 hours of speech data recorded from 353 speakers by using a carbon microphone at a sampling rate of 16 KHz. The entire data set consists of three parts: a ubm-set, an enroll-set, and a test-set. The ubm-set consists of 4771 utterances recorded from 100 male and 100 female speakers. The enroll-set consists of 153 utterances recorded from 153 (87 female and 66 male) speakers. Each enrollment utterance is 30 seconds long. The test-set also consists of 153 speakers including 87 female and 66 male. Each utterance in the test-set is about ten seconds long. After closely observing the data set, we found some participants do not have any test utterance, and others have a different number of test utterances. Furthermore, we found that 141 speakers have at least three test utterances. Therefore, we did not include those speakers who have less than three test utterances. Moreover, for users who have more than three test utterances, we have only included their first three test utterances to ensure that each speaker has an equal number of test utterances. From every frame of each utterance in the THUYG-20 SRE data set, we extracted a 39 dimensional feature vector including 13 Mel Frequency Cepstral Coefficients (MFCC), 13 delta MFCC, and 13 double delta MFCC. Afterwards, the Cepstral Mean Variance Normalization (CMVN) was applied at utterance level to normalize the feature vectors. Features extracted from the ubm-set were pooled together to train a speaker independent UBM with 512 Gaussian components. Features extracted from the enrollment-set were used to adapt speaker specific models, and features extracted from the test-set were used to evaluate the performance of the plugin. We have obtained an EER of 2.3641% (Fig. 6).

The MIT-MDSVC data set is a text-dependent data set, and it was recorded in variable acoustic conditions at a sampling rate of 16 KHz by using a hand-held device. These acoustic conditions involve three different locations (a quiet office, a noisy hallway, and a busy intersection) and two different microphones (the internal microphone of the hand-held device and the microphone of an external headset). The combinations of microphones and locations resulted in six different acoustic conditions. In each acoustic condition, subjects were asked to recite nine short phrases such as "mint chocolate chip" which were displayed on the device. Thus, for all six acoustic conditions, a total of 54 speech samples were recorded for each subject. The speech data were recorded from 88 subjects including 45 female and 43 male participants. From each frame of every utterance of the data set, we extracted 39 dimensional feature vectors. The speech data from 40 (23 female and 17 male) subjects was recorded in a single session and it was used to train the UBM with 256 Gaussian components, and the data of the remaining 48 subjects (22 female and 26 male) were recorded twice in two different sessions, where each session lasted about 20 minutes. The data from the first session were used to adapt speaker specific models and the data from the second session were used to evaluate the plugin. The EER obtained using this setup is 5.3498% (Fig. 6).

4.4 Face Recognition

Face authentication verifies users based on their facial features. With modern mobile devices, embedded cameras serve as the source for face images containing those features. Mobile face authentication is possible both in explicit and implicit form. Explicit means users deliberately positioning device camera and face to each other,

while implicit means that cameras capture facial features without users deliberately cooperating, e.g., during conventional device usage.

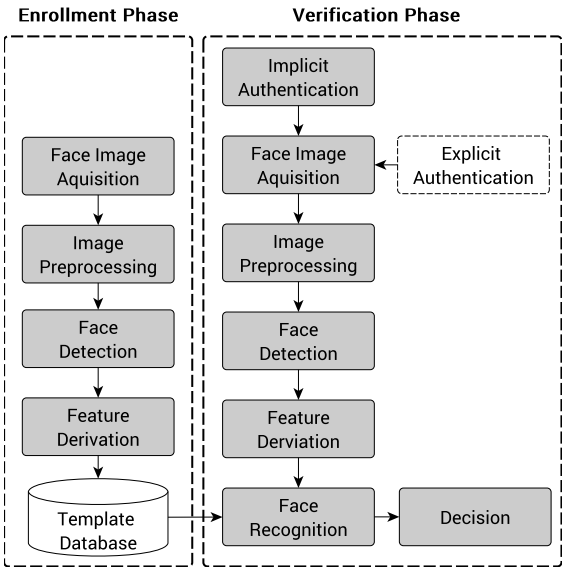

Figure 7: Face authentication overview

With the *Cormorant* face authentication plugin [6] enrollment is explicit, while verification can be both explicit and implicit. For enrollment users need to deliberately take a predefined number of images of their faces in different settings, e.g., with variation in face illumination. For explicit face verification with our plugin, users deliberately position the device's camera and their face to each other to provide an authentication confidence measurement to *Cormorant*. In contrast, with implicit authentication our plugin continuously utilizes faces visible to the device camera and provides the corresponding authentication confidence measurements to *Cormorant*. The latter can typically be done while users interact with mobile devices and the embedded front facing cameras capture their faces. In both settings, the faces captured during authentication are compared to enrollment faces to deduce an authentication acceptance or rejection.

In terms of processing, the plugin at first applies grayscaling, downscaling, and histogram equalization to an original image. It then performs Viola and Jones face detection [34, 55] and segments the quadratic region around the largest detected face, should its diagonal be at least $\frac{1}{4}$ of the diagonal of the image. Segmented faces are downscaled and their histogram is equalized again. Then features are extracted as coefficients of a multiresolution analysis 2D discrete wavelet transform using the Daubechies Least-Asymmetric 2D wavelet [8]. During enrollment the features of each captured face are stored in the enrollment template. During authentication those are compared pairwise with features from newly captured faces, using the pairwise absolute distance between feature vectors with a precomputed Linear Discriminant Analysis (LDA) model (shipped with the face authentication plugin) to deduce the final authentication acceptance or rejection [12] as depicted in fig. 7.

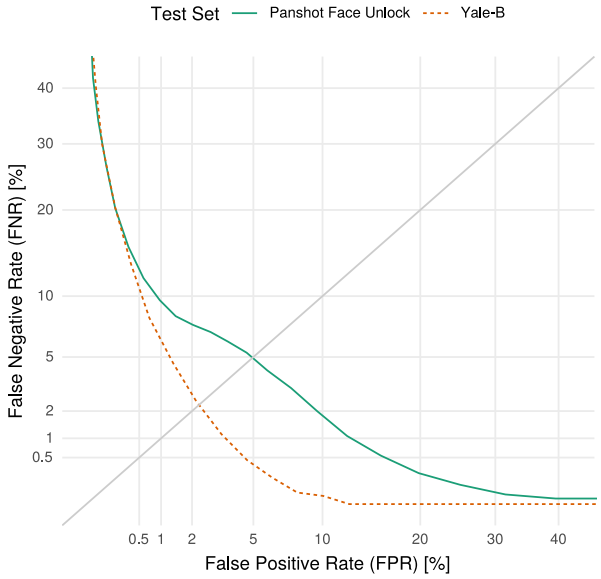

Figure 8: Face authentication evaluation results

The face authentication approach underlying the *Cormorant* face authentication plugin has been evaluated on subsets of the Yale-B [33] and the Panshot Face Unlock database [11], containing 511 pictures of 27 participants and 600 pictures of 30 participants. The evaluation procedure utilized a 50/50 gallery insensitive training and test set split. Evaluation of different setups and selection of the optimal choice thereof only utilized data from the training set, internally using a ten-fold cross validation as resampling technique. The selected setup from training data uses eight face images in the enrollment template and four newly recorded face images for authentication, which allows for a total computation time of about one second, once faces have been captured. Keeping this duration short is important, as results should be available as quickly as possible when performing face authentication explicitly. This setup further achieved an EER of 2.4% and 5.3% for the Yale-B and the Panshot Face Unlock database on the previously held-back test set [12] (fig. 8).

5 LESSONS LEARNED

In this section we share our perspective on some of the successes and shortcomings we encountered implementing and evaluating *Cormorant* in a hope to inform others working on similar projects.

One design decision that in particular proofed very valuable was to introduce a plugin mechanism that decoupled different implementations of risk evaluation and authentication from the main framework and from one another. Since essentially each of these plugins was developed by a different researcher across a number of research groups and institutes, it maximized autonomy and flexibility that only a minimal API needed to be implemented. Dependencies, architecture, permissions, configurations, enrollment and alike were handled in the individual plugin projects which lowered the

[6] The *CORMORANT* face authentication plugin source code is public at https://github.com/mobilesec/authentication-framework-plugin-face.

entry barrier for onboarding new plugins significantly. Since plugins are run as standalone applications as far as the Android runtime is concerned, crashes also only affected an individual plugin and not the entire system. However, we also noted a number of downsides to this approach. On the framework part, it significantly increases the complexity since it needs to handle intra process communication as well as the unpredictable appearance and disappearance of individual plugin processes. Another consequence was that it became much harder to create an appealing user experience. For instance, installing the framework without a minimal set of risk and authentication plugins limits its utility substantially. Also the look and feel of each plugin is naturally different from the core framework and the rest, making it a rather patchy experience. We still believe that in the context of a research project, the benefits of the plugin approach outweigh its drawbacks.

A critical capability when developing a novel authentication mechanism for mobile devices is the ability to reliably lock and unlock the device. On modern mobile operating systems, applications are sandboxed and confined to prevent a rogue app from taking control over the device [36], thus a dedicated API for controlling the lockscreen is necessary. When we started this project in 2014, such an API still existed in form of the `KeyguardManager` which allowed locking – and more importantly – unlocking an Android device programmatically. However, `KeyguardManager` was deprecated with API level 15 and no longer works on modern devices. Another still existing API is the `DevicePolicyManager`, mainly intended to allow companies to remotely manage their mobile device fleet. It facilitates locking a device programatically, yet does not support unlocking it. Still an API perfect for projects like *Cormorant* exists in form of the `TrustAgentService`, a "service that notifies the system about whether it believes the environment of the device to be trusted"[7], which enables Android's smart lock features like face unlock or unlock by Bluetooth devices in range. Unfortunately, the API is marked as *@SystemApi* and thus hidden from the public API. To circumvent the visibility restriction, it is necessary to "root" the device, thereby obtaining more or less full control over the operating system. However, this would mean violating Android's security model [36] which might be acceptable for a research project but is ill advised to apply on actual consumer devices. So after `KeyguardManager` ceased working on current devices, we unfortunately had to abandon ambition to integrate directly with the native Android lock screen and had to resort – like other authentication research projects (e.g. [45]) – to a purely visual indicator to signify the lock state. Since this does not offer any protection against unauthorized access, third-party authentication systems like *Cormorant* can not practically be used to unlock devices aside from demonstrating the concept unless they are either adopted by the Android Open Source Project or a public API similar to *TrustAgentService* is introduced. This also limits how such systems can be evaluated without exposing study participants to the risk of unauthorized device access.

This leads us to the most challenging problem we encountered: The question of how to evaluate a complex and dynamic authentication system like *Cormorant*. Fundamentally, the two conflicting objectives *usability* and *security* are of most interest. For subcomponents like individual biometrics, we could use established techniques like cross-validation to measure their performance in both regards, quantified e.g., by their Equal Error Rate. For assessing the overall usability of the system, handset-based user studies can be conducted. Usually, usability is then quantified by comparing the number of explicit authentication interactions against either a control group or a baseline established transparently in the background [57]. Measuring security e.g., by evaluating unauthorized access is practically impossible [22] to do in such a study, as theft, robbery, and device loss are luckily rather infrequent on the level of an individual – besides being also hard to reliably detect and quantify. This is problematic, as measuring only usability without accounting for security at the same time can arguably be misleading, given that ultimately authentication could be disabled entirely to maximize usability. Lab studies are sometimes used to overcome this limitation by simulating unauthorized access, but they don't scale beyond a controlled environment and a limited number of participants (e.g., $n = 9$ [43]) and are thus limited in their generalizability. We therefore chose to refrain from evaluating the performance of *Cormorant* in that way and refer to the extensive simulation-based evaluation of the underlying concept of *Cormorant* as reported in [22], which relies on real world experimental data gathered during the implementation of the individual biometrics in this paper and therefore as close to the implementation presented here as possible. Since the problem of how to evaluate the overall system generally applies to similar projects as well, one potential solution could be for the research community to converge on a common simulation model that could be applied to assess individual implementations, similar to how standardized datasets allow to compare the performance of different implementations of biometric systems.

6 CONCLUSION AND FUTURE WORK

In this paper we presented the lessons learned from designing and implementing *Cormorant*, an open source Android authentication framework that facilitates to reduce the overhead of manual authentication by up to 97% [22] whilst improving security at the same time. While challenging, we found no insurmountable obstacles with regards to implementing various biometrics, risk estimations or the framework itself. However, given the for security reasons restrictive Android API and runtime environment, we note that support from the operating system vendor is generally necessary for *Cormorant* and similar projects to be of practical applicability. We also found that evaluating such complex authentication systems in form of user studies is infeasible as security can't be adequately measured and consider standardized large-scale simulations a viable alternative. Finally we invite interested researches to make use of our open source contributions presented in this paper.[8]

REFERENCES

[1] Yusuf Albayram, et al. 2017. "...better to use a lock screen than to worry about saving a few seconds of time": Effect of Fear Appeal in the Context of Smartphone Locking Behavior. In *SOUPS*. 49–63.
[2] Sayedul Aman, et al. 2016. Reliability Evaluation of iBeacon for Micro- Localization. In *Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. 1–5.

[7] https://android.googlesource.com/platform/frameworks/base/+/d4efaac/core/java/android/service/trust/TrustAgentService.java

[8] https://github.com/mobilesec/cormorant

[3] Frédéric Bimbot, et al. 2004. A Tutorial on Text-independent Speaker Verification. *EURASIP J. Appl. Signal Process.* 2004 (Jan. 2004), 430–451.

[4] Jagmohan Chauhan, et al. 2018. Performance Characterization of Deep Learning Models for Breathing-based Authentication on Resource-Constrained Devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 158 (Dec. 2018), 24 pages.

[5] Richard Chow, Philippe J. P. Golle, and Jessica N. Staddon. 2012. Adjusting security level of mobile device based on presence or absence of other mobile devices nearby.

[6] Katriel Cohn-Gordon, et al. 2017. A Formal Security Analysis of the Signal Messaging Protocol. *Proceedings - 2nd IEEE European Symposium on Security and Privacy, EuroS and P 2017* November (2017), 451–466.

[7] Heather Crawford, Karen Renaud, and Tim Storer. 2013. A framework for continuous, transparent mobile device authentication. *Computers and Security* 39, PART B (2013), 127–136.

[8] Ingrid Daubechies. 1993. Orthonormal bases of compactly supported wavelets II. Variations on a theme. *SIAM Journal on Mathematical Analysis* 24, 2 (1993), 499–519.

[9] Mohammad Omar Derawi. 2012. *Smartphones and Biometrics: Gait and Activity Recognition.* Ph.D. Dissertation. Gjøvik University College.

[10] Serge Egelman, et al. 2014. Are You Ready to Lock? *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), 750–761.

[11] Rainhard D. Findling. 2013. *Pan Shot Face Unlock: Towards Unlocking Personal Mobile Devices using Stereo Vision and Biometric Face Information from multiple Perspectives.* Master's thesis. University of Applied Sciences Upper Austria, Hagenberg, Austria.

[12] Rainhard D. Findling, Michael Hölzl, and René Mayrhofer. 2018. Mobile Match-on-Card Authentication Using Offline-Simplified Models with Gait and Face Biometrics. *IEEE Transactions on Mobile Computing* 17, 11 (Nov 2018), 2578–2590.

[13] Rainhard D. Findling and René Mayrhofer. 2013. Towards Pan Shot Face Unlock: Using Biometric Face Information from Different Perspectives to Unlock Mobile Devices. *International Journal of Pervasive Computing and Communications* (2013), 190–208.

[14] Dawud Gordon, John Tanios, and Oleksii Levkovskyi. 2019. Deep Learning for Behavior-Based, Invisible Multi-Factor Authentication. https://patents.justia.com/patent/20190044942

[15] Nazirah Abd Hamid, et al. 2011. Mouse movement behavioral biometric systems. In *2011 International Conference on User Science and Engineering (i-USEr).* 206–211.

[16] Marian Harbach, et al. 2016. Keep on Lockin' in the Free World: A Multi-National Comparison of Smartphone Locking. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16* (2016), 4823–4827.

[17] Marian Harbach, et al. 2014. It's a Hard Lock Life: A Field Study of Smartphone (Un) Locking Behavior and Risk Perception. *Symposium on Usable Privacy and Security (SOUPS)* (2014), 213–230.

[18] Avinatan Hassidim, et al. 2016. Ephemeral Identifiers : Mitigating Tracking & Spoofing Threats to BLE Beacons. (2016), 1–11.

[19] Eiji Hayashi, et al. 2013. CASA: context-aware scalable authentication. In *Symposium on Usable Privacy and Security (SOUPS).*

[20] Daniel Hintze. 2015. Towards transparent multi-device-authentication. In *UbiComp/ISWC'15 Adjunct.* ACM, 435–440.

[21] Daniel Hintze, et al. 2015. CORMORANT: Towards Continuous Risk-Aware Multi-Modal Cross-Device Authentication. *UbiComp 2015 Adjunct Publication* (2015).

[22] Daniel Hintze, et al. 2019. CORMORANT: Ubiquitous Risk-Aware Multi-Modal Biometric Authentication Across Mobile Devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 85 (2019). Issue 3.

[23] Daniel Hintze, et al. 2017. A Large-Scale, Long-Term Analysis of Mobile Device Usage Characteristics. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 1–21.

[24] Daniel Hintze, et al. 2015. Confidence and Risk Estimation Plugins for Multi-Modal Authentication on Mobile Devices using CORMORANT. In *Proceedings of MoMM 2015.* 384–388.

[25] Daniel Hintze, et al. 2016. Location-based Risk Assessment for Mobile Authentication. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct.* http://dx.doi.org/10.1145/2968219.2971448

[26] Christopher G. Hocking, et al. 2011. Authentication Aura - A distributed approach to user authentication. *Information Assurance and Security* 6, 2 (2011).

[27] Adam Hurkala and Jaroslaw Hurkala. 2014. Architecture of Context-Risk-Aware Authentication System for Web Environments. *ICIEIS'2014* (2014), 219–228.

[28] Markus Jakobsson, et al. 2009. Implicit authentication for mobile devices. *HotSec'09* (2009).

[29] Philipp Kapfer. 2016. *PhonyKeyboard: Sensor-enhanced Keystroke Dynamics Authentication on Mobile Devices.* Master Thesis. Johannes Kepler University Linz.

[30] Hassan Khan, Aaron Atwater, and Urs Hengartner. 2014. Itus : An Implicit Authentication Framework for Android. *In Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), 507–518.

[31] Dong Ju Kim, Kwang Woo Chung, and Kwang Seok Hong. 2010. Person Authentication using Face, Teeth and Voice Modalities for Mobile Device Security. *IEEE Transactions on Consumer Electronics* 56, 4 (2010), 2678–2685.

[32] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. 2010. Cell phone-based biometric identification. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on.* IEEE, 1–7.

[33] Kuang-Chih Lee, J. Ho, and D. J. Kriegman. 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 5 (May 2005), 684–698.

[34] Rainer Lienhart and Jochen Maydt. 2002. An Extended Set of Haar-Like Features for Rapid Object Detection. In *IEEE International Conference on Image Processing 2002.* 900–903.

[35] Emanuele Maiorana, et al. 2011. Keystroke dynamics authentication for mobile phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing - SAC '11.* ACM Press, New York, New York, USA, 21.

[36] René Mayrhofer, et al. 2019. The Android Platform Security Model. *CoRR* abs/1904.05572 (2019). arXiv:1904.05572 http://arxiv.org/abs/1904.05572

[37] Muhammad Muaaz and Rene Mayrhofer. 2014. Orientation Independent Cell Phone Based Gait Authentication. *Proceedings of MoMM 2014* (2014).

[38] M. Muaaz and R. Mayrhofer. 2017. Smartphone-Based Gait Recognition: From Authentication to Imitation. *IEEE Transactions on Mobile Computing* 16, 11 (Nov 2017), 3209–3221.

[39] Claudia Nickel. 2012. *Accelerometer-based Biometric Gait Recognition for Authentication on Smartphones.* Ph.D. Dissertation. TU Darmstadt.

[40] Open Whisper Systems. 2018. Signal Specification. https://signal.org/docs/

[41] OSGi Alliance. 2004. Listeners Considered Harmful: The Whiteboard Pattern. (2004), 16 pages.

[42] Douglas A. Reynolds, Thomas F. Quatieri, and Robert B. Dunn. 2000. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing* 10, 1 (2000), 19 – 41.

[43] Oriana Riva, et al. 2011. Progressive Authentication: Deciding When to Authenticate on Mobile Phones. *Proceedings of the 21st USENIX Security Symposium* (2011), 1–16.

[44] Arun Ross and Anil K Jain. 2004. Multimodal Biometrics: an Overview. *Signal Processing* September (2004), 1221–1224.

[45] Stefan Schneegass, et al. 2014. SmudgeSafe: Geometric Image Transformations for Smudge-resistant User Authentication. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14).* ACM, New York, NY, USA, 775–786.

[46] S. Shekhar, et al. 2014. Joint Sparse Representation for Robust Multimodal Biometrics Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 1 (Jan 2014), 113–126.

[47] Signal Messenger. 2016. Facebook Messenger deploys Signal Protocol for end-to-end encryption. (2016). https://signal.org/blog/facebook-messenger

[48] Signal Messenger. 2016. Open Whisper Systems partners with Google on end-to-end encryption for Allo. (2016). https://signal.org/blog/allo/

[49] Signal Messenger. 2016. WhatsApp's Signal Protocol integration is now complete. (2016). https://signal.org/blog/whatsapp-complete

[50] Hiew Moi Sim, et al. 2014. Multimodal biometrics: Weighted score level fusion based on non-ideal iris and face images. *Expert Systems with Applications* 41, 11 (2014), 5390–5404.

[51] Frank Stajano. 2011. Pico: No more passwords! *Lecture Notes in Computer Science* 7114 LNCS (2011), 49–81.

[52] Jiayao Tan, et al. 2018. SilentKey: A New Authentication Framework through Ultrasonic-based Lip Reading. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2 (2018), 1–18.

[53] P. Tresadern, et al. 2013. Mobile Biometrics: Combined Face and Voice Verification for a Mobile Platform. *IEEE Pervasive Computing* 12, 01 (2013), 79–87.

[54] Alex Varshavsky, et al. 2007. Amigo: Proximity-Based Authentication of Mobile Devices. In *UbiComp 2007: Ubiquitous Computing.* Berlin, Heidelberg, 253–270.

[55] P. Viola and M. Jones. 2001. Rapid object detection using a boosted cascade of simple features. *Proceedings of these 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (2001), 511–518.

[56] Lei Wang, et al. 2018. Unlock with Your Heart: Heartbeat-based Authentication on Commercial Mobile Phones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 140 (Sept. 2018), 22 pages.

[57] Emanuel Von Zezschwitz, Paul Dunphy, and Alexander De Luca. 2013. Patterns in the Wild: A field study of the usability of pattern and pin-based authentication on Mobile Devices. *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2013), 261–270.