2018

BIOKEY

FINAL REPORT

SE4450 – SOFTWARE ENGINEERING DESIGN II

INSTRUCTOR: DR. LUIZ CAPRETZ
FACULTY SUPERVISOR: DR. JAGATH SAMARABANDU
SUBMISSION DATE: APRIL 11 2018

CONNOR GILES	250722866	CGILES6@UWO.CA
BRANDON KUCERA	250740274	MKUCERA3@UWO.CA
SANKALP HARIHARAN	250737042	SHARIHA2@UWO.CA
JOSHUA WEINSTEIN	250725229	JWEINST3@UWO.CA
TONY WU	250736257	TWU244@UWO.CA

Abstract

In a world of increasing security vulnerability and high-profile data breaches, there is clearly a need for improved enterprise security applications. While traditional security frameworks tend to be limited to simply static authentication of users when they log in, BioKey seeks to expand on this by moving towards continuous authentication (CA). Our CA is implemented via the biometric of keystroke dynamics – specifically by continuously verifying the user based on the pattern of the keys they type, how long they hold each down for, and how quickly they transition between certain sets of keys. Through a complex machine learning algorithm which generates a profile for each user based on their previous keystrokes, a prediction can be made for the likelihood that the current user of a computer is the user that is assigned to it. In the event that the user is an imposter, BioKey will fully lock them out of the computer until the user can be re-authenticated. Re-authentication can be established with a code sent via text message, or alternatively, given potential business needs where the computer may not have internet access (i.e. on an airplane), users can re-authenticate via time-based, one-time passwords implemented with Google Authenticator. BioKey adds an additional level of security to enterprises by expanding the scope of authentication to include all uses of the computer, as opposed to just login.

Acknowledgements

Our team would like to thank Dr. Capretz for organizing this year's capstone project. We would also like to thank our advisor, Dr. Samarabandu, for his help throughout the project and specific advice on our machine learning algorithm. Further, we are very grateful to our fellow students who provided us critical feedback which helped develop our ideas and put us on a trajectory of success.

Table of Contents

Table of Figures

1. Introduction to User Authentication

1.1 Increasing Need for Enterprise Cybersecurity

Cybersecurity is defined as the practice of defending computers, servers, mobile devices, networks and all other electronic systems from cyberattacks.[1] Cyberattacks fall into two broad categories: data security and sabotage. The former includes attempting to steal users' personal data, intellectual property, trade secrets or any other information that an enterprise chooses to not publicly disclose. Sabotage means purposely trying to break or crash a website using a variety of techniques such as a denial-of-service attack.[2] With so many high-profile breaches happening in this decade, it is no wonder that firms in all industries are taking notice and improving their information security systems. It is predicted that the world-wide cybersecurity market will increase from $75 billion in 2015 to $150 billion in 2020.[3] Despite this massive increase in expenditure and awareness, cybercrimes cost the global economy over $400 billion annually.[4]

1.2 Improvements in Static Authentication

As enterprise security becomes more consequential, the need to adopt techniques which preclude or even stymie those with nefarious purposes has increased. We have entered an era where traditional passwords are considered inadequate and have been replaced by more advanced forms of authentication. For example, some companies will send a new password daily to their employees' mobile devices. Other businesses are developing other techniques such as facial and voice recognition or fingerprint scanning. Another approach, spearheaded by Google, is a USB device called a Security Key which performs the verification.[5] One common characteristic amongst these techniques is that they are static security control systems and determination processes. Once the user has passed the initial authentication, there is significantly less security monitoring the user. However, these systems also require specialized hardware to implement, which deters companies from investing in them. As a result, weak static authentication leaves many firms vulnerable to data breaches, as exemplified by Equifax earlier this year.[6]

In today's environment of widespread cyber-intrusions, enterprises and governments need a well-designed and seamlessly integrated solution that transforms the current static practices into dynamic ones that provide continuous authentication.[7]

1.3 Continuous Authentication

Continuous authentication looks to increase systems security using a range of methods to verify the identity of the user throughout the session. Academic research has been performed on many biometric methods such as walking gait, touch gestures, multi-modal, input patterns, location familiarity, and even device power consumption.[8] Non-biometric methods have also been considered, including the proximity of a device to the system and the detection of anomalies within the use of the system or network traffic.[9] [10]

---

[1] https://usa.kaspersky.com/resource-center/definitions/what-is-cyber-security
[2] https://www.whitecase.com/publications/insight/cyber-risk-why-cyber-security-important
[3] http://blogs.wsj.com/venturecapital/2016/02/17/the-daily-startup-increased-spending-in-cybersecurity-drives-funding-surge/
[4] https://www.reuters.com/article/us-cybersecurity-mcafee-csis/cyber-crime-costs-global-economy-445-billion-a-year-report-idUSKBN0EK0SV20140609
[5] https://www.pwc.com/sg/en/publications/assets/pwc-global-state-of-information-security-survey-2016.pdf - page 10 & 11
[6] https://krebsonsecurity.com/2017/05/fraudsters-exploited-lax-security-at-equifaxs-talx-payroll-division/
[7] https://www.cio.gov/fed-it-topics/cybersecurity/continuous-monitoring/
[8] http://delivery.acm.org/10.1145/2990000/2984403/p72-stylios.pdf?ip=129.100.253.83&id=2984403&acc=ACTIVE%20SERVICE&key=FD0067F557510FFB%2E26B1F5E6B598D80D%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=814194254&CFTOKEN=20072440&__acm__=1506558735_59aaa469736ec1c04d866ecab54c9b36
[9] http://www.sciencedirect.com/science/article/pii/S0167404813000874
[10] http://ieeexplore.ieee.org/document/6966361/

For example, BehavioSec uses multiple behavioural biometrics to implement continuous authentication as an SDK for web and mobile applications rather than on the OS level.[11] They have built models to continuously authenticate the identity of the user for 3rd party applications. This multi-featured strategy, or multimodal strategy, is a fusion of multiple biometric indicators.[12] Another example is Entrust Datacard which is preparing to offer continuous authentication into their authentication strategy and have already used proprietary strategies to assure the validity of application requests.[13]

Overall, continuous authentication and behavioural biometrics will continue to grow as companies look to increase security without compromising user experience and ease of use. Protecting employees' devices with reliable OS level, continuous authentication methods is an essential step in improving information security. Further, this level of security allows business and IT personnel to act pre-emptively to possible security threats instead of merely reacting to them.

1.3.1 Keystrokes as a Form of Continuous Authentication

Research surrounding keystroke dynamics - the pattern and timing of a user's typing - as a distinguishing biometric has been ongoing since the late 19th century when telegraph operators could identify the source of messages based on rhythm.[14] With the invention of the computer, research surrounding keystroke dynamics has grown into how it could be used for continuous authentication. The authentication problem can be divided into static and dynamic keystroke authentication.[15] Static keystroke authentication applies to validating the similarity of a fixed typed phrase versus historical entries. Dynamic keystroke authentication is a larger problem in which authentication is performed against the user's overall typing style, allowing free-text entry.

A Survey of Keystroke Dynamics Biometrics aggregated results from 187 publications in this space, 11 of which focused on dynamic keystroke authentication.[16] The results are outlined in Exhibit 1 and 2 in the Appendix and show that static authentication is a reliably unique biometric and that there are promising improvements in dynamic authentication. Much of the academic research thus far has focused on statistical models using Gaussian probabilities,[17] decision trees,[18] clustering methods.[19] Most of the recent research has attempted to implement newer machine learning methods such a long short term memory (LSTM) neural networks.[20] [21] Models for dynamic authentication have been built with false acceptance rates (FAR - how often an imposter is accepted) as low as 0.14% and false rejection rates (FRR - how often the correct user is rejected) as low as 0% in academic settings (note: these results were not achieved in the same study). These results demonstrate the feasibility of free-text input for authentication. Many of these models were hindered by their relatively small data sets to train from. The small datasets can be attributed to the need for a controlled academic environment to eliminate bias. This factor will not be necessary for the purposes of a corporate research and development project as the primary responsibility of the system is to fulfil its purpose within the confines of the requirements of the system. The technical team worked with the existing dynamic authentication research while improving on the use of neural network structures through improved training dataset sizes.

[11] https://www.behaviosec.com
[12] http://delivery.acm.org/10.1145/2990000/2984403/p72-stylios.pdf?ip=129.100.253.83&id=2984403&acc=ACTIVE%20SERVICE&key=FD0067F557510FFB%2E26B1F5E6B598D80D%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=814194254&CFTOKEN=20072440&__acm__=1506558735_59aaa469736ec1c04d866ecab54c9b36
[13] https://www.entrustdatacard.com/
[14] https://www.hindawi.com/journals/tswj/2013/408280/
[15] Ibid.
[16] Ibid.
[17] http://ieeexplore.ieee.org/document/7844242/authors
[18] http://ieeexplore.ieee.org/document/5696310/
[19] http://ieeexplore.ieee.org/document/6117480/
[20] http://www.covert.io/research-papers/deep-learning-security/Application%20of%20Recurrent%20Neural%20Networks%20for%20User%20Verification%20based%20on%20Keystroke%20Dynamics.pdf
[21] http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6515332

Companies such as Keytrac, Typing DNA, and BehavioSec have all developed commercial solutions for continuous authentication using keystroke dynamics; however, they strictly focus on web and mobile application authentication.[22] [23] [24] Application specific authentication can be considered an easier problem due to the more limited range of input. The technical team believes a large gap exists in the security marketplace for a complete PC authentication system to protect computers and their contents.

1.3.2 Facial and Iris Continuous Authentication

Facial authentication is another form of biometric authentication that was considered as an indicator. This method is a popular topic of research in continuous authentication due to the reliability of facial authentication methods. Privacy concerns make a real-world implementation challenging as users are uncomfortable with a webcam monitoring their use of the computer. To counteract this challenge, researchers have looked at developing solutions to scramble faces.[25]

Much of the work in facial authentication has been performed in research with a small number of commercial applications being built such as ULSee Face Verification for mobile platforms.[26] Samsung and Apple have both used facial authentication as a source of initial authentication; however, they do not use it to continuously authenticate the session.[27] [28]

[22] https://www.keytrac.net/en/
[23] https://typingdna.com
[24] https://www.behaviosec.com/
[25] http://ieeexplore.ieee.org/document/7084305/
[26] https://ulsee.com/en/products/face-verification
[27] http://www.samsung.com/ca/smartphones/galaxy-s8/security/
[28] https://images.apple.com/business/docs/FaceID_Security_Guide.pdf

2. Project Objectives, Scope and Constraints

2.1 Objectives

2.1.1 Accuracy

The solution must be accurate enough to provide additional security while not impeding customers' employee productivity. Our goal was to achieve values for FAR and FRR of 0.005% and 4.833% respectively, as achieved by Ahmed and Traore.[29] This proved to be difficult given the much larger scope of our project and the lack of confinement to an academic setting. As such, our goal evolved to a system that locked out 99% of imposter sessions, while allowing for at least 2000 keystrokes on average prior to an erroneous lockout of an actual user.

2.1.2 Performance

The solution must meet performance metrics to be usable by employees working on average enterprise laptops and desktops. There are two relevant metrics:

- Speed of authentication. We want to be able to lock out imposters by the time they type 50 characters. This value could be changed by administrators at the cost of accuracy.
- Resource requirements. Using a standard enterprise laptop model (Lenovo T430s),[30] we want to consume no more than 5% of the CPU's time, require no more than 100MB of disk storage, and require no more than 50MB of RAM.

2.1.3 Portability

Most major business (outside of niche ones such as software development or design) rely on Windows computers. Therefore, our solution must, at the very least, be fully executable and operational on Windows 7 or later operating systems. In future iterations of the project, it is expected that the software will be fully runnable on the other two major operating systems: macOS and Linux.

2.1.4 Usability

Good security software should be working seamlessly in the background and should not be a burden to users or the administration team. In the event of a false rejection, however, when a user is mistakenly locked out, the steps the user needs to take in order to regain access to their computer should be immediately obvious and intuitive. This is a combination of how the lock screen is designed including the clarity of steps the user can take, as well as the user manual which can be found later in the report. The locked-out user can re-authenticate their computers in three ways: contacting their administrator, or by re-authenticating themselves using Google Authenticator or SMS.

2.2 Scope

2.2.1 Limiting the Scope to the Minimum Viable Product

Due to the novelty of the technology and our inexperience with the techniques required, the effort needed to complete several features of the project was uncertain. The minimum viable product (MVP) was a useful structure to identify the use cases that are most critical to our project. By identifying features that are part of the MVP and focusing on completing those features first, we maximized our chances of accomplishing the objectives set out in the previous section.

[29] http://ieeexplore.ieee.org/document/6515332/
[30] https://www.laptopmag.com/reviews/laptops/lenovo-thinkpad-t430s

2.2.2 Minimum Viable Product Definition

2.2.2.1 Ability to Lock Out User Based on Keystrokes

Our solution is an application that runs in the background of the user's machine. It performs continuous authentication via free-text analysis. To perform authentication, the solution compares the user's typing profile to their current keystrokes. The typing profile is a machine learning classifier that decides whether a pattern of keystrokes is characteristic of the user. The solution builds this profile using a machine learning algorithm that will continuously refine the profile. This component is the core innovation of the project and determined the extent to which the objectives are achieved.

2.2.2.2 Lock Screen

When an uncharacteristic sequence of key presses is encountered, the machine is locked. The keyboard and mouse will still function, but the user will not be able to close the lock screen. For example, using ALT+TAB will not work nor will opening the task manager via CTRL + ALT + DELETE. Further, if the user has multiple screens connected to their computer, the lock screen will be present on all screens. The user has three ways to unlock their computers and continue on with their work. First, they can contact their IT personnel and/or administrator who can use the client portal to unlock the computer instantaneously. Alternatively, the locked user can re-authenticate themselves via SMS or Google Authenticator.

2.2.2.3 Administrative Portal

An administrative portal will be available for the admin/IT personnel team within the organization and includes a number of features. First, it lists all the user accounts within the organization and includes information for each user. This information includes whether or not their computer is currently locked, whether or not the user is currently online, and a graph showing their typing profile scores as a function of time. The admin will have full access to CRUD (create, read, update and delete) operations for users. Second, the admin team will have the ability to view the full list of machines that exist within the organization. As with users, the admin will have full access to CRUD operations for machines.

2.2.3 What's Not Included in the MVP

We consider everything not in the MVP to be out-of-scope for this project due to the constraints mentioned below in the report.

We have identified a number of ways to extend the overall functionality of our solution:

- Continuous authentication via mouse movements: It's quite clear that a big assumption in our implementation of our MVP of BioKey is that it assumes nefarious activity involves using the keyboard to a large degree. This is not necessarily true as one can do a lot of damage just through mouse movements and clicks. Therefore, using mouse movement profiles for users simultaneously with keystroke profiles will significantly improve the overall efficacy of BioKey.
- Implementation across other operating systems: The current MVP is limited to computer systems that run Windows 7 or later. In order to increase the total addressable market of BioKey, it would be beneficial to ensure BioKey has full functionality on macOS and Linux.
- Disabling port input: The current MVP fails to preclude a user's computer from reading information from input ports (e.g. USB keys) when the computer is locked. It's possible that an imposter could load a script on a USB key to override the lock screen and thus circumvent the software. This is clearly a significant threat to the efficacy of BioKey and should most certainly be addressed in future iterations.

2.3 Constraints

2.3.1 Time

Time constrained our project in two ways. First, we only had approximately six months to develop our solution. Second, since our solution depends on machine learning, we first had to collect a large enough data set while leaving ourselves enough time to train and test the solution. To address this constraint, we diligently planned the data procurement and development phases of the project.

2.3.2 Budget

We had no allocated budget for this project, which constrained our access to some resources. To address this constraint, we maximize our use of open source tools and libraries to build our solution. In addition, we used student credits in order to access additional resources through AWS.

2.3.3 Data Privacy

Our solution tracks keystrokes which are sensitive information and require protection. We can address this constraint in two ways. First, we can encrypt and store keystroke data locally and never transmit them. Second, we can delete redundantly stored keystrokes in normal operation. Once a sequence of keystrokes is verified by the solution and the machine learning component has "studied" it, the keystrokes can be discarded. The storage and transmission of keystroke data will always be encrypted.

Software Requirements Specification

BioKey

Version 2.0

REVISION HISTORY

Version	Primary Author(s)	Description of Version	Date Completed
0.1	Joshua Weinstein & Sankalp Hariharan	Completed Introduction and Overall Description.	24/11/17
0.2	Joshua Weinstein	Added walkthrough presentation as notes to Specific Requirements and Other Non-Functional Requirements.	25/11/17
0.3	Tony Wu	Added detail and edited document.	26/11/17
0.4	Tony Wu	Edited Specific Requirements after consulting SDS and Partial Test Plan.	27/11/17
1.0	Joshua Weinstein & Sankalp Hariharan	Completed a final edit based on all available information on requirements.	29/11/17
1.1	Martin Kucera & Connor Giles	Began update draft for Final Report. Updated Sections 1 and 2.	4/4/18
1.2	Martin Kucera & Connor Giles	Continued update draft for Final Report. Updated Sections 3 and 4. Ready for touch-ups.	5/4/18
1.3	Martin Kucera	Completed update draft.	6/4/18
2.0	Martin Kucera & Sankalp Hariharan	Completed version.	8/4/18

3 SRS - Introduction

3.1 Document Purpose

The purpose of this document is to provide a basic description of the software system. Essential sections include *3.2 Product Scope*, *4.2 Product Functionality*, *5.2 Functional Requirements*, and *5.3 Behavioural Requirements*. After reading this document, the reader should understand the services that the application provides at a high level. *The reader is encouraged to use this paragraph to refer to the important sections of the document.* The document pertains explicitly to the version of BioKey which the technical team has completed as of April 11, 2018.

3.2 Product Scope

BioKey's target customers are enterprises that regard informational security and protection of intellectual property as paramount. The primary purpose of the software is to complement existing security solutions as a second factor of authentication.

BioKey's customer-facing product consists of two parts: a locally installed client on users' computers, and a admin portal to administer these users.

The local client module leverages keystroke dynamics to provide continuous authentication. This is an application of academic research that is novel to BioKey - most current keystroke dynamics solutions do not provide continuous authentication. If an unauthorized user attempts to type on the computer, the software locks the computer, preventing the user from taking any further action. If the computer is connected to the internet, the client module notifies the administrator through the web application. The user is then provided a challenge that, when passed, unlocks the computer. If no connection is available, the computer remains locked until an administrator unlocks it.

The admin portal provides a central dashboard for administrators to monitor their organization's user activity. Administrators can remotely lock and unlock computers in their organization. Furthermore, non-administrator users can log in to the web application to view their profile and recent activity. The admin portal is enabled by an API provided by a backend web server.

3.3 Intended Audience and Document Overview

This document is intended for two audiences: security and technology teams within enterprise customers, and academic readers as part of SE 4450.

The first intended audience is comprised of enterprise customers. Since the target customers are enterprises who have the capital and incentives to protect their information, it is vital to provide this document to their security team. This document will allow potential customers to determine whether BioKey is a good fit for their needs.

The second party of consequence are the academic conductors of SE 4450. At a high level, the document will outline the work that the technical team has done over the academic year give evaluators confidence that best practices have been followed. This document will also provide evidence to the faculty supervisor that their recommendations have been followed.

The remainder of the document is organized into three sections. Section 2 describes the high-level requirements and characteristics of the system. Section 3 describes the functional requirements of the system. Section 4 describes the non-functional requirements and constraints of the system.

3.4 Definitions, Acronyms, and Abbreviations

Term	Meaning (if acronym)	Definition
API	Application Programming Interface	A set of defined methods of communication between software components.[31]
AWS	Amazon Web Services	The set of cloud services offered by Amazon.
False Acceptance	-	An instance of an imposter being accepted by the system.
FAR	False Acceptance Rate	The probability that an imposter will be accepted by the system.
False Rejection	-	An instance of a legitimate user being rejected by the system.
FRR	False Rejection Rate	The probability that a legitimate user will be rejected by the system.
SMS	Short Messaging Service	Text messaging.
SQS	Simple Queue Service	A server implementation of a queue offered by AWS.

3.5 Document Conventions

IEEE formatting requirements (size 11 Times New Roman font and 2.54 cm margins) were used.

3.6 References and Acknowledgements

The following are a number of research papers that have helped the technical team develop this project:

- Application of Recurrent Neural Networks for User Verification based on Keystroke Dynamics
    - http://www.covert.io/research-papers/deep-learning-security/Application%20of%20Recurrent%20Neural%20Networks%20for%20User%20Verification%20based%20on%20Keystroke%20Dynamics.pdf
- Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences
    - https://arxiv.org/pdf/1610.09513
- A Survey of Keystroke Dynamics Biometrics
    - http://downloads.hindawi.com/journals/tswj/2013/408280.pdf
- Typing Patterns: A Key to User Identification
    - http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1341408
- A Continuous Identity Verification Method based on Free-Text Keystroke Dynamics
    - http://ieeexplore.ieee.org/document/7844242/

[31] https://en.wikipedia.org/wiki/Application_programming_interface

4 SRS - Overall Description

The purpose of this section is to provide a high-level description of BioKey's operation. Important sections include:

- *Section 4.2 Product Functionality*, which outlines the main functions of BioKey.
- *Section 4.5 Design and Implementation Constraints*, which outlines the main assumptions and decisions that guided the design of BioKey.

4.1 Product Perspective

BioKey is a standalone system that only requires publicly available and widely-used software to operate. The BioKey client will require that the Java Runtime Environment be installed on user machines. The BioKey admin portal will require a modern web browser.

Furthermore, BioKey is not a replacement for existing systems. The technical team considered BioKey to complement the current security solutions that already exist in the market. BioKey will not need to interface with other security systems.

4.2 Product Functionality

4.2.1 Client Module

The following is a list of the *primary* functions that the BioKey client module supports:

- ***Perform local, continuous keystroke analysis.*** The client module accepts a continuous stream of user input and performs analysis.
- ***Detect suspicious behaviour.*** Through the analysis process, the client module identifies suspicious user behaviour.
- ***When suspicious behaviour is detected, lock the machine.*** When the client module identifies suspicious user behaviour, it locks the machine. This includes extended displays, keyboard shortcuts (e.g. as ALT+TAB, CTRL+ALT+DELETE), and mouse movements.
- ***Maintain the client status.*** The client module continuously updates the admin portal of changes in its analysis results and machine status, and accepts updates pushed from the admin portal. Furthermore, the client state persists through system reboots.
- ***Recover from a lost connection.*** If the internet connection is lost, the client module continues to operate. When the connection is re-established, the client module automatically applies any missed commands from the admin portal.

4.2.2 Web Application

The following is a list of the primary functions that the BioKey admin portal supports:

- ***Allow administrators to view the organization dashboard.*** Administrators have access to a page that summarizes the status of users in their organization. This includes users' lock/unlock status and online/offline status. The dashboard also provides a link that allows users to make an account for their organization.
- ***Allow admins to remotely lock or unlock user machines.*** Administrators can lock or unlock user machines from the administrative dashboard or from user profile detail pages.
- ***Allow admins to manage entities in their organization.*** Administrators can view all details of users, user typing profiles, machines, and activities within the organization. Administrators can also make changes to many of these entities.
- ***Allow users to manage their own profiles.*** Non-administrator users are provided a personal details page through the web application. Users can make limited updates, such as changing their phone number and password.

These details of these functions will be expanded upon in Section 6.

4.3 Users and Characteristics

BioKey will have two types of users: regular enterprise users and enterprise security staff. It is important that BioKey satisfies both user types.

Regular enterprise users ideally interact very minimally with BioKey. The client module runs in the background of their machine, and they very rarely need to access the web application. This user requires that BioKey does not interrupt their routine tasks through false rejections. Since BioKey will be installed on every enterprise machine, it is crucial that false rejections do not materially impact the productivity of enterprise employees.

Security staff are characterized by their high level of security authority compared to regular users. Furthermore, this subset of user interacts potentially very frequently with the BioKey admin portal to monitor organizational security. Security staff prefers that BioKey be as accurate as possible by minimizing false acceptances. It is vital that both the FAR and the FRR are minimized as they determine the value that BioKey creates.

4.4 Operating Environment

4.4.1 Client Module

BioKey's client module has been designed and tested for PCs running Windows 10. Other than what is listed in Section 2.1, no specific software components, applications, or minimum platform specifications are required for its operation.

4.4.2 Web Application

BioKey's web application has been designed and tested for Google Chrome 65. No specific software components or applications are required for its operation.

4.5 Design and Implementation Constraints

When deployed, BioKey is a critical and complex component of an organization's security. As such, several constraints were identified to guide implementation and design:

- ***Hardware constraint: access to sufficient GPU processing power.*** Training CA models comes at significant computational, and therefore financial, cost. The training algorithm was designed to be as computationally efficient as possible to minimize this cost.
- ***Quantity of keystroke information available.*** Ideally, keystroke information should be collected from hundreds of individuals to improve testing of the model. Due to the short time frame, data from around 20 people suited our needs.
- ***Limited machine learning expertise.*** The field is new to most of the technical team, which led to significant time spent on research trial-and-error.
- ***Time limitation.*** As a course project, relatively limited time was available to iterate open the solution and will especially be problematic in testing and accommodating of edge cases
- ***Security constraints.*** The technical team's limited experience with sophisticated penetration tests increases the possibility of security failures.
- ***Security protocols.*** All web communications are facilitated using the secure HTTPS protocol.

4.6 User Documentation

Due to the relatively simply nature of user interactions with BioKey, a user manual is the only necessary documentation to provide. This user manual is provided in Part 8 of BioKey's final report. The manual is positioned towards organizational security teams to decrease the friction of installing the client module on machines and using the admin portal. Regular users do not need an instruction manual and any questions they have should be taken up with their administrator.

4.7 Assumptions and Dependencies

Some simplifying assumptions were made to reduce the complexity of BioKey's implementation:

- ***Existing security.*** It was assumed that current security platforms implemented by enterprises do not preclude BioKey from saving keystroke data.
- ***Enterprise hardware.*** It was assumed that most enterprises issue employee machines with the Windows 10 operating system, the Java Runtime Environment, and Chrome 65.
- ***Typing behaviour.*** It was assumed that regular users use machines solely for business applications and will not change their typing behaviour drastically by using the computer for other applications (e.g. gaming).
- ***Machine ownership.*** It was assumed that machines are assigned to, and used by, one employee at a time.
- ***AWS commercial components.*** BioKey depends on the AWS Simple Queue Service for client-server communications and AWS Elastic Compute Service for web hosting.
- ***Twilio commercial components.*** BioKey depends on Twilio's SMS Service for sending mobile administrator alerts.

5 SRS - Specific Requirements

The purpose of this section is to provide a more granular description of BioKey's operation than what was presented in Section 2. Important sections include:

- *Section 5.2 Functional Requirements*, which elaborates upon the functionality identified in Section 4.2.
- *Section 5.3 Behaviour Requirements*, which outlines the main use cases to implemented.

5.1 External Interface Requirements

5.1.1 User Interfaces

5.1.1.1 Web Application

The web application's GUI is built using Ember.js. A persistent navigation bar is shown on the left side of the screen. A persistent title bar is shown at the top of the screen with a button to navigate to the BioKey homepage. Standard buttons include Cancel, Save, and Back buttons. Please refer to Figures 1 and 2 for sample screenshots.

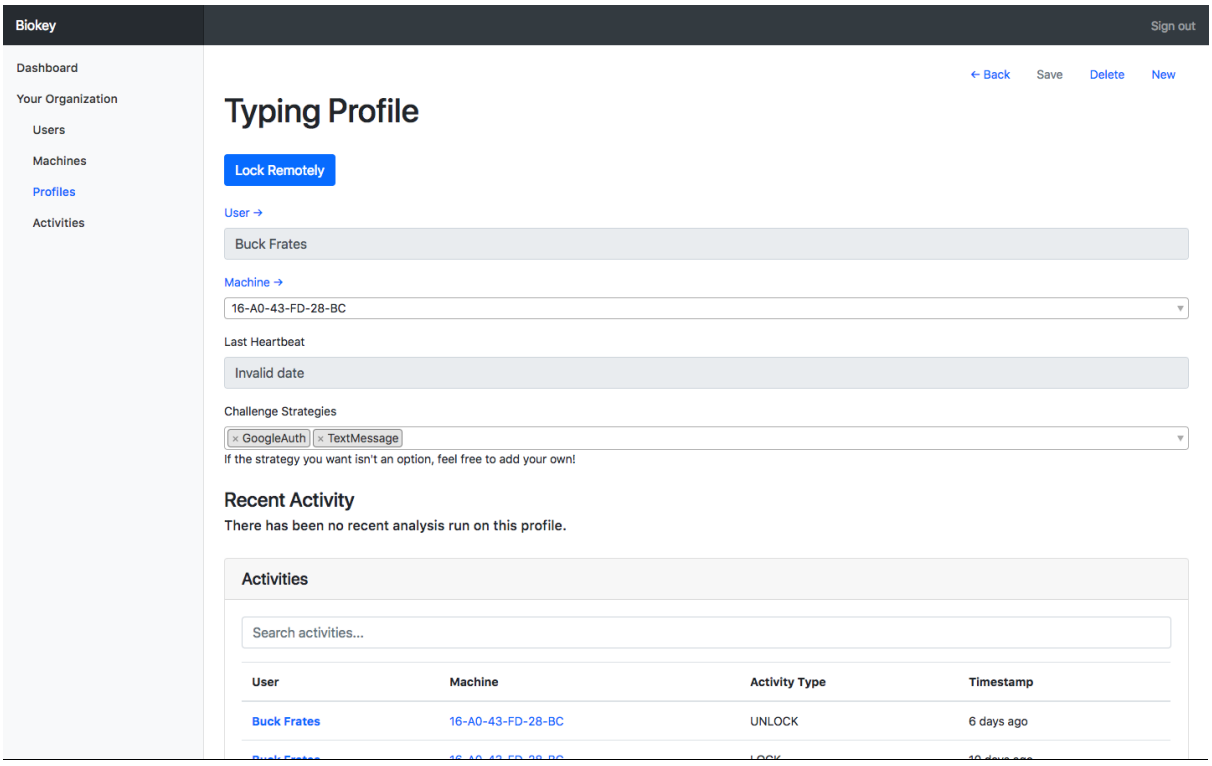


Figure 1: A typical detail page on the BioKey web application

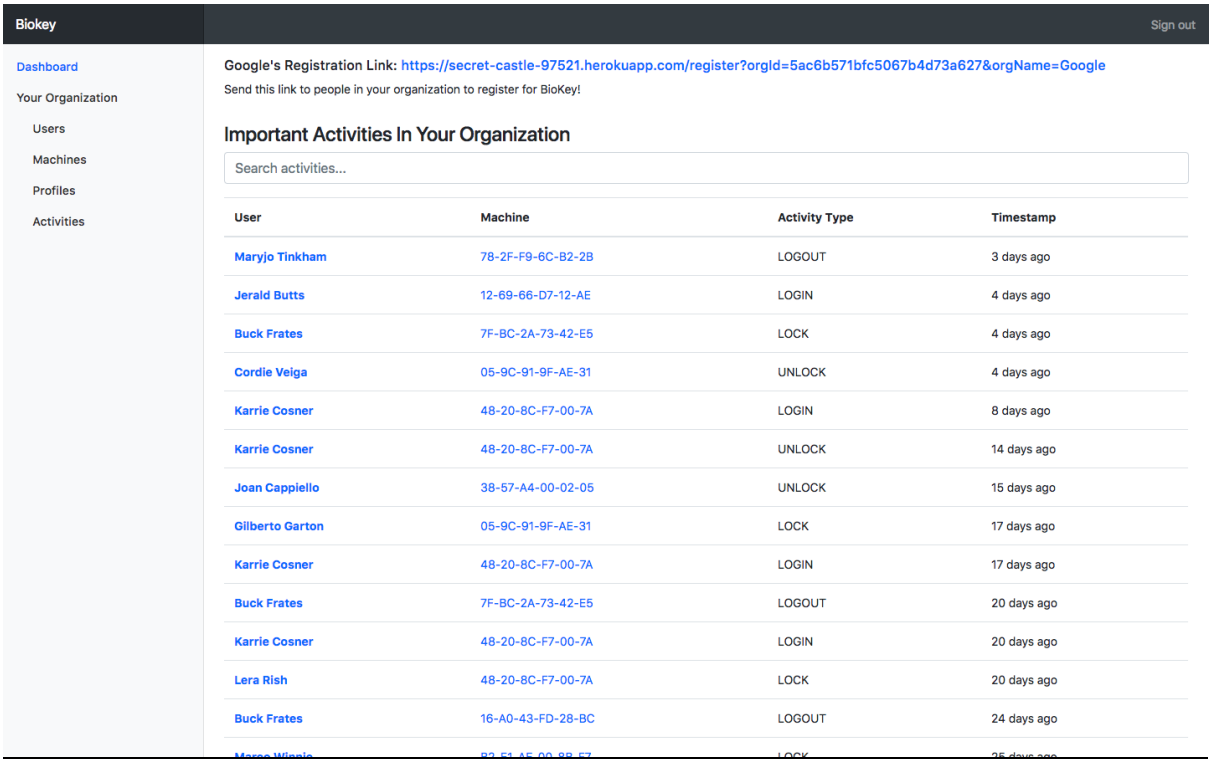


Figure 2: Administrator's organization dashboard

5.1.1.2 Client Taskbar Display

The client module was built using the Java Spring framework. The taskbar icon will be visible when a user's machine has BioKey installed and running. When clicked, the user can see a graphical depiction of their analysis results, shown in Figures 3 and 4.

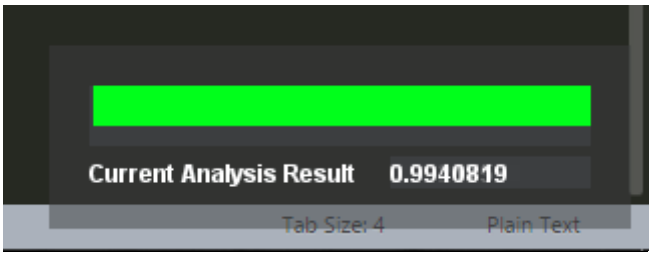


Figure 3: Expanded taskbar icon showing a favourable analysis result

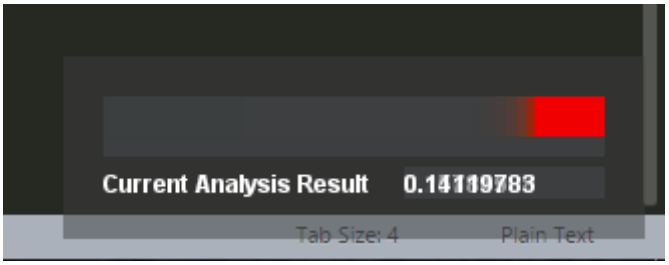


Figure 4: Expanded taskbar icon, showing an unfavourable analysis result

5.1.1.3 Client Lock Page

The lock screen is shown when a user is locked out of their machine. It is imperative that the lock page take up the entire screen. The lock screen has minimal styling as it is simply meant to convey important information to the user. Please refer to Figures 5 through 8.

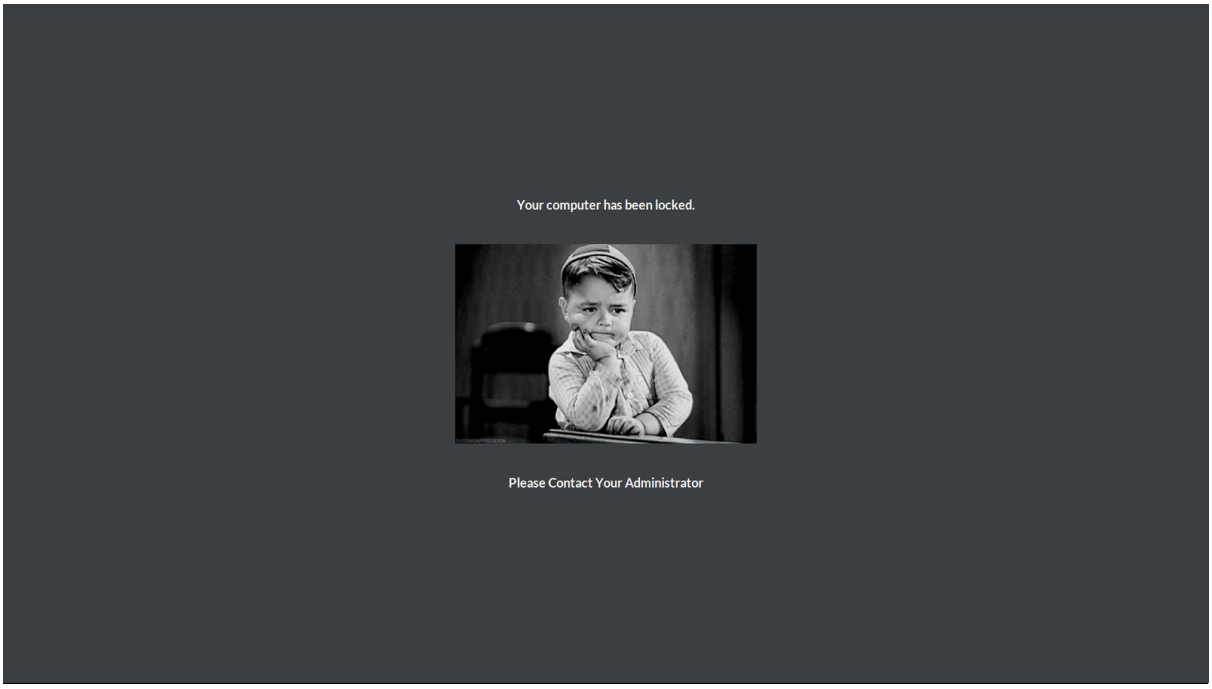


Figure 5: Lock screen, locked by an administrator

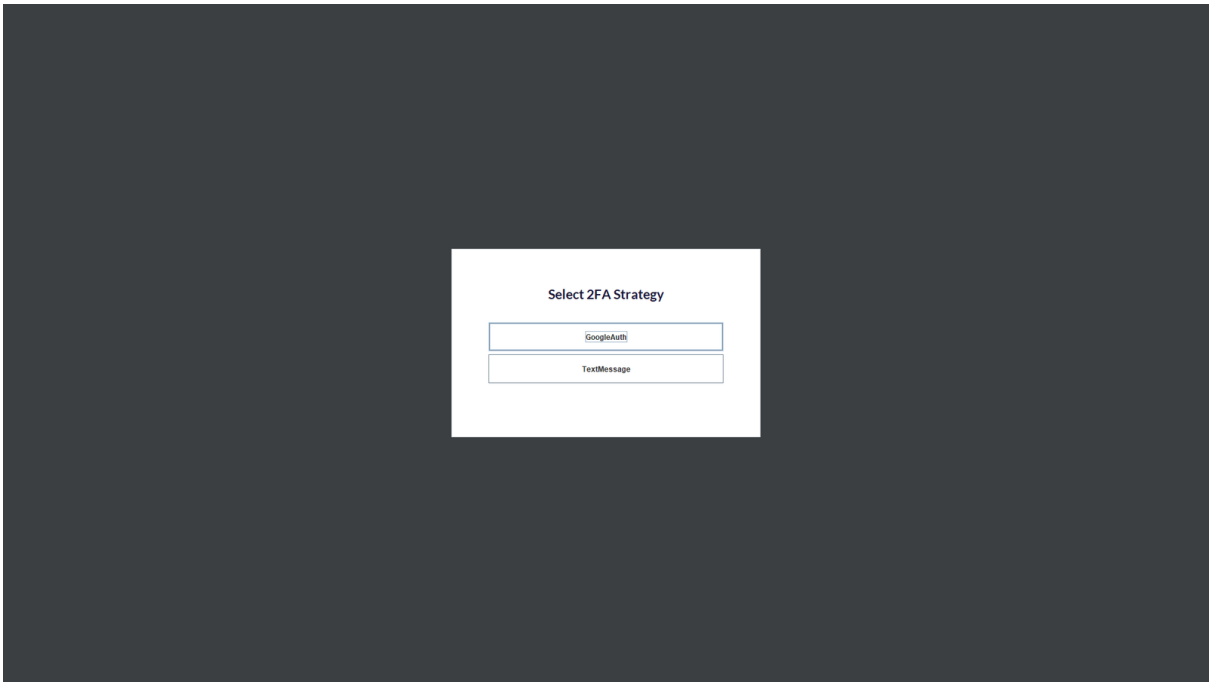


Figure 6: Lock screen, locked by analysis

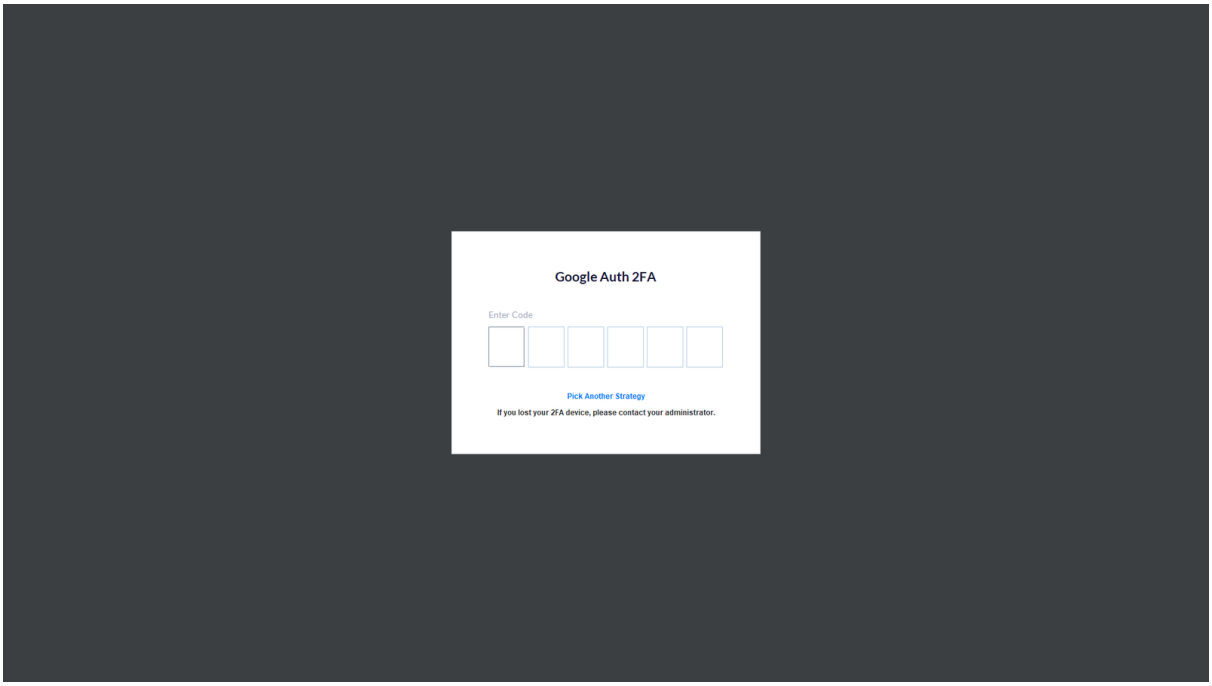


Figure 7: Challenge screen, re-authenticating via Google Auth

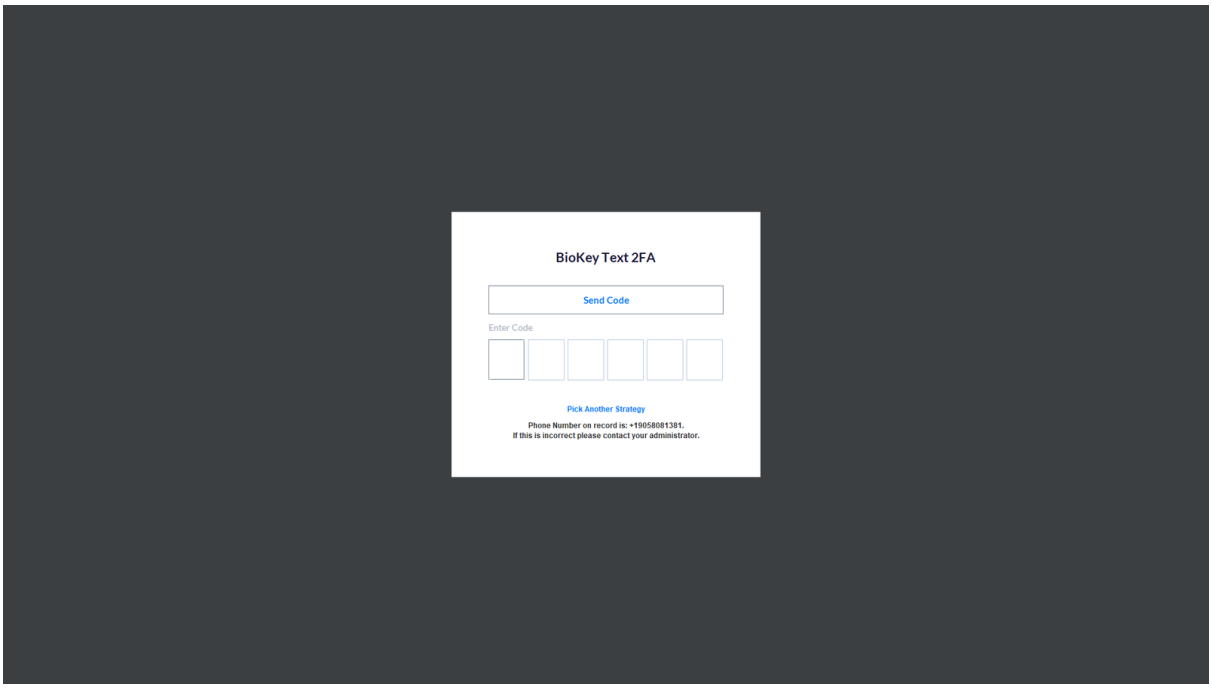


Figure 8: Lock screen, re-authenticating via text message

5.1.2 Hardware Interfaces

Two hardware interfaces are essential to BioKey:

- ***Keystroke capture.*** BioKey uses the JNativeHook library for keystroke bindings.
- ***Machine lockout.*** When a user is locked out of the machine, BioKey uses the Java.AWT library to lock the keyboard and the JavaX.Swing library to lock the monitor.

5.1.3 Software Interfaces

As stated in Section 2.4.1, BioKey is designed and tested to run on a PC with the Windows 10 operating system. Furthermore, the hardware interfaces defined in Section 3.1.2 are facilitated by the operating system.

5.1.4 Communications Interfaces

For web communications, HTTPS will be used extensively. It will be used for communication from the client to the web server and between the web server and the web application. The ciphersuite chosen for HTTPS will be strong and considered in the TLSv1.2 family.

In addition, the commercial components that BioKey uses support HTTPS encryption. Communication from the web server to the client will be facilitated through Amazon SQS, which adheres to the highest encryption standards. Administrator alerts are sent through Twilio SMS messages, which are encrypted in transit with HTTPS.[32]

5.2 Functional Requirements

5.2.1 Client Module: Perform local, continuous keystroke analysis

Requirement	Explanation
Record keystroke data (1)	Typed keystrokes are recorded while the user is logged in. This recording is carried out regardless of the network connection status.
Analyze keystroke data (2)	Recorded keystrokes are analyzed using the machine learning model. The resulting output is a score representing the percentage chance that the analyzed keystrokes were generated by the legitimate user. This recording is carried out regardless of the network connection status but will not run in the absence of an authenticated user. The client module streams the analysis results to the server.

5.2.2 Client Module: Detect suspicious behaviour

Requirement	Explanation
Assess analysis output (3)	The output of the analysis is compared against the user's "threshold value". This value represents that certainty that the administrator requires in authenticating users. If the analysis output falls below the threshold value, the client module recognizes this as suspicious behaviour and locks the machine.

5.2.3 Client Module: Lock the machine

Requirement	Explanation
Lock the machine (4)	When the client module detects suspicious behaviour, it locks the machine, regardless of network connectivity. A locked machine cannot accept input from the keyboard or mouse, nor can it take away focus from the lock screen. A locked machine does maintain networking ability.
Issue challenges and	If configured, and if a network connection exists, the client module issues a challenge

[32]https://www.twilio.com/blog/2016/05/introducing-end-to-end-encryption-for-twilio-ip-messaging-with-virgil-security.html

determine results (5)	prompt when the machine is locked. The user can pass the challenge (Google Auth or Text Message) to unlock the machine. If the user fails the challenge, the client module requires administrator action to unlock.
Unlock the machine (6)	If a user passes a challenge or an administrator performs the appropriate action, the client modules unlocks the machine. An unlocked machine resumes normal operation.

5.2.4 Client Module: Maintain client status

Requirement	Explanation
Maintain login status of a user (7)	Authenticated users maintain their authenticated status when the machine reboots. This status also includes whether the machine is locked. The login status cannot be changed by the user through the desktop client.
Maintain state agreement with the server (8)	The client module and web server maintain a heartbeat that allows for agreement of state changes. The state recorded by the server takes precedence if there is a conflict. The state includes the profile, authentication status, lock status, access token, keystroke data, and previous state data yet to be synced. The heartbeat occurs once per minute.
Respond to server messages from administrator (9)	The client listens for server messages containing state changes and acts accordingly. The server can ask the client to change any component of its state. This only works while the machine has a network connection.

5.2.5 Client Module: Recover from a lost connection

Requirement	Explanation
Update state after connection is re-established (10)	If the machine goes offline, the heartbeat is paused. When the network connection is re-established, the client module resumes the heartbeat, sends its new state to the server, and executes any waiting server requests. A new state is then agreed upon.

5.2.6 Admin Portal: Allow administrators to view dashboard

Requirement	Explanation
View activity logs for users within organization (11)	The dashboard summarizes all changes made to assigned machines, account information, typing profiles, and client state data of the organization's users.
Notify administrator of important activity for users within organization (12)	The dashboard and SMS infrastructure notifies the administrator of changes in the activity log that may be of concern. These include warnings coming from client machines.

5.2.7 Admin Portal: Allow administrators to remotely lock or unlock machines

Requirement	Explanation
Remotely lock or unlock machines (13)	Administrators can lock or unlock user machines from the administrative dashboard or from user profile detail pages.

5.2.8 Admin Portal: Allow administrators to manage entities

Requirement	Explanation
Create administrator account and organization (14)	New administrators can create an account which automatically creates an associated organization.
Assign users within organization as administrators (15)	Administrators can assign administrator status to users within the organization.
Create or delete users (16)	Administrators can create and delete user accounts from the organization. A deleted user's account is deactivated.
Edit user accounts (17)	Administrators can edit user account information such as the password, and phone number.
Create typing profiles (18)	Administrators can create a new typing profile for a user and assign a machine to the typing profile.
Edit typing profiles (19)	Administrators can edit the settings of typing profiles, such as accepted challenge strategies.
Edit organization settings (20)	Administrators can change settings such as the default security threshold, accepted challenge strategies, and alert settings for their organization.

5.2.9 Admin Portal: Allow users to manage their own profiles

Requirement	Explanation
Create regular user account using company email and a unique password (21)	Users can create an account with an existing organization. Using a unique email and password. For security, the password is not stored directly - it is salted and hashed. This is elaborated upon in Section 4.2.
Regular users can edit their profile but cannot see their activity (22)	Users can login to their account and edit their profile information such as their phone number and password. They cannot perform any other tasks such as viewing their recent activity.

5.3 Behaviour Requirements

5.3.1 Use Case View

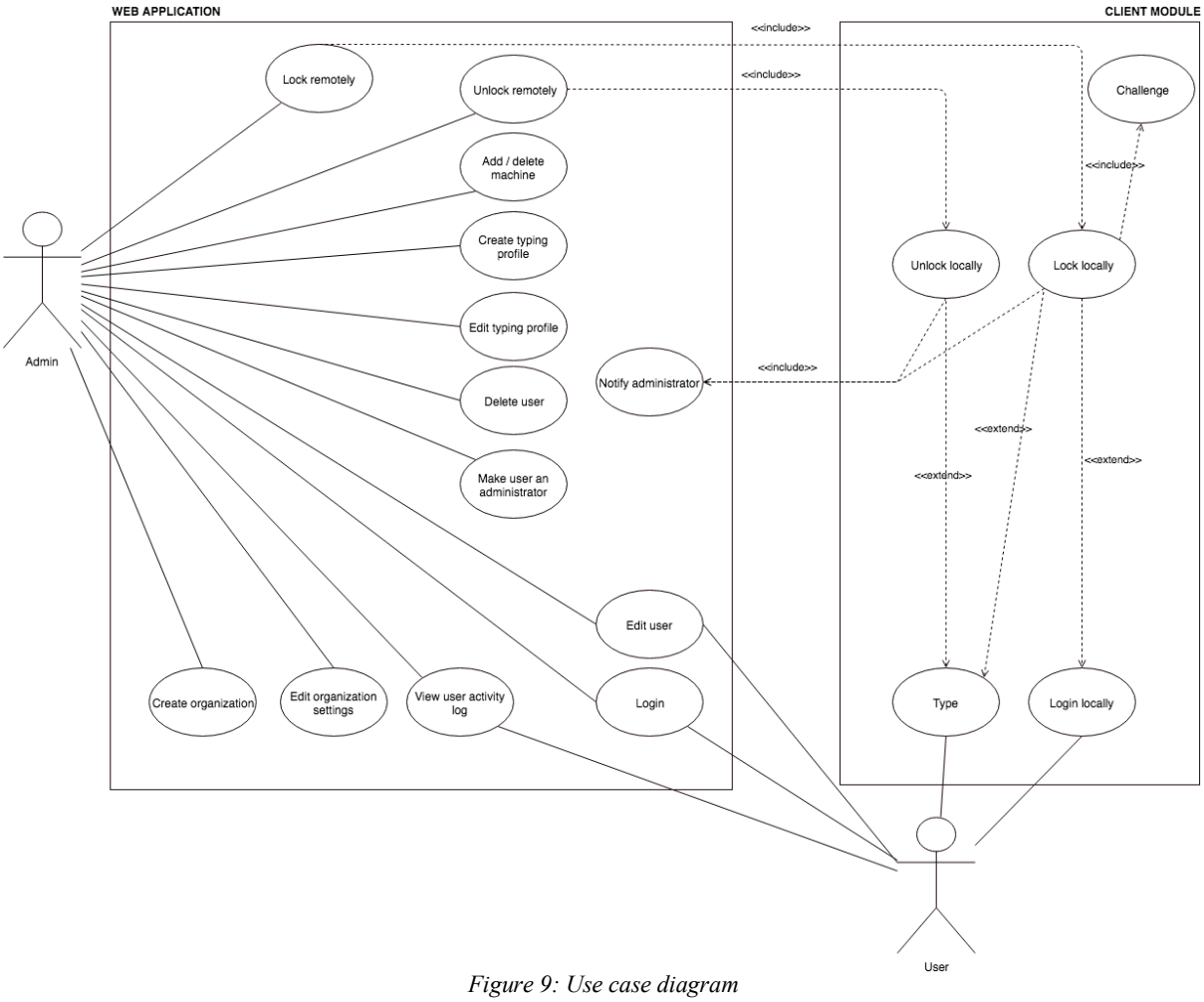


Figure 9: Use case diagram

5.3.2 Use Case Descriptions

Use Case	Actor	Description
Lock remotely	Admin	From the admin portal, send a message to remotely lock a client.
Unlock remotely	Admin	From the admin portal, send a message to remotely unlock a client.
Add / delete machine	Admin	From the admin portal, add or delete a machine from the database.
Create typing profile	Admin	From the admin portal, create or a typing profile associated with a user.
Edit typing profile	Admin	From the admin portal, edit the typing profile of the user such as accepted challenge strategies.
Delete user	Admin	From the admin portal, delete a user's profile.
Make user an	Admin	From the admin portal, change a user's permissions to match those of the

administrator		administrator.
Create organization	Admin	Create a new organization to logically group users.
Edit organization settings	Admin	Change the settings of an organization such as the default challenges strategies and thresholds.
View activity log	Admin	View the recent activity of a user organized chronologically such as recent lock / unlock events.
Edit user	Admin, User	From the web portal, edit user account information such as the password, and phone number.
Login	Admin, User	Log into the web portal.
Type	User	User types a key. May lead to a change in the OS lock state.
Login locally	User	Log into a machine that is associated with the user.
Lock locally	-	Unlock the machine locally after the detection of suspicious typing behaviour. May trigger a challenge to be issued.
Unlock locally	-	Unlock the machine locally via authenticating through challenge or administrator action.
Challenge	-	Issues the user a challenge to unlock the OS.
Notify administrator	-	Sends an SMS message informing the administrator of a client action.

6 SRS - Other Non-Functional Requirements

The purpose of this section is to provide BioKey's non-functional requirements. Important sections include:

- *Section 6.1 Performance Requirements*, which provides quantifiable metrics for the functional requirements specified in Section 3.2.
- *Section 6.3 Software Quality Requirements*, which provides important quality considerations for BioKey's design.

6.1 Performance Requirements

Performance Requirements can be related directly to Functional Requirements detailed in Section 3.2 and provide specific metrics by which to measure the performance of these functions. Testing against these metrics was conducted using a Lenovo T430s.

6.1.1 Client Module: Perform local, continuous keystroke analysis

Requirement	Metric	Explanation
Record keystroke data (1)	CPU usage	At any time, the client module must use no more than 10% of CPU time on our benchmark machine.
Record keystroke data (1)	Memory usage	At any time, the client module must use no more than 100MB of disk memory or RAM on our benchmark machine.
Analyze keystroke data (2)	% missed keys	The client module must analyze keystrokes quickly enough so that 0% of keys are missed.

6.1.2 Client Module: Detect suspicious behaviour

Requirement	Metric	Explanation
Assess analysis output (3)	# of keys to detect imposter	The client module must detect that the user is an imposter within 50 keystrokes.
Assess analysis output (3)	FAR	The client must commit false acceptance at a rate less than 0.5%.
Assess analysis output (3)	FRR	The client must commit false rejection at a rate less than 0.05%.

6.1.3 Client Module: Lock the machine

Requirement	Metric	Explanation
Lock the machine (4)	Time to lock	The client module must lock the machine immediately upon detection of suspicious behaviour.
Issue challenges and determine results (5)	Time to issue challenge	The client module must issue the challenge immediately machine lock.
Unlock the machine (6)	Time to unlock	The client module must unlock the machine immediately if a challenge was passed locally, or within 10 seconds if unlocked via administrator action.

6.1.4 Client Module: Maintain client status

Requirement	Metric	Explanation
Maintain login status of a user (7)	N/A	-
Maintain state agreement with the server (8)	N/A	-
Respond to server messages from administrator (9)	Time to change	An online client must notice a server message and executing its contents within 10 seconds of it being enqueued.

6.1.5 Client Module: Recover from a lost connection

Requirement	Metric	Explanation
Update state after connection is re-established. (10)	Time to verify state	An offline client that regains an internet connection must resume a heartbeat with the web server within five seconds of restoration of network connectivity.

6.1.6 Admin Portal: Allow administrators to view dashboard

Requirement	Metric	Explanation
View summary of user details (11)	N/A	-
View activity logs for users within organization (12)	N/A	-
Notify administrator of important activity for users within organization (13)	Time to notify	The client module must notify the administrator within 10 seconds of the action taking place.

6.1.7 Admin Portal: Allow administrators to remotely lock or unlock machines

Requirement	Metric	Explanation
Remotely lock or unlock machines (14)	Time to lock/unlock	The client module must lock or unlock the machine within 10 seconds of the administrator action.

6.1.8 Admin Portal: Allow administrators to manage entities

Since all functional requirements under this heading are basic CRUD operations, no performance metrics apply.

6.1.9 Web Application: Allow users to manage their own profiles

Since all functional requirements under this heading are basic CRUD operations, no performance metrics apply.

6.2 Safety and Security Requirements

Since BioKey's value proposition is increased security, the application itself must confirm to high security standards. This is a complex requirement to meet given the handling of sensitive data, communication between many modules.

- ***Communication encryption.*** As discussed in Section 3.1.4, all web communications must be encrypted in a method that is computationally infeasible to crack using HTTPS.
- ***Data storage.*** All data stored on the client module or web server must either not be personally identifiable to the user or be encrypted in a method that is computationally infeasible to crack. Furthermore, user passwords are not stored directly - they are salted and hashed.
- ***User consent***. Since BioKey collects sensitive information (in the form of keystrokes), it is crucial that enterprise clients give their consent.

6.3 Software Quality Attributes

6.3.1 Functionality

Specific functionality and related performance requirements are listed in Sections 3.2 and 4.1, respectively.

6.3.2 Reliability

BioKey relies on two aspects of reliability: recoverability and fault tolerance.

As an enterprise security application, it is crucial that BioKey is recoverable. The BioKey client module recovers from failure by automatically restarting after system failures or reboots and resuming the state from the local backup files. The client also notifies the server of when it loses network connectivity as a result of such an event. The web application can very easily be rebooted in the event of a failure since it is designed as a stateless application.

Fault tolerance is also important for BioKey because it interfaces with the operating system. All possible faults and exceptions that don't result in the process being killed are caught and handled. In the case that there is a fault with the server, the client simply acts as if it was offline, and no fault occurs. Finally, the BioKey client module detects corrupted local files, avoiding faults from that origin.

6.3.3 Usability

BioKey relies on three aspects of usability: understandability, learnability, and operability.

The software should be easily understood by both users and security staff. Users do not normally interact with the BioKey client module - the only case is when their machine is locked. When this occurs, the lock screen clearly communicates what has happened, as well as the series of steps to unlock the machine. The understandability of the web application is provided by the persistent navigation bar and intuitive layout, as depicted in Section 3.1.1.1.

BioKey must also be easily learned. Regular users undergo no training to interact with BioKey's client module - the steps to unlock the computer should be very clear. Furthermore, regular users' web applications consist of just one page with under 10 fields, making it very easy to learn and understand. Administrators have more web functionality available to them and thus will have more to learn. Administrators should be able to fully understand the hierarchy, navigation, and functionality of BioKey after reading the training manuals for around 15 minutes.

Finally, the software should be easily operated. Since interactions with BioKey are the equivalent to browsing web pages, BioKey is very easily operated.

6.3.4 Efficiency

Several requirements related to efficiency were given in Section 4.1. To summarize, BioKey must meet its functional and non-functional requirements on a standard enterprise laptop and work with only limited time and resources. This is achieved through the use of efficient data structures, algorithms, and protocols.

6.3.5 Maintainability

BioKey relies on two aspects of maintainability: analyzability and testability. BioKey was made to be easily analyzed by organizing classes into dedicated services. Furthermore, the BioKey client module has a single source of truth for client state, and all services subscribe to changes to this source. Furthermore, there is only one service for the client to modify the state to avoid race conditions. Finally, BioKey's service-oriented architecture is easily unit-tested, providing testability.

6.3.6 Portability

BioKey relies on two aspects of portability: adaptability and installability. To be a complete part of enterprise's security, the application must run adaptably on all standard operating systems. Using platform independent technologies such as Java has helped BioKey achieve this goal. Furthermore, BioKey is easily installed on machines via a single executable. The installation process takes less than one minute per client.

Software Design Specification

BioKey

Version 2.0

REVISION HISTORY

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Martin Kucera and Tony Wu	First draft of design specification.	24/11/17
1.1	Martin Kucera and Tony Wu	Updated architectural design.	25/11/17
1.2	Martin Kucera and Tony Wu	Updated sequence diagram.	26/11/17
2.0	Martin Kucera, Tony Wu and Sankalp Hariharan	Updated the ER diagram, class diagram + descriptions, added server route descriptions, and added admin client descriptions.	05/04/18

7. SDS - Introduction

7.1 Purpose of this Document

The purpose of this document is to provide the reader with clear architectural, logical, and structural design specifications for BioKey. Further, the document is to provide a rationale and justification for the design choices that were made. As it provides the low-level description of BioKey's implementation, readers should find all sections to be relevant.

7.2 Scope of the Development Project

BioKey's target customers are enterprises that regard informational security and protection of intellectual property as paramount. The primary purpose of the software is to complement existing security solutions as a second factor of authentication.

BioKey's design consists of three parts:

1. The ***client module***, locally installed on users' machines
2. The ***server***, which provides the database and communicates with commercial components
3. The ***admin portal***, which security staff can use to administer users

The local client module leverages keystroke dynamics to provide continuous authentication. This is an application of academic research that is novel to BioKey - most current keystroke dynamics solutions do not provide continuous authentication. If an unauthorized user attempts to type on the computer, the software locks the computer, preventing the user from taking any further action. If the computer is connected to the internet, the client module notifies the administrator through the web application. The user is then provided a challenge that, when passed, unlocks the computer. If no connection is available, the computer remains locked until an administrator unlocks it.

The server provides the BioKey database and provides full CRUD operations through a ReSTful API. Furthermore, when appropriate, the server handles admin alerts through Twilio and client messages through the AWS SQS endpoints.

The web application provides a central dashboard for administrators to monitor their organization's user activity. Administrators can remotely lock and unlock computers in their organization. Furthermore, non-administrator users can log in to the web application to view their profile and recent activity. The web application is enabled by an API provided by a backend web server.

7.3 Definitions, Acronyms, and Abbreviations

Term	Meaning (if acronym)	Definition
API	Application Programming Interface	A set of defined methods of communication between software components.[33]
AWS	Amazon Web Services	The set of cloud services offered by Amazon.
POJO	Plain Old Java Object	A Java object that does not require any special classpath.[34]
SDS	Software Design Specification	This document.
SQS	Simple Queue Service	A server implementation of a queue offered by AWS.

[33] https://en.wikipedia.org/wiki/Application_programming_interface
[34] https://en.wikipedia.org/wiki/Plain_old_Java_object

SRS	Software Requirements Specification	The document which lays out functional and non-functional requirements of the system.

7.4 References

[1] SRS: Please refer to the submitted Software Requirement Specification document.

7.5 Overview of Document

The purpose of this document is to provide the reader with clear architectural, logical, and structural design specifications for BioKey. First, a high-level view of BioKey's architecture is presented. Then, each software package in the client module will be described – the server and admin portal are omitted from this section as they do not contain software packages in the traditional sense. Then, the low-level class and route definitions of the client module, admin portal, and server will be presented in sequence. Finally, sample module interactions are provided via sequence diagrams, followed by BioKey's design rationale.

8. SDS - Logical Architecture

8.1 Overview

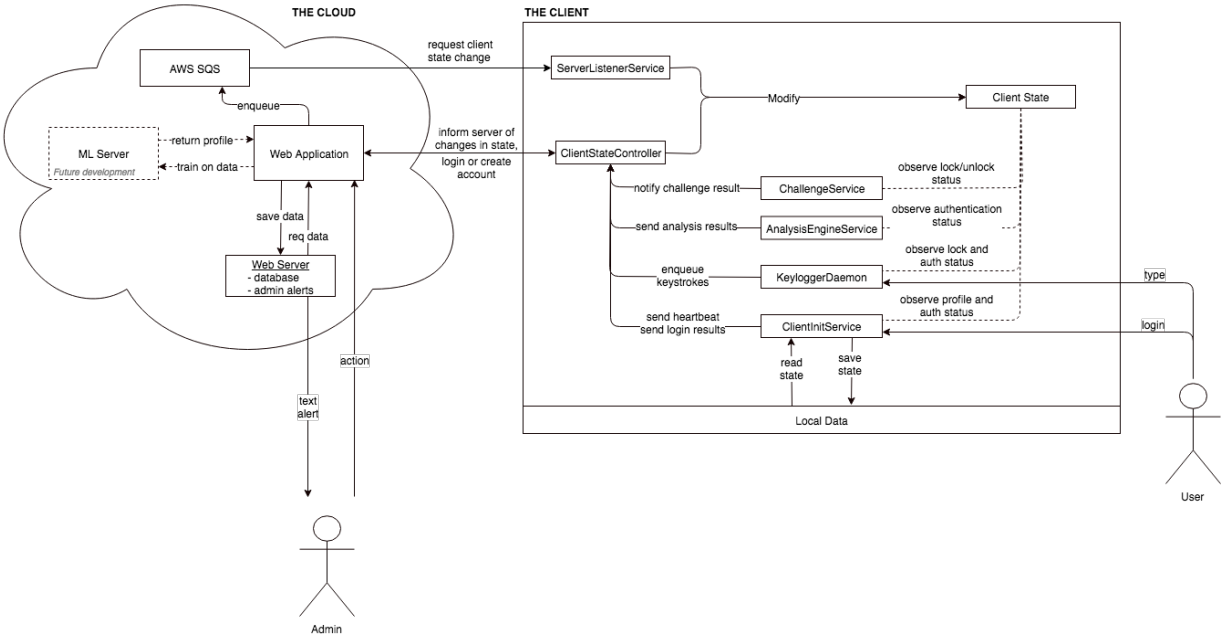


Figure 10: Architecture diagram

8.2 Packages

The following section provides a high-level overview of all software packages that comprise BioKey. Package diagrams may be found in *Section 3: Detailed Description of Components*.

8.2.1 Client Module

The client module includes all software packages that run locally on user machines. Three packages contain the important functionality for BioKey's client module: Models, Controllers, and Services. The details of the Helpers, Views, Providers, and Constants packages can be found in the full Javadocs at https://biokey.github.io/biokey-client/. Dependence upon these modules will be shown in Section 3 where necessary.

8.2.1.1 Models

The Models package defines the ClientStateModel class, which is used as the single source of truth for the client state. This includes the current typing profile, the user's authentication status, the machine's lock status, the session access token, and deques for analysis results and keystrokes.

The Models package also defines two sub-packages:

1. ***POJOs***. This package defines the Java objects to hold the various attributes of the client state.
2. ***Responses***. This package defines helper classes to handle server HTTP responses.

8.2.1.2 Services

The Services package defines five decoupled services running in BioKey:

1. The **ChallengeService**. This service listens for changes in the client state and issue challenges and locks the machine accordingly. It then passes challenge results to the controller.
2. The **AnalysisEngineService**. This service houses the machine learning model that performs the continuous authentication. This service listens for enqueued keystrokes and performs the appropriate analysis. It then passes analysis results to the controller.
3. The **KeyLoggerDaemon**. This service listens for typed keystrokes and passes them to the controller for enqueuing.
4. The **ClientInitService**. This service has three main responsibilities:
    a. *Login*. Upon launch, the ClientInitService instantiates all other services and presents the login page.
    b. *State saves/retrieval*. This service handles all writes and reads to and from local memory.
    c. *Maintaining the heartbeat*. This service is responsible for maintaining the client's online status through periodic heartbeat messages.
5. The **ServerListenerService**. This service is responsible for listening to the AWS SQS server assigned to the client and making any necessary state change requests to the Controllers package.

8.2.1.3 Controllers

The Controllers package defines the ClientStateController class, which has two main responsibilities:

1. *Forwarding client messages*. This class forwards any messages received from the services package to the web server for further processing.
2. *Maintaining the client state*. This class is responsible for making changes to client state according to the messages it receives from the Services package.

This package also defines the ***Challenges*** sub-package, contains the logic for the various re-authentication strategies.

8.2.1.4 Helpers

The Helpers package defines several helper classes for various tasks such as locking the machine, casting to POJOs, and building server requests.

8.2.1.5 Providers

The Provider package provides a series of classes that implement the inversion of control offered by the Spring Framework.

8.2.1.6 Views

The Views package defines the GUI components of the BioKey client module. This includes the lock panel, the challenge panels, the login panel, as well as the debugging view in the system tray.

8.2.1.7 Constants

The Constants package defines all constant values and enumerations used throughout the BioKey client module.

8.2.2 Server

As the server does not contain packages in the traditional sense, please refer to Section 4 for the routes and database schemas provided.

8.2.3 Admin Portal

As the server does not contain packages in the traditional sense, please refer to Section 11 for the routes and interfaces provided.

9. SDS - BioKey Client

9.1 Client State Classes

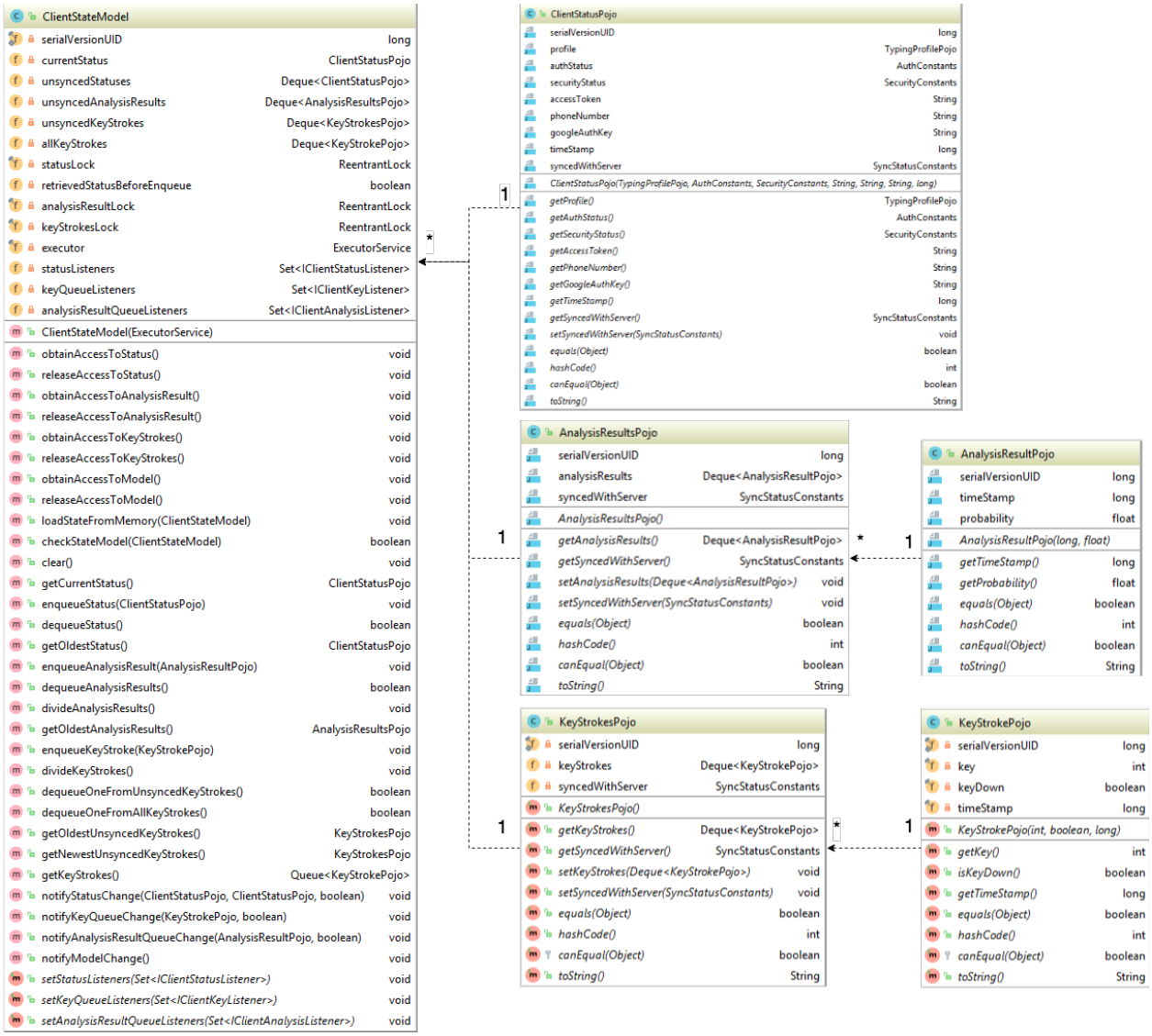


Figure 11: Class diagram - ClientStateModel

9.1.1 ClientStateModel

The ClientStateModel class provides the single source of truth for the client state, which the Services package subscribes to. This class is modified by the ServerListenerService. Furthermore, it uses POJO classes to store client state information. Finally, this class maintains sets of IClientStatusListeners, IClientKeyListeners, and IClientAnalysisListeners. These are interfaces that services implement to subscribe to changes in the state and are defined in the ClientStateModelClass.

9.1.1.1 ClientStateModel Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
currentStatus	ClientStatusPojo	The current status of the user's machine.
unsyncedStatuses	Deque<ClientStatusPojo>	The queue of status changes to be sent to the server.
unsyncedAnalysisResults	Deque<AnalysisResultsPojo>	The queue of analysis results to be sent to the server.
unsyncedKeyStrokes	Deque<KeyStrokesPojo>	The queue of keystrokes to be sent to the server.
allKeyStrokes	Deque<KeyStrokesPojo>	All keystrokes submitted by the user.
statusLock	ReentrantLock	Lock required to access currentStatus.
retrievedStatusBeforeEnqueue	boolean	Flag to check whether the most recent state has been accessed before modification.
analysisResultLock	ReentrantLock	Lock required to access analysis results.
keyStrokesLock	ReentrantLock	Lock required to access keystrokes.
executor	ExecutorService	Handles notification threads.
statusListeners	Set<IClientStatusListener>	The subscribers to status changes.
keyQueueListeners	Set<IClientKeyListener>	The subscribers to the keystroke deque.
analysisResultQueueListeners	Set<IClientAnalysisListener>	The subscribers to the analysis result deque.

9.1.1.2 ClientStateModel Operations

Operation	Description
ClientStateModel	Constructor.
obtainAccessToStatus	Obtain the statusLock.
releaseAccessToStatus	Release the statusLock.
obtainAccessToAnalysisResult	Obtain the analysisResultLock.
releaseAccessToAnalysisResult	Release the analysisResultLock.
obtainAccessToKeyStrokes	Obtain the keyStrokesLock.
releaseAccessToKeyStrokes	Release the keyStrokesLock.
obtainAccessToModel	Obtain all ReentrantLocks.
releaseAccessToModel	Release all ReentrantLocks.
loadStateFromMemory	Load the current state from local memory.
checkStateModel	Ensures all state fields are non-null.
clear	Clear all data from the model.
getCurrentStatus	Returns the current status of the user's machine.
enqueueStatus	Add a new (immutable) status to unsynced queue.
dequeueStatus	Dequeue the oldest status from the unsynced queue.
getOldestStatus	Peek at the oldest status from the unsynced queue.
enqueueAnalysisResult	Add a new (immutable) analysis result to unsynced queue.
dequeueAnalysisResults	Dequeue the oldest analysis results from the unsynced queue.

divideAnalysisResults	Divides the unsynced analysis results queue and bundles all the analysis results since last division together. dequeueAnalysisResults() will dequeue the last bundle.
getOldestAnalysisResults	Peek at the oldest analysis results from the unsynced queue.
enqueueKeyStroke	Add a new (immutable) key stroke to unsynced and all key strokes queues.
divideKeyStrokes	Divides the unsynced key strokes queue and bundles all the key strokes since last division together. dequeueOneFromUnsyncedKeyStrokes() will dequeue the last bundle.
dequeueOneFromUnsyncedKeyStrokes	Dequeue the oldest window of key strokes from the unsynced queue.
dequeueOneFromAllKeyStrokes	Dequeue the oldest key stroke from the all key strokes queue.
getOldestUnsyncedKeyStrokes	Peek at the oldest window of key strokes from the unsynced queue.
getNewestUnsyncedKeyStrokes	Peek at the newest window of key strokes from the unsynced queue.
getKeyStrokes	Get all the keystrokes in client memory.
notifyStatusChange	Notifies all the listeners of a status change.
notifyKeyQueueChange	Notifies all the listeners of a key queue change.
notifyAnalysisResultQueueChange	Notifies all the listeners of an analysis result queue change.
notifyModelChange	Notifies all the listeners of all the queues that the model has changed.
setStatusListeners	Set the subscribers to status changes.
setKeyQueueListeners	Set the subscribers to the keystroke deque.
setAnalysisResultQueueListeners	Set the subscribers to the analysis result deque.

9.1.1.3 ClientStateModel Design Specification/Constraints

- ClientStateModel is designed to be the model entity in the MVC architecture
- ClientStateModel is the observable entity in the Observer design pattern

9.1.2 ClientStatusPojo

The ClientStatusPojo class is used to store the client state as a plain Java object. This class uses the Constants package to define state values.

9.1.2.1 ClientStatusPojo Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
profile	TypingProfilePojo	A POJO of the user's typing profile.
authStatus	AuthConstants	The authentication status of the profile.
securityStatus	SecurityConstants	The lock/unlock status of the profile.
accessToken	String	Used to maintain the server session.
phoneNumber	String	The user's phone number.
googleAuthKey	String	The token used for Google Authentication.
timeStamp	long	The timestamp of the last modification.

syncedWithServer	SyncStatusConstants	The status of client-server agreement.

9.1.2.2 ClientStatusPojo Operations

Operation	Description
ClientStatusPojo	Constructor.
getProfile	Get the user's typing profile.
getAuthStatus	Get the authentication status of the profile.
getSecurityStatus	Get the lock/unlock status of the profile.
getAccessToken	Get the token used to maintain the server session.
getPhoneNumber	Get the user's phone number.
getGoogleAuthKey	Get the token used for Google Authentication.
getTimeStamp	Get the timestamp of the last modification.
getSyncedWithServer	Get the status of client-server agreement.
setSyncedWithServer	Set the status of client-server agreement.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.
toString	Converts the POJO to String format.

9.1.2.3 ClientStatusPojo Design Specification/Constraints

- As a simple data storage class, this class follows design of the server-defined typing profile entity. This design is given in *Section 15: Implementation* of the BioKey Final Report.
- As a simple data storage class, there are no meaningful constraints to its design.

9.1.3 AnalysisResultsPojo

The AnalysisResultsPojo class is container class for AnalysisResultPojo objects. The class provides methods to conveniently enqueue unsynced analysis results. This class uses the Constants package to define state values.

9.1.3.1 AnalysisResultsPojo Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
analysisResults	Deque<AnalysisResultPojo>	The deque of current analysis results.
syncedWithServer	SyncStatusConstants	The state of client-server agreement.

9.1.3.2 AnalysisResultsPojo Operations

Operation	Description
AnalysisResultsPojo	Constructor.
getAnalysisResults	Get the deque of current analysis results.
getSyncedWithServer	Get the state of client-server agreement.
setAnalysisResults	Set the deque of current analysis results.
setSyncedWithServer	Set the state of client-server agreement.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok. Not used.
toString	Converts the POJO to String format.

9.1.3.3 AnalysisResultsPojo Design Specification/Constraints

- As a simple container class, there are no meaningful specifications or constraints to its design.

9.1.4 AnalysisResultPojo

The AnalysisResultPojo class is used to store analysis results as plain Java objects.

9.1.4.1 AnalysisResultPojo Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
timeStamp	long	The timestamp of the last modification.
probability	float	The percentage confidence in the user.

9.1.4.2 AnalysisResultPojo Operations

Operation	Description
AnalysisResultPojo	Constructor.
getTimeStamp	Get the timestamp of the last modification.
getProbability	Get the percentage confidence in the user.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.
toString	Converts the POJO to String format.

9.1.4.3 AnalysisResultPojo Design Specification/Constraints

- As a simple data storage class, this class follows the analysis output format of the AnalysisEngineService. More detail on keystroke analysis is given in *Section 15: Implementation* of the BioKey Final Report.
- As a simple data storage class, there are no meaningful constraints to its design.

9.1.5 KeyStrokesPojo

The KeyStrokesPojo class is a container class for KeyStrokePojo objects. The class provides methods to conveniently enqueue unsynced keystrokes. This class uses the Constants package to define state values.

9.1.5.1 KeyStrokesPojo Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
keyStrokes	Deque<KeyStrokePojo>	The deque of keystrokes.
syncedWithServer	SyncStatusConstants	The state of client-server agreement.

9.1.5.2 KeyStrokesPojo Operations

Operation	Description
KeyStrokesPojo	Constructor.
getKeyStrokes	Get the deque of keystrokes.
getSyncedWithServer	Get the state of client-server agreement.

setKeyStrokes	Set the deque of keystrokes.
setSyncedWithServer	Set the state of client-server agreement.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.
toString	Converts the POJO to String format.

9.1.5.3 KeyStrokesPojo Design Specification/Constraints

- As a simple container class, there are no meaningful specifications or constraints to its design.

9.1.6 KeyStrokePojo

The KeyStrokePojo class is used to store keystrokes as plain Java objects.

9.1.6.1 KeyStrokePojo Attributes

Attribute	Type	Description
serialVersionUID	long	ID used for serialization.
key	int	The key that was submitted.
keyDown	boolean	A flag for whether the key was pressed or released.
timeStamp	long	The timestamp of the last modification.

9.1.6.2 KeyStrokePojo Operations

Operation	Description
KeyStrokePojo	Constructor.
getKey	Get the key that was submitted.
isKeyDown	Get the keyDown flag.
getTimeStamp	Get the timestamp of the last modification.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.
toString	Converts the POJO to String format.

9.1.6.3 KeyStrokePojo Design Specification/Constraints

- As a simple data storage class, this class follows the analysis input format of the AnalysisEngineService. More detail on keystroke analysis is given in *Section 15: Implementation* of the BioKey Final Report.
- As a simple data storage class, there are no meaningful constraints to its design.

9.2 ClientInitService

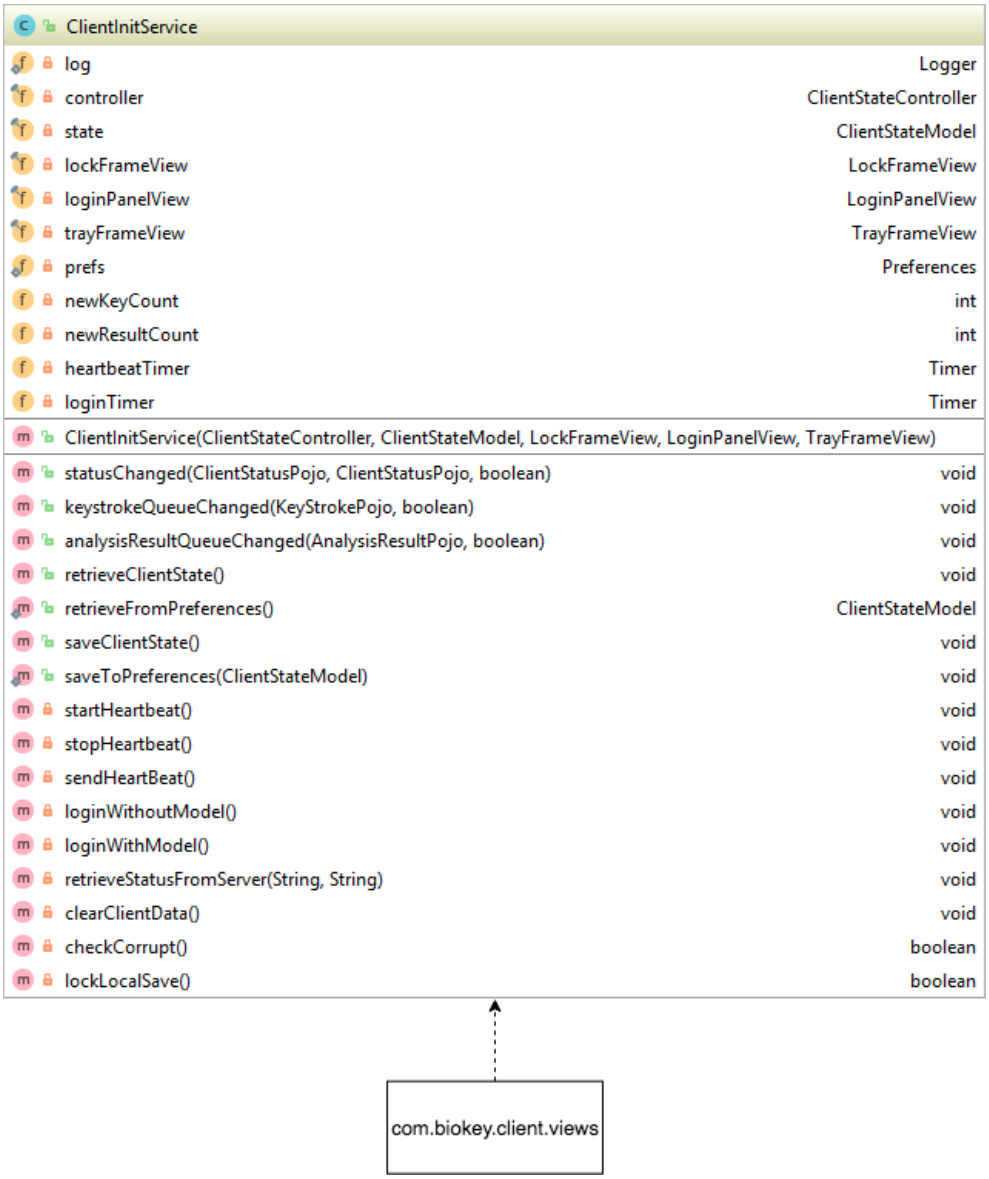


Figure 12: Class diagram - ClientInitService

9.2.1 ClientInitService

The ClientInitService is defined in the Services package. A high-level overview is provided in Section 2.2.1.2.

9.2.1.1 ClientInitService Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
controller	ClientStateController	A reference to the application's controller.
state	ClientStateModel	A reference to the application's current state.

lockFrameView	LockFrameView	The view shown when the machine is locked.
loginPanelView	LoginPanelView	The view shown while the user is logging in.
trayFrameView	TrayFrameView	The view provided to show the system tray interface.
prefs	Preferences	A reference to the local Preferences.
newKeyCount	int	A count of the new keystrokes.
newResultCount	int	A count of the new analysis results.
heartbeatTimer	Timer	Used to time heartbeat messages.
loginTimer	Timer	Used to schedule login attempts.

9.2.1.2 ClientInitService Operations

Operation	Description
ClientInitService	Constructor.
statusChanged	Implementation of listener to the ClientStateModel's status. The service will save the client state periodically.
keystrokeQueueChanged	Implementation of listener to the ClientStateModel's keystroke queues. The service will save the client state periodically.
analysisResultQueueChanged	Implementation of listener to the ClientStateModel's analysis results queue. The service will save the client state periodically.
retrieveClientState	Entry point into this service. Tries to retrieve the client state and in the process calls the checkCorrupt() and at least one of the login functions.
retrieveFromPreferences	Retrieve model from preferences.
saveClientState	Called when the status of the client changes. Saves the entire state to Preferences.
saveToPreferences	Save model to preferences.
startHeartbeat	Starts the controller's heartbeat function.
stopHeartbeat	Stops the controller's heartbeat function.
sendHeartbeat	Sends a heartbeat using the controller's interface.
loginWithoutModel	Login without any local credentials. Will prompt the user for the credentials.
loginWithModel	Login with local credentials (auth token). Will call the no parameter version if the credentials are not validated by server.
retrieveStatusFromServer	Retrieve status from server given the authToken and computer's MAC address.
clearClientData	Clear both the model and Preferences.
checkCorrupt	Check the local files and OS for signs of corruption.
lockLocalSave	Lock the local files securely.

9.2.1.3 ClientInitService Design Specification/Constraints

- Saving the profile, saving the access token, and making login requests are be done with an acceptable degree of security
- This class serves as the entry point to other classes in the software
- This class is an observer in the Observer design pattern that BioKey implements

9.3 KeyLoggerDaemonService

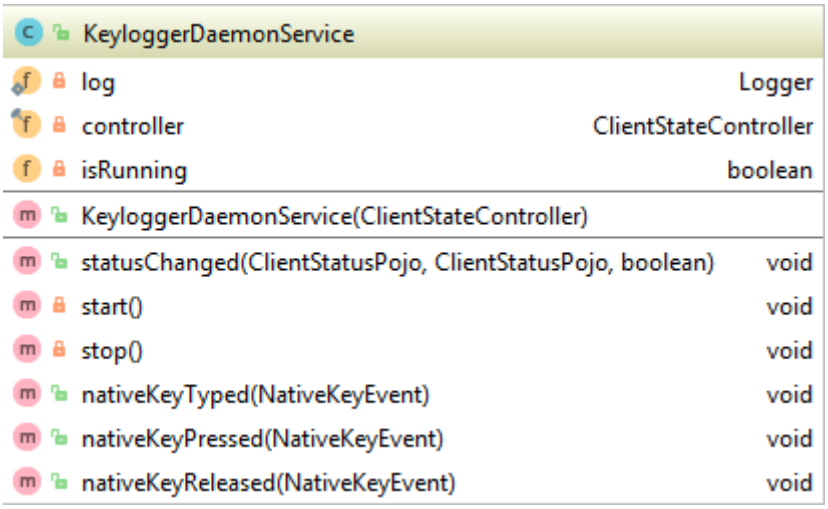


Figure 13: Class diagram - KeyLoggerDaemonService

9.3.1 KeyloggerDaemonService

The KeyloggerDaemonService is defined in the Services package. A high-level overview is provided in Section 2.2.1.2.

9.3.1.1 KeyloggerDaemonService Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
controller	ClientStateController	A reference to the application's controller.
isRunning	boolean	A flag that reflects whether the daemon is running.

9.3.1.2 KeyloggerDaemonService Operations

Operation	Description
KeyLoggerDaemonService	Constructor.
statusChanged	Implementation of listener to the ClientStateModel's status. The status contains a flag for whether the daemon should be running.
start	Start running the daemon.
stop	Stop running the daemon.
nativeKeyTyped	Unused.
nativeKeyPressed	Enqueue the "down" part of a keystroke.
nativeKeyReleased	Enqueue the "up" part of a keystroke.

9.3.1.3 KeyloggerDaemonService Design Specification/Constraints

- Keystrokes should be sent in a completely secure manner
- Pausing the keystroke collection should not be executable by the user
- This class is an observer in the Observer design pattern that BioKey implements

9.4 AnalysisEngineService Classes

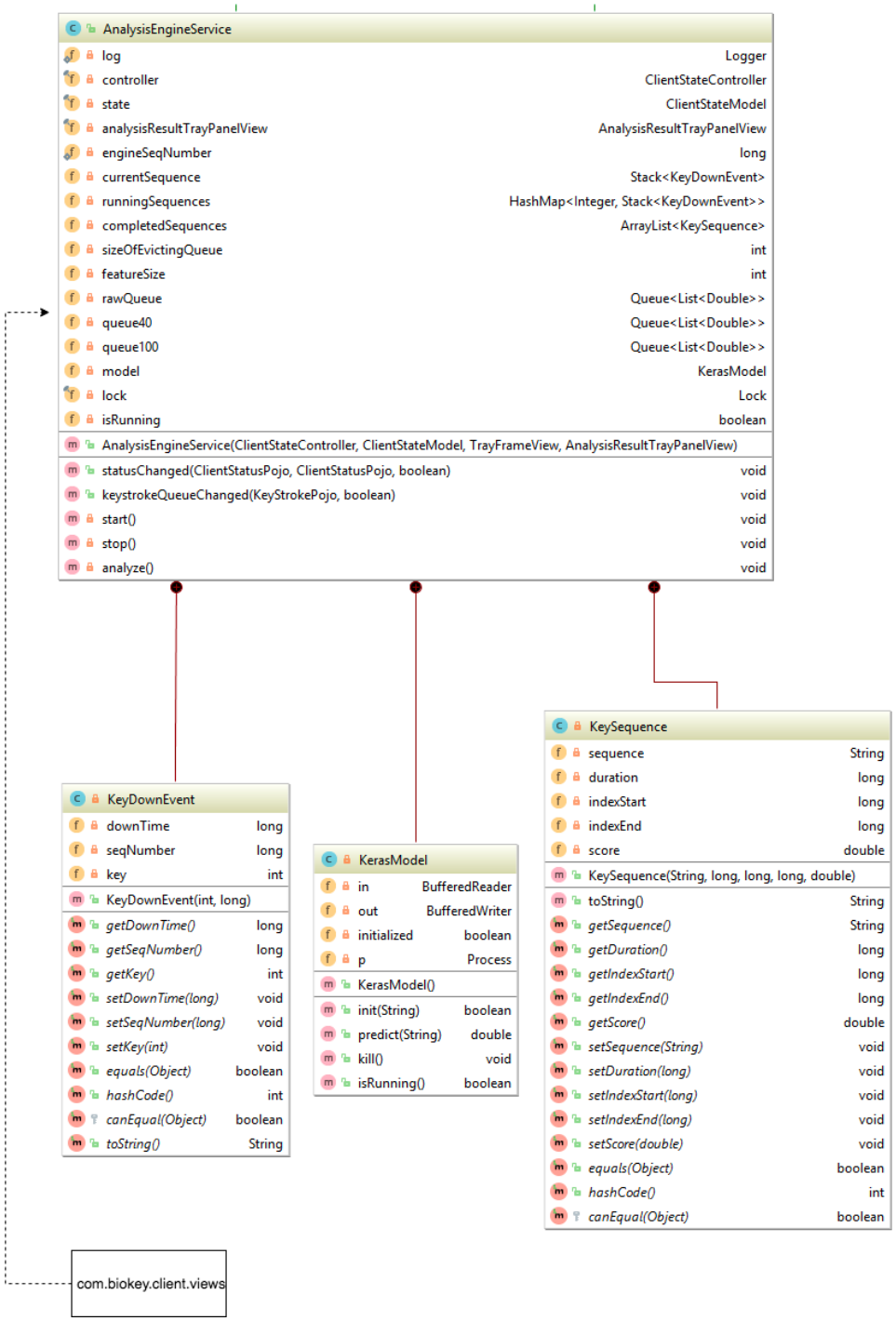


Figure 14: Class diagram - AnalysisEngineService

9.4.1 AnalysisEngineService

The AnalysisEngineService is defined in the Services package. A high-level overview is provided in Section 2.2.1.2.

9.4.1.1 AnalysisEngineService Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
controller	ClientStateController	A reference to the application's controller.
state	ClientStateModel	A reference to the application's current state.
analysisResultTrayPanelView	AnalysisResultTrayPanelView	The view for the analysis tray panel.
engineSeqNumber	long	Keeps track of the current order
currentSequence	Stack<KeyDownEvent>	A reference to the longest current running sequence, in the form of a LIFO stack.
runningSequences	HashMap<Integer,Stack<Key DownEvent>	A map of all of the sequences that a user is currently building.
completedSequences	ArrayList<KeySequence>	A list of all sequences that the user has completed.
sizeOfEvictingQueue	int	The size of the evicting queues.
featureSize	int	The number of features in the user's Gaussian profile.
rawQueue	Queue<List<Double>>	An evicting queue holding the last 100 raw analysis inputs.
queue40	Queue<List<Double>>	An evicting queue holding the last 100 40 key lookback analysis inputs.
queue100	Queue<List<Double>>	An evicting queue holding the last 100 100 key lookback analysis inputs.
model	KerasModel	The neural net model for the user represented as a Keras model.
lock	Lock	A lock to control access to the Queues and prevent concurrency issues.
isRunning	boolean	Keeps track of whether the analysis engine is running.

9.4.1.2 AnalysisEngineService Operations

Operation	Description
AnalysisEngineService	Constructor.
statusChanged	Implementation of listener to the ClientStateModel's status. The status will contain the details on the analysis model to run through the typing profile.
keystrokeQueueChanged	Implementation of listener to the ClientStateModel's keystroke queues. New keys need to be fed into the model.
start	Start running the analysis engine.
stop	Stop running the analysis engine.
analyze	Use the current keystroke data to generate the likelihood that the user's typing matches their current profile.

9.4.1.3 AnalysisEngineService Design Specification/Constraints

- Analysis results should be sent in an acceptably secure manner
- Locking the OS should not be executable by the user
- This class is an observer in the Observer design pattern that BioKey implements

9.4.2 KeyDownEvent

A KeyDownEvent stores the information associated with every key that a user presses down. They are created when NativeKeyDownEvents are raised.

9.4.2.1 KeyDownEvent Attributes

Attribute	Type	Description
downTime	long	The time the key was pressed down.
seqNumber	long	The engine sequence number when the key was pressed down.
key	int	An integer representation of which key was pressed down.

9.4.2.2 KeyDownEvent Operations

Operation	Description
KeyDownEvent	Constructor.
getDownTime	Get the down time.
getSeqNumber	Get the sequence numbers.
getKey	Get the key that was pressed.
setDownTime	Set the down time.
setSeqNumber	Set the sequence numbers.
setKey	Set the key that was pressed.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.
toString	Converts the object to String format.

9.4.2.3 KeyDownEvent Design Specification/Constraints

- As a simple data storage class, there are no meaningful constraints to its design.

9.4.3 KerasModel

The KerasModel holds a representation of the neural net that is used to make predictions about whether a current user is an imposter or not.

9.4.3.1 KerasModel Attributes

Attribute	Type	Description
int	BufferedReader	The reader used to receive information to the model.
out	BufferedWriter	The writer used to send information to the model.
initialized	boolean	Whether the model is initialized or not.

p	Process	A process representing the running model.

9.4.3.2 KerasModel Operations

Operation	Description
KerasModel	Constructor.
init	Initializes the model.
predict	Run a prediction given the current keys typed.
kill	Kills the model.
isRunning	Determines whether the model is running or not.

9.4.3.3 KerasModel Design Specification/Constraints

- This class follows the design specification of our machine learning model. More detail on keystroke analysis is given in *Section 15: Implementation* of the BioKey Final Report.

9.4.4 KeySequence

A KeySequence is a single key or series of keys that have been typed consecutively within a threshold. The scores in a KeySequence are the main inputs to the Neural Net.

9.4.4.1 KeySequence Attributes

Attribute	Type	Description
sequence	String	A string representation of the sequence (e.g. A-R).
duration	long	The length of the sequence (ms).
indexStart	long	The engine sequence number associated with the start of the sequence.
indexEnd	long	The engine sequence number associated with the end of the sequence.
score	double	The score the sequence received when compared to the user's Gaussian profile.

9.4.4.2 KeySequence Operations

Operation	Description
KeySequence	Constructor.
toString	Returns a string representation of the sequence.
getSequence	Returns the sequence.
geDuration	Returns the duration of the sequence.
getIndexStart	Returns the engine sequence number associated with the start of the sequence.
getIndexEnd	Returns the engine sequence number associated with the end of the sequence.

getScore	Returns the score the sequence received when compared to the user's Gaussian profile.
setSequence	Sets the sequence.
setDuration	Sets the duration of the sequence.
setIndexStart	Sets the engine sequence number associated with the start of the sequence.
setIndexEnd	Sets the engine sequence number associated with the end of the sequence.
setScore	Sets the score the sequence received when compared to the user's Gaussian profile.
equals	Checks object equality.
hashCode	Calculate the hash of the POJO.
canEqual	Method generated by Lombok.

9.4.4.3 KeySequence Design Specification/Constraints

- This class follows the design specification of our machine learning model. More detail on keystroke analysis is given in *Section 15: Implementation* of the BioKey Final Report.

9.5 ChallengeService

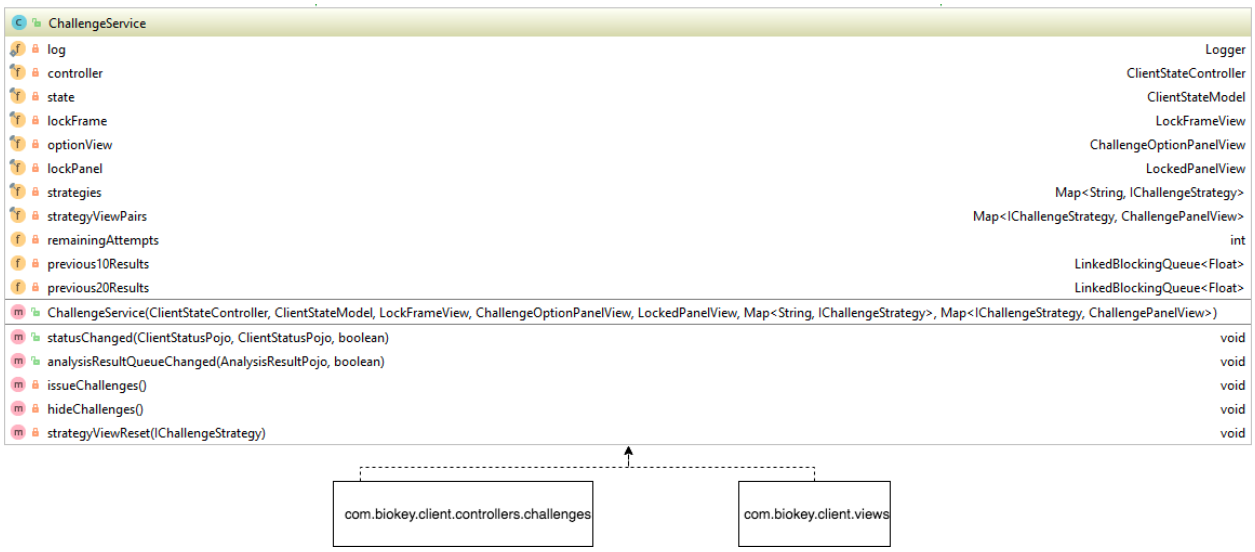


Figure 15: Class diagram - ChallengeService

9.5.1 ChallengeService

The ClientInitService is defined in the Services package. A high-level overview is provided in Section 2.2.1.2.

9.5.1.1 ChallengeService Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
controller	ClientStateController	A reference to the application's controller.
state	ClientStateModel	A reference to the application's current state.
lockFrame	LockFrameView	The view shown when the machine is locked.
optionView	ChallengeOptionPanelView	The view shown when choosing a re-authentication method.
lockPanel	LockedPanelView	The panel that is added to the lockFrame.
strategies	Map<String, IChallengeStrategy>	The list of configured challenge strategies.
strategyViewPairs	Map<IChallengeStrategy, ChallengePanelView>	The panels corresponding to each configured challenge strategy.
remainingAttempts	int	The number of attempts remaining.
previous10Results	LinkedBlockingQueue<Float>	The previous 10 analysis results.
previous20Results	LinkedBlockingQueue<Float>	The previous 20 analysis results.

9.5.1.2 ChallengeService Operations

Operation	Description
ChallengeService	Constructor.
statusChanged	Implementation of listener to the ClientStateModel's status. The status will contain the accepted strategies to challenge the user and a flag for whether the locker should lock or unlock.
analysisResultQueueChanged	Implementation of listener to the ClientStateModel's analysis results queue. The queue contains updates to the client's authentication, and the locker will decide whether a state change is necessary.
issueChallenges	Issues the challenges based on accepted strategies and notify client state of the result.
hideChallenges	Hides the challenges.
strategyViewReset	Reset the view associated with a strategy.

9.5.1.3 ChallengeService Design Specification/Constraints

- Challenges are issued based on a configurable priority list
- Locking the OS, issuing a challenge, and unlocking the OS are performed in a seamless manner in order to preserve user experience
- This class listens to changes in the client state
- This class is an observer in the Observer design pattern that BioKey implements

9.6 ClientStateController

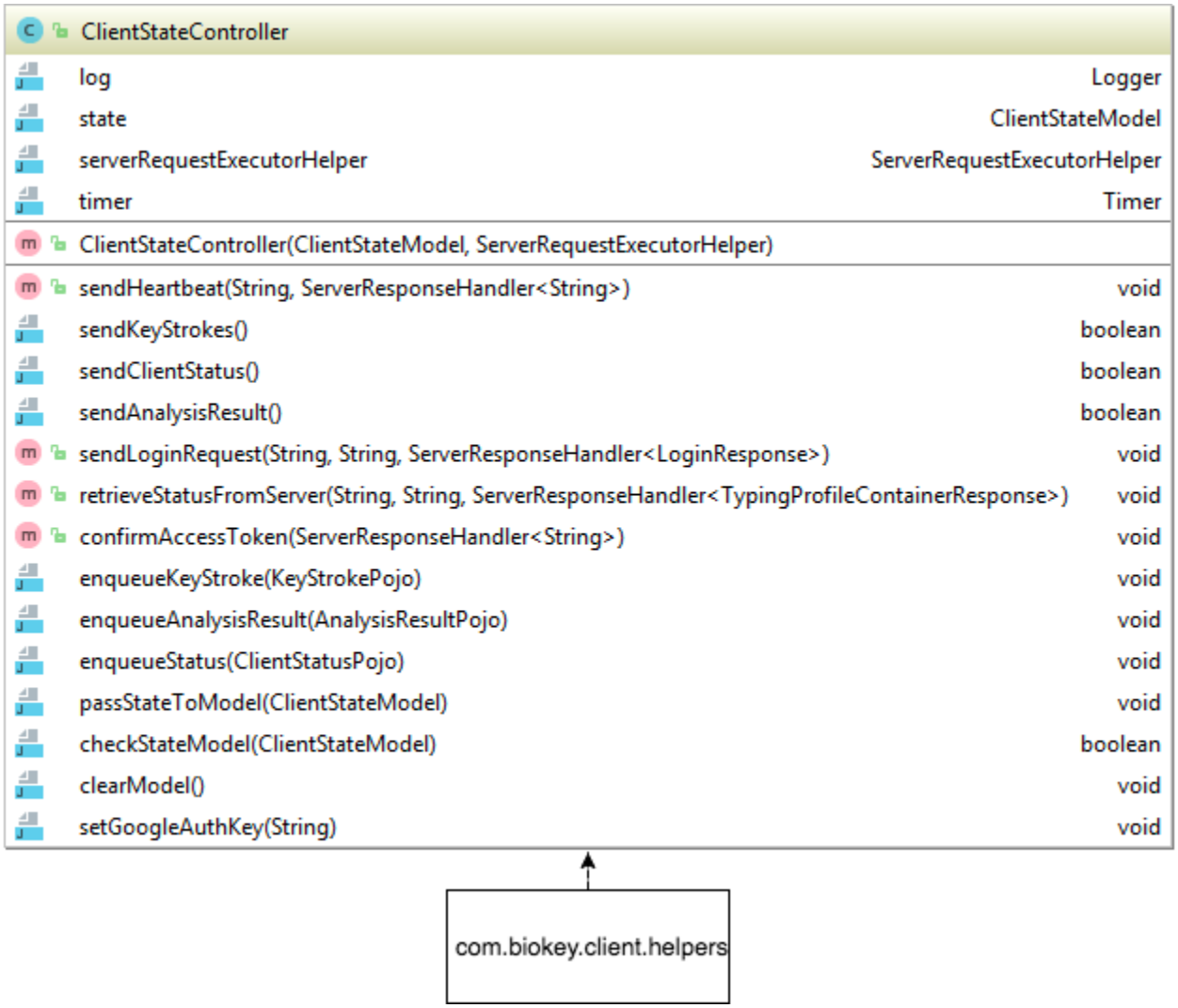


Figure 16: Class diagram - ClientStateController

9.6.1 ClientStateController

The ClientStateController is defined in the Controllers package. A high-level overview is provided in Section 2.2.1.3.

9.6.1.1 ClientStateController Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
state	ClientStateModel	A reference to the application's current state.
serverRequestExecutorHelper	ServerRequestExecutorHelper	Helps make requests to the server.
timer	Timer	Used to schedule update and heartbeat requests.

9.6.1.2 ClientStateController Operations

Operation	Description
ClientStateController	Constructor.
sendHeartbeat	Sends the server a message at fixed time intervals to let it know when the client is alive.
sendKeyStrokes	Sends the server a message at fixed time intervals to let it know when the client is alive.
sendClientStatus	Sends the server a request with the oldest status not yet known to server. Modifies the sync status for status sent.
sendAnalysisResult	Sends the server a request with the oldest analysis result not yet known to server. Modifies the sync status for result sent.
sendLoginRequest	Send a login request to the server.
retrieveStatusFromServer	Send a GET request to retrieve a user's typing profile.
confirmAccessToken	Sends the server a request with the access token to confirm the client is still authenticated.
enqueueKeyStroke	Modify the model to include the new keystroke.
enqueueAnalysisResult	Modify the model to include the new analysis result.
enqueueStatus	Modify the model to include the new status.
passStateToModel	Passes the client state read from memory to the model.
checkStateModel	Checks if the client state model read from memory is valid.
clearModel	Tell the model to clear itself.
setGoogleAuthKey	Modify the model with the new Google Auth key.

9.6.1.3 ClientStateController Design Specification/Considerations

- All communications with the server should be sent in a completely secure manner
- Changing the server status endpoint should not be executable by the user

9.7 ServerListenerService Classes

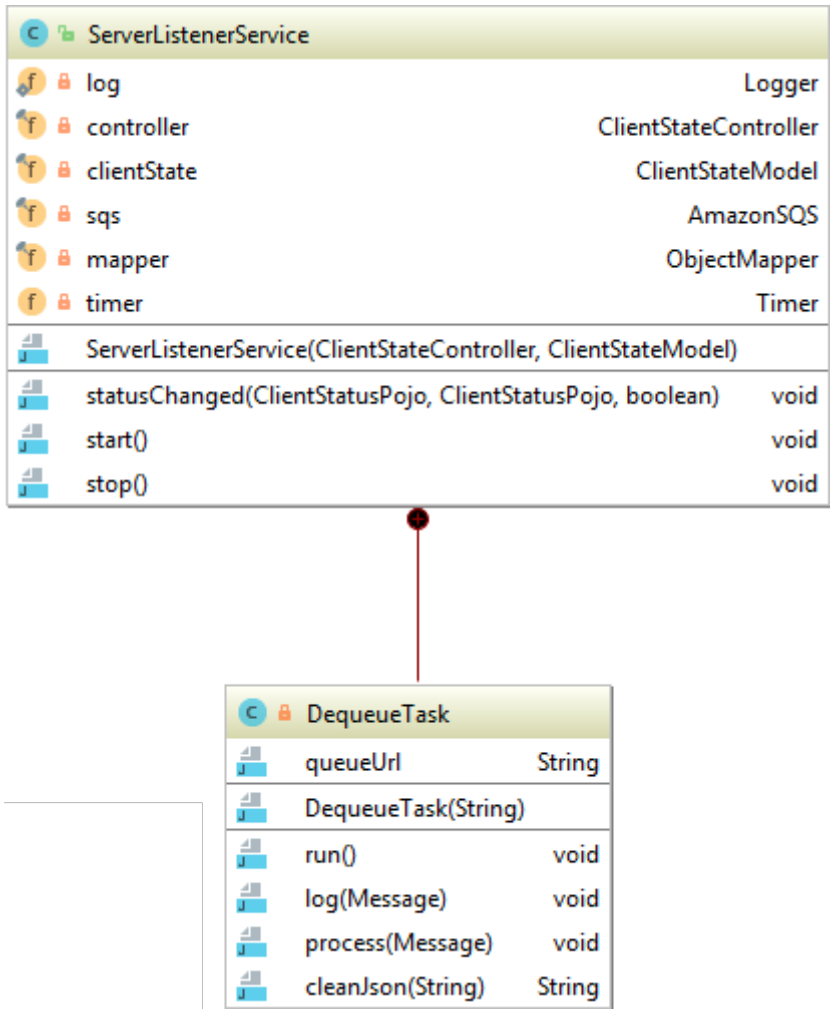


Figure 17: Class diagram - ServerListenerService

9.7.1 ServerListenerService

The ServerListenerService is defined in the Services package. A high-level overview is provided in Section 2.2.1.2.

9.7.1.1 ServerListenerService Attributes

Attribute	Type	Description
log	Logger	Provides application logging.
controller	ClientStateController	A reference to the application's controller.
clientState	ClientStateModel	A reference to the application's current state.
sqs	AmazonSQS	The AWS SQS endpoint to listen to.
mapper	ObjectMapper	Used to read JSON server messages.
timer	Timer	Used to time server polling.

9.7.1.2 ServerListenerService Operations

Operation	Description
ServerListenerService	Constructor.
statusChanged	Implementation of listener to the ClientStateModel's status. The status will contain a flag for whether the service should be running.
start	Start running the server listener.
stop	Stop running the server listener.

9.7.1.3 ServerListenerService Design Specifications/Considerations

- All communications with the server should be sent in a completely secure manner
- Changing the server status endpoint should not be executable by the user
- This class listens to changes in the SQS server

9.7.2 DequeueTask

The DequeueTask class is an inner class, defined in ServerListenerService. It is used scheduled by the ServerListenerService to periodically check the AWS SQS endpoint and dequeue a task if one exists.

9.7.2.1 DequeueTask Attributes

Attribute	Type	Description
queueUrl	String	Used to time server polling.

9.7.2.2 DequeueTask Operations

Operation	Description
DequeueTask	Constructor.
run	The scheduled task. Dequeues a task from the SQS endpoint.
log	Prints debugging output to the console.
process	Upon receipt of a message, makes the required change to the state.
cleanJson	A helper function to parse JSON.

9.7.2.3 DequeueTask Design Specifications/Considerations

- Since this task is run frequently, it must take very few computational resources.

10. SDS - BioKey Server

10.1 Overview

This subsection will explain the design rationale for the BioKey server and the database schema. It includes an entity-relationship diagram as well as a description and definition of all the server routes.

10.2 Database Schema

Figure 18 shows the entity-relationship diagram with which we used to build the various database schemas in our application. Since we used MongoDB as our database, our database is a non-relational database. There were a number of reasons we chose to do this. First, we decided to use Express and Node extensively in our application which has extensive synergies with Mongo. Second, MongoDB can store generic data types and larger record sizes, which enables us to store model definitions and weights on the typing profiles. With that being said, a relational database schema can still give the user a good understanding of the relevant entities and the relationships that exist with them.

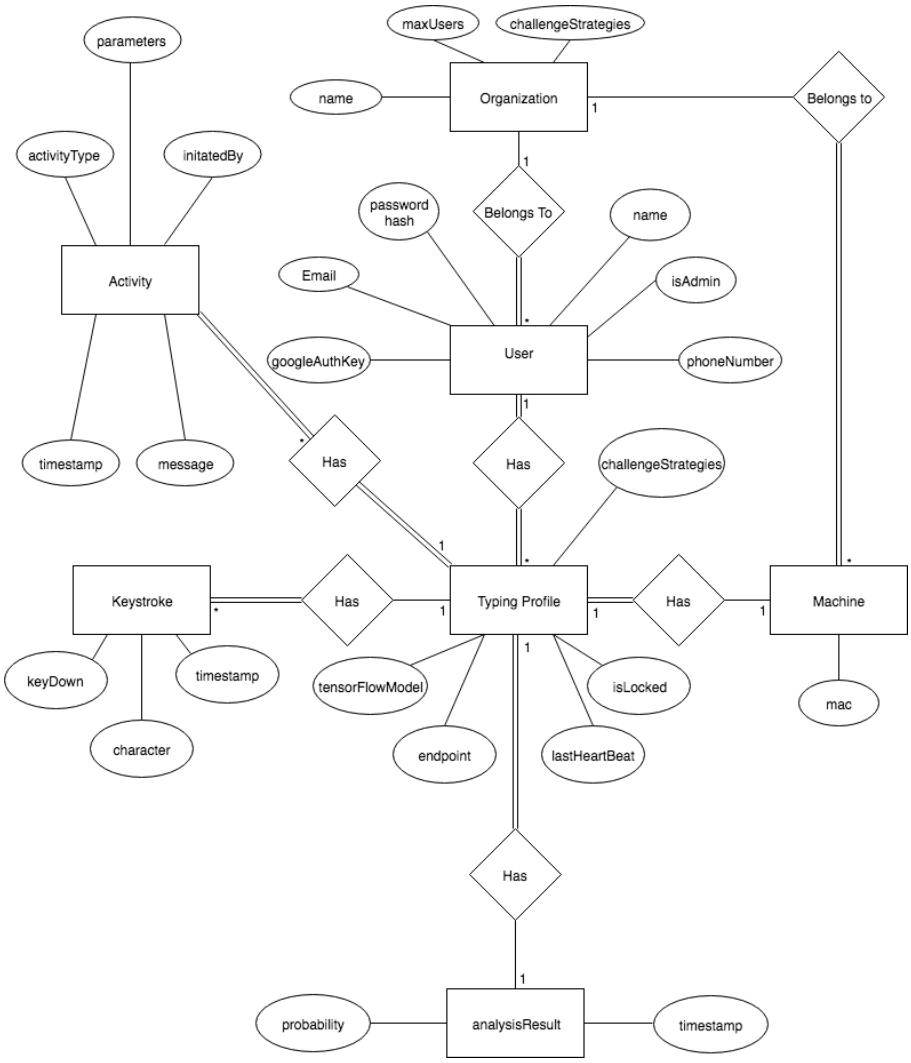


Figure 18: Entity-relationship diagram

10.3 Server Endpoints and Descriptions

10.3.1 Summary

The API-Docs IO framework was used to build the server documentation using. The fully formatted file can be found on our GitHub at: https://biokey.github.io/biokey-backend/. The routes can be segregated into eight sections are as follows:

10.3.2 Activities

Route: GET /api/activities
Name: ListActivities
Description: Get a list of all activities for the requesting user's organization. The requesting user will not be able to request activities outside of the organization. Can specify the limit, page, and sort for pagination.

Route: GET /api/activities/:id
Name: GetActivity
Description: Get the activity within the requesting user's organization given an id.

Route: POST /api/activities
Name: PostActivity
Description: Create a new activity within the requesting user's organization. If the user is an admin, they can create an activity associated with any typing profile within their organization. If the requesting user is not an admin, they can only create an activity for themselves.

Route: PUT /api/activities/:id
Name: UpdateActivity
Description: Update an existing activity within the requesting user's organization given an id. Not implemented because no user should need to update activities (they should be immutable).

Route: DELETE /api/activities/:id
Name: DeleteActivity
Description: Deletes an existing activity within the requesting user's organization given an id. Not implemented because no user should need to delete activities (for recordkeeping purposes).

10.3.3 Analysis Results

Route: GET /api/analysisResults
Name: ListAnalysisResults
Description: Get a list of all analysis results for the requesting user's organization. The requesting user will not be able to request results outside of the organization. Can specify the start and end timestamp to look within.

Route: POST /api/analysisResults
Name: PostAnalysisResult
Description: Create a new analysis result within the requesting user's organization. The requesting user can only create an analysis result for themselves.

Route: PUT /api/analysisResults/:id
Name: UpdateAnalysisResult
Description: Update an existing analysis result within the requesting user's organization given an id. Not implemented because no user should be able to change analysis results (they should be immutable).

Route: DELETE /api/analysisResults/:id
Name: DeleteAnalysisResult
Description: Delete an existing analysis result within the requesting user's organization given an id. Not implemented because no user should be able to delete analysis results (for recordkeeping purposes).

Route: GET /api/analysisResults/:id
Name: GetAnalysisResults
Description: Get the analysis results within the requesting user's organization given an id. Not implemented because no user should need to get individual analysis results (not useful by themselves

10.3.4 Keystrokes

Route: GET /api/keystrokes
Name: ListKeystrokes
Description: Get a list of all keystrokes for the requesting user's organization. The requesting user will not be able to request keystrokes outside of the organization. Not implemented for now because system currently blocks access from sensitive keystroke information.

Route: GET /api/keystrokes/:id
Name: GetKeystroke
Description: Get the keystrokes within the requesting user's organization given an id. Not implemented for now because system currently blocks access from sensitive keystroke information.

Route: POST /api/keystrokes
Name: PostKeystroke
Description: Create new keystrokes within the requesting user's organization. The requesting user can only create a keystrokes for themselves.

Route: PUT /api/keystrokes/:id
Name: UpdateKeystroke
Description: Update an existing keystroke within the requesting user's organization given an id. Not implemented for now because system currently blocks access from sensitive keystroke information.

Route: DELETE /api/keystrokes/:id
Name: DeleteKeystroke
Description: Delete an existing analysis result within the requesting user's organization given an id. Not implemented for now because system currently blocks access from sensitive keystroke information.

10.3.5 TypingProfiles

Route: GET /api/typingProfiles
Name: ListTypingProfiles
Description: Get a list of all typing profiles for the requesting user's organization. The requesting user will not be able to request profiles outside of the organization. Can specify the limit, page, and sort for pagination.

Route: GET /api/typingProfiles/:id
Name: GetTypingProfile
Description: Get the typing profile within the requesting user's organization given an id.

Route: POST /api/typingProfiles
Name: PostTypingProfile

Description: Create a new typing profile within the requesting user's organization. If the user is an admin, they can create a typing profile associated with any user within their organization. If the requesting user is not an admin, they can only create an typing profile for themselves. Also creates an activity that the user created a typing profile. Also alerts the admin through text message that the user created a typing profile.

Route: PUT /api/typingProfiles/:id
Name: UpdateTypingProfile
Description: Update an existing typing profile within the requesting user's organization given an id. If specified, can also update a user's phone number and google authentication key. Also creates an activity with the change. If the change was important (lock, unlock), alerts the admin through text message of the change.

Route: DELETE /api/typingProfiles/:id
Name: DeleteTypingProfile
Description: Deletes an existing typing profile within the requesting user's organization given an id.

Route: POST /api/typingProfiles/:id/heartbeat
Name: PostHeartbeat
Description: Update the typing profile with the latest heartbeat from the client

Route: POST /api/typingProfiles/machine/:mac
Name: PostMachineTypingProfile
Description: Create a new machine and/or typing profile if they do not exist. If they do, then get the typing profile given a user's token and machine mac.

10.3.6 Machines

Route: GET /api/machines
Name: ListMachines
Description: Get a list of all machines for the requesting user's organization. The requesting user will not be able to request machines outside of the organization. Can specify the limit, page, and sort for pagination.

Route: GET /api/machines/:id
Name: GetMachine
Description: Get the machine within the requesting user's organization given an id.

Route: POST /api/machines
Name: PostMachine
Description: Create a new machine within the requesting user's organization.

Route: PUT /api/machines/:id
Name: UpdateMachine
Description: Update an existing analysis result within the requesting user's organization given an id.
Route: DELETE /api/machines/:id
Name: DeleteMachine
Description: Delete an existing analysis result within the requesting user's organization given an id.

10.3.7 Organizations

Route: GET /api/organizations
Name: ListOrganizations

Description: Get the requesting user's organization as no user should have access to details about all organizations.

Route: GET /api/organizations/:id
Name: GetOrganization
Description: Get the requesting user's organization as no user should have access to details about other organizations. The id parameter must match the requesting user's organization id.

Route: POST /api/organizations
Name: PostOrganization
Description: Create a new organization. Not implemented because organizations are created by their founding user instead.

Route: PUT /api/organizations/:id
Name: UpdateOrganization
Description: Update an organization given an id. The id parameter must match the requesting user's organization id.

Route: DELETE /api/organizations/:id
Name: DeleteOrganization
Description: Delete an organization given an id. The id parameter must match the requesting user's organization id.

10.3.8 Users

UsersRoute: GET /api/users/me
Name: Me
Description: Empty route that requires authentication.

Route: GET /api/users
Name: ListUsers
Description: Get a list of all users for the requesting user's organization. The requesting user will not be able to request users outside of the organization. Can specify the limit, page, and sort for pagination.

Route: GET /api/users/:id
Name: GetUser
Description: Get the user within the requesting user's organization given an id. If the requesting user is not an admin, they can only get themselves.

Route: POST /api/users
Name: PostUser
Description: Create a new user within the requesting user's organization.

Route: PUT /api/users/:id
Name: UpdateUser
Description: Update a user within the requesting user's organization. Also creates activities for each of the typing profiles owned by the user with this change.

Route: DELETE /api/users/:id
Name: DeleteUser
Description: Delete an existing user within the requesting user's organization

11. SDS - BioKey Admin Portal

11.1 Overview

This subsection will explain the design rationale for the administrator portal through the use of annotated user interfaces. The annotations will attempt to bring the user experience to life. Important sections include *Section 11.2.1 Dashboard Interface* and *Section 11.2.5 Typing Profile Details Interface*.

11.2 Annotated User Interfaces

11.2.1 Dashboard Interface ('/' or '/dashboard')

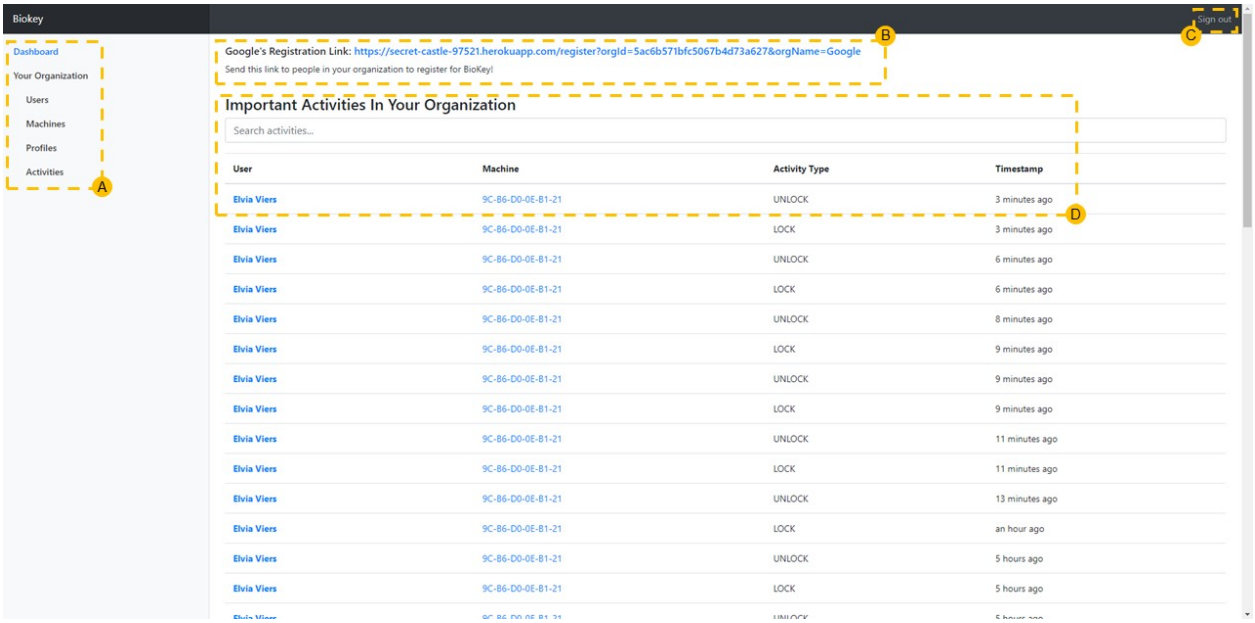


Figure 19: Admin portal - dashboard interface

The dashboard interface is the landing page for the administrator portal after an administrator signs in. Navigating to the index of the administrator portal automatically redirects to the dashboard.

A. This section is the left navigation panel that will be displayed on every interface after an administrator signs in. This panel is designed to be consistent for easy navigation. The administrator will be able to navigate to the interfaces for each of the important models for BioKey.
B. This section is the registration link for the administrator's organization. The administrator can easily copy this link and distribute it to potential users in the organization to have them sign up for an account.
C. This section shows that the 'Sign Out' link is in the top navigation panel which is always shown no matter where the user scrolls down the page. This panel is designed so that administrators can easily sign out for security reasons.
D. This section shows that the dashboard also includes important activities in the organization sorted by the recency of the activity. This panel is designed so that the administrator can easily see new activities that may require attention. The administrator can also search for activities in that list on any column. By clicking any of the rows, the administrator can see more details about the activities. By clicking specifically on the user and machine, they can see details about

them. Finally, the activities are paginated as the list can be overwhelmingly large.

11.2.2 Register Interface (‘/register’)

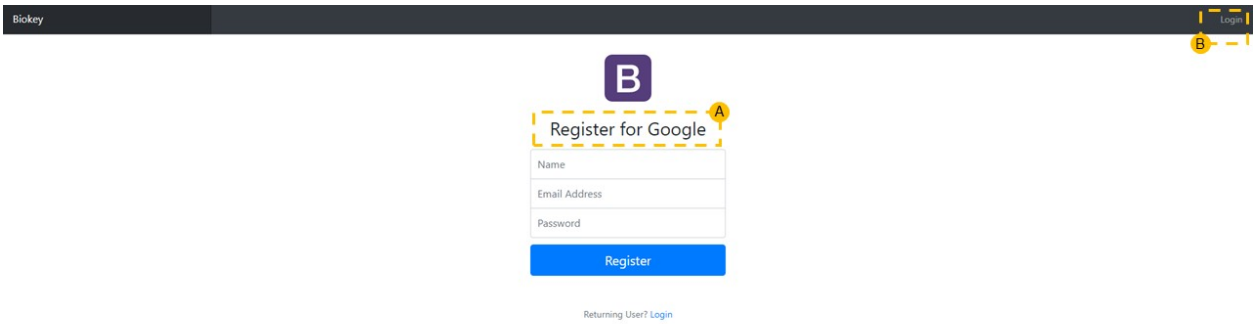


Figure 20: Admin portal - register interface

The register interface is very similar to the login interface. This particular register interface is specific to an organization.

A. This section is the splash message for registration. The organization name is displayed so that the user understands which organization they are registering for.
B. This section shows that just like the ‘Sign Out’ link, the ‘Login’ link is displayed in the same location.

11.2.3 Non-Admin Interface (‘/’ or ‘/users/{id}’)

Figure 21: Admin portal - non-admin interface

The non-admin interface is very similar to admin interfaces. This design is intentional so that administrators are familiar with the interface if their users have any questions.

A. Notice that the left navigation panel is no longer present as non-administrators should not have access to any of the navigation links provided to administrators.
B. Users can change their password, but this process is separate from editing their other information. This design choice is made so that users don't accidentally change their password.
C. Users can save and delete by clicking these links. Saving is not done automatically because users may not be ready to commit their changes. The links are very visible in an uncluttered interface. The delete link also warns the user before committing the change.

11.2.4 Typing Profiles List Interface ('/profiles')

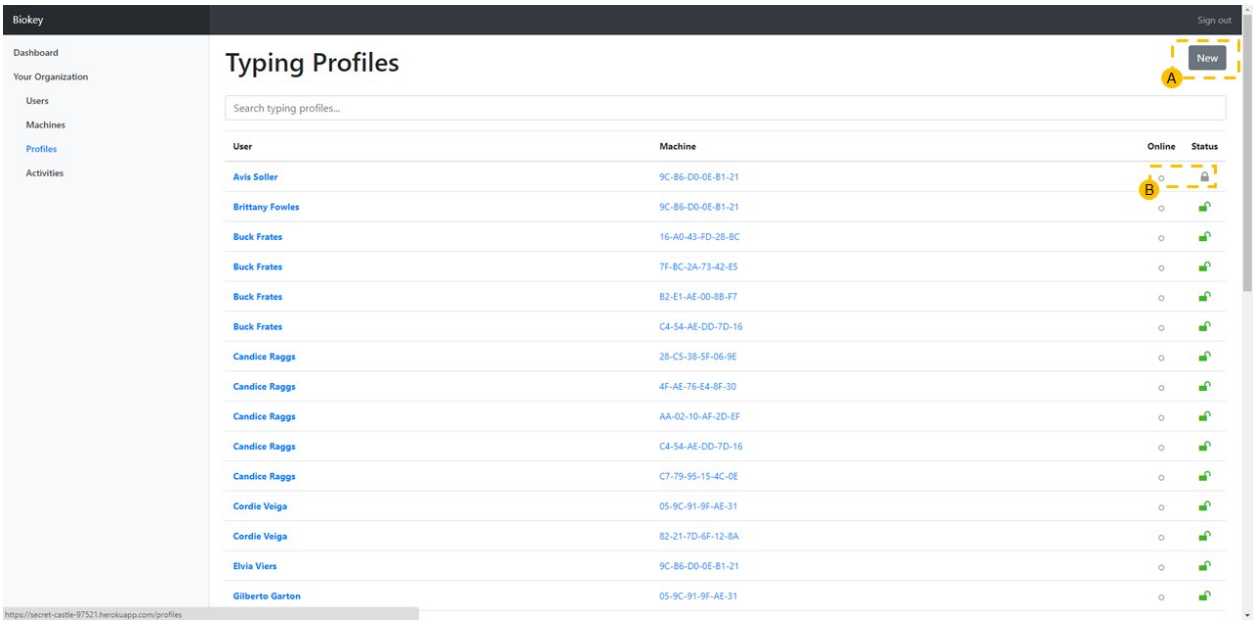


Figure 22: Admin portal - typing profiles list interface

The typing profiles list interface looks very similar to the list of activities in 4.2.1. Again, this is intentional, so administrators can ramp up quickly with the interfaces. This list interface looks the same for all the other entities.

A. There is a link at the top to add a new typing profile.
B. The portal automatically polls the server and updates whether the user is online on a particular machine and whether their status is locked or unlocked. The administrator can easily see typing profiles of interest.

11.2.5 Typing Profile Details Interface (‘/profiles/{id}’)

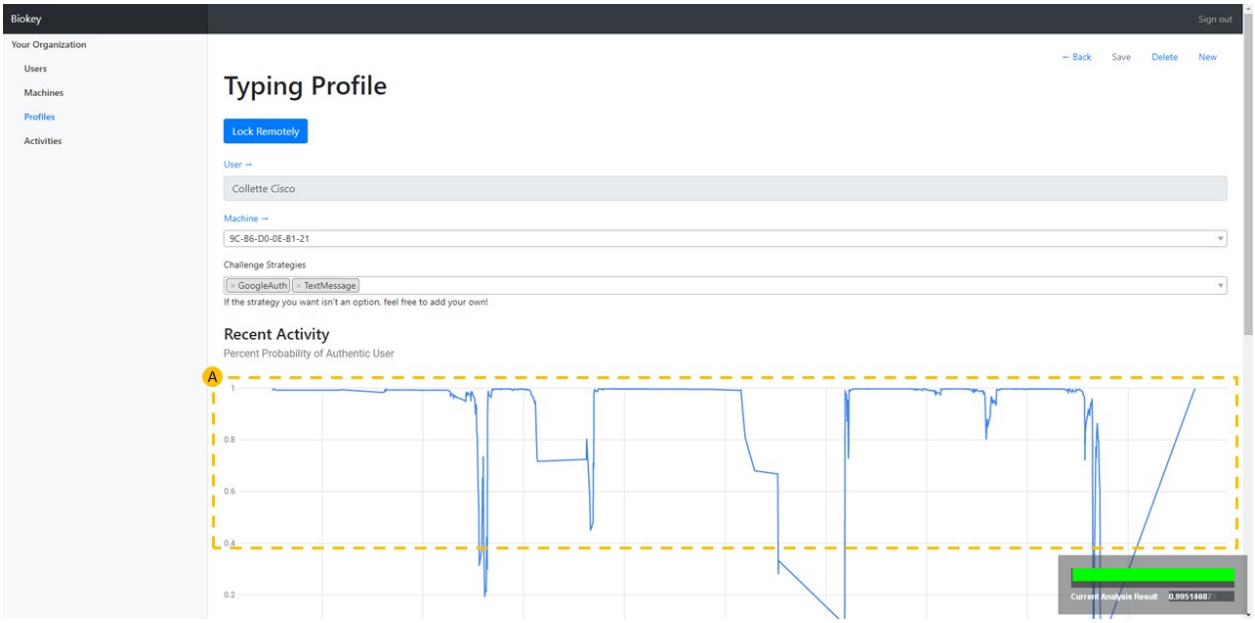


Figure 23: Admin portal - typing profile detail interface, first example

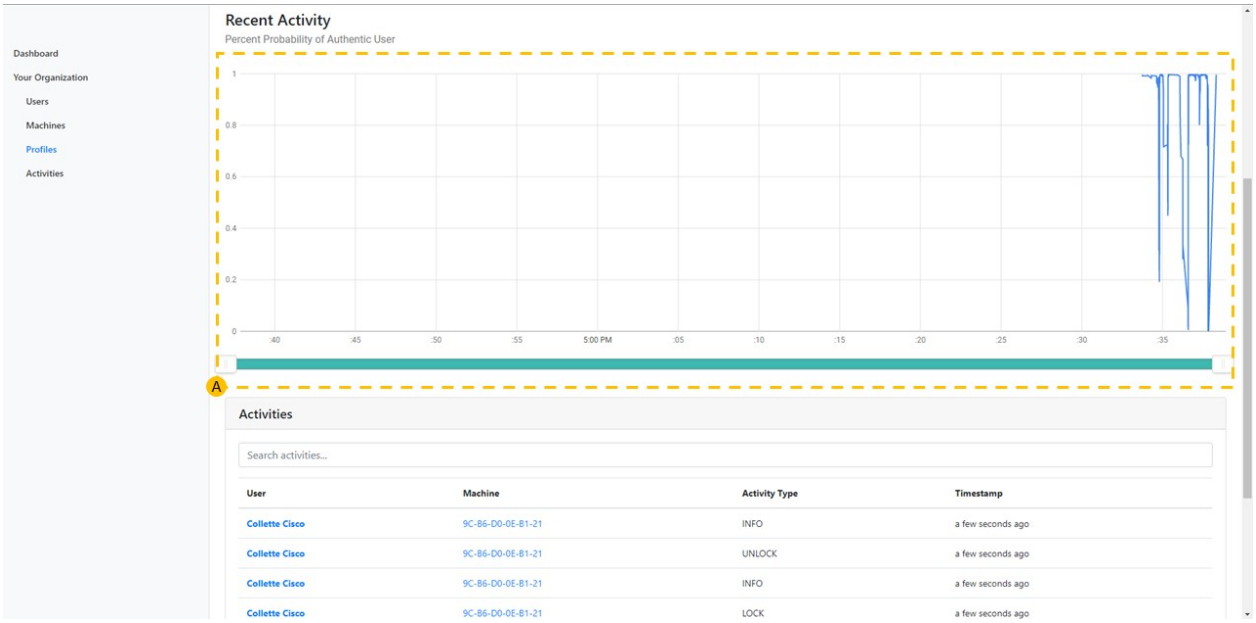


Figure 24: Admin portal - typing profile detail interface, second example

The typing profiles detail page simultaneously also allows administrators to edit the details like whether the profile should be locked or not. This details interface looks the same for all the other entities.

A. The recent activity graph shows the probability that the model thinks the actual user is typing. The graph extends for the last one hour and updates automatically. The bottom scroll bar allows the administrator to adjust the scale of the graph to get a better view of the data.

11.2.6 New Typing Profile Interface (‘/profiles/{id}’)

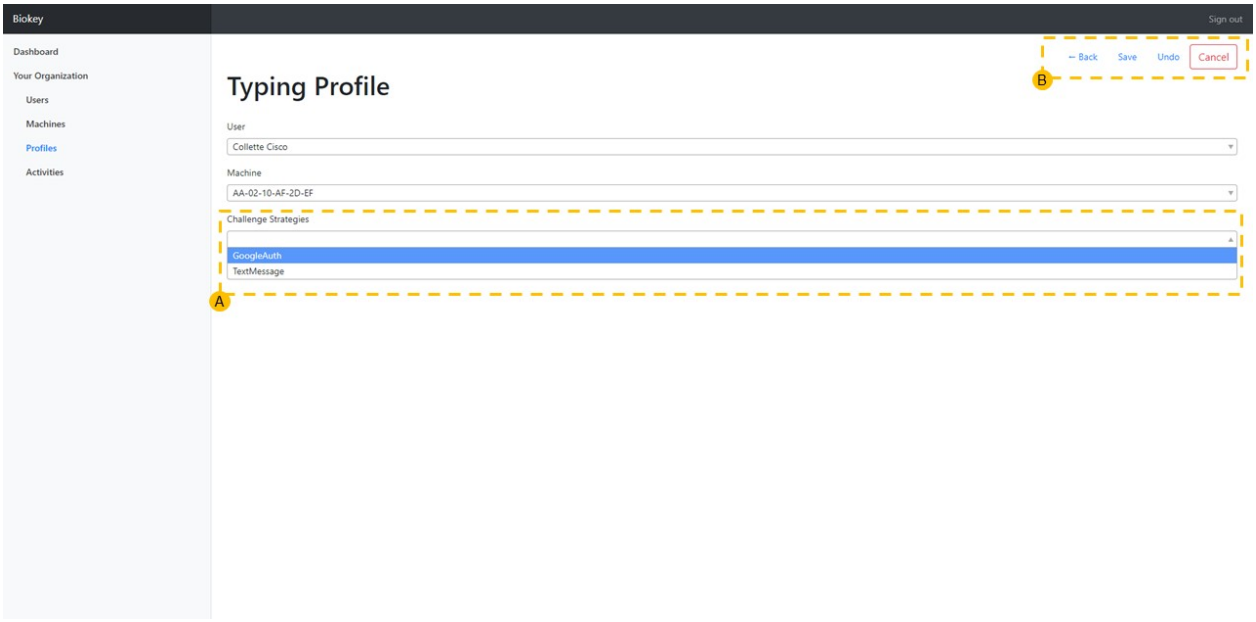


Figure 25: Admin portal - new typing profile interface

The new typing profile page lets an administrator fill in information through dropdowns for the users, machines, and challenge strategies. This new entity interface looks the same for all the other entities.

A. This dropdown menu lets the administrator choose an existing strategy or add a new one.
B. The navigation is still similar to the details interfaces intentionally.

12. SDS – Interactions

12.1 Overview

This subsection will provide the sequence diagrams for selected interactions between the major modules of BioKey. This is intended to clarify the inner workings of BioKey's key functions.

12.2 Boot and Login Sequence

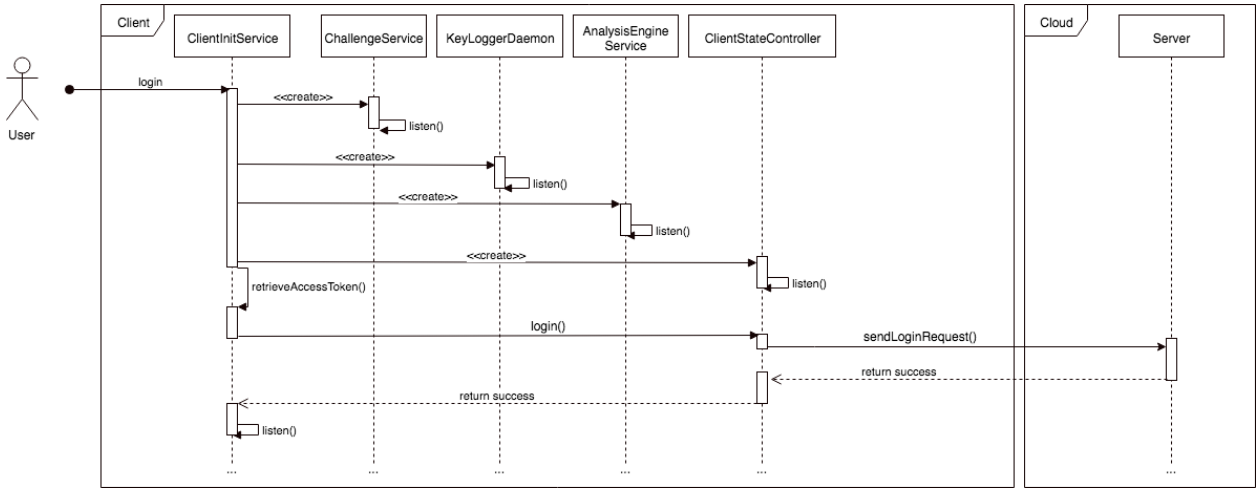


Figure 26: Sequence diagram - module interaction for a system start-up and subsequent login

Upon BioKey's launch, the ClientInitService instantiates all other services, which immediately subscribe to changes in the ClientStateModel. If the ClientInitService detects there is no typing profile, it presents the login frame to the user. When the user makes a login attempt, the ClientStateController submits it to the server. If the submitted credentials are valid, the server returns a success message, and the ClientInitService begins listening to the ClientStateModel.

12.3 Analysis and Lock Sequence

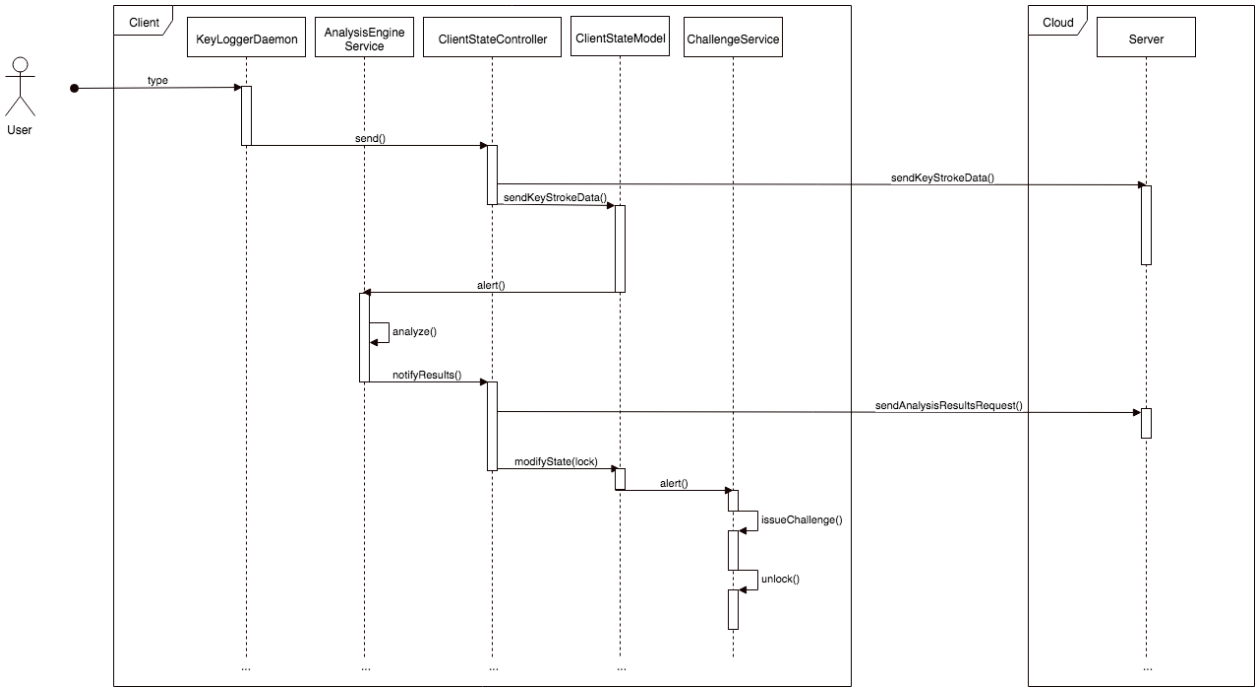


Figure 27: Sequence diagram - module interaction for keystroke analysis and machine locking

During normal operation, the KeyLoggerDaemon accepts typed keystrokes from the user. These are first passed to the ClientStateController, which passes them to storage on the server. Next, the ClientStateController notifies the AnalysisEngineService, which performs analysis on the new keystroke(s). The results of the analysis are sent to the server, and the appropriate changes are made to the ClientStateModel. Finally, the ClientStateModel alerts the ChallengeService if necessary, which takes the necessary actions.

13. SDS - Design Rationale

13.1 Abandoned Modules

There are several ideas that were abandoned prior to the current version of BioKey. Initially, a 'file locker' function was planned for inclusion. This would allow the user to apply authentication via keystroke dynamics to any folder on the machine. It was decided that such a function exceeded reasonable scope for this project.

Furthermore, a 'guest mode' was planned for inclusion that would allow users to temporarily override the typing analysis engine when a colleague uses their machine. This functionality was excluded for two reasons. First, a built-in ability to override the analysis engine poses a security risk. Second, such a function would exceed reasonable scope.

Finally, keystrokes were originally stored locally. By eliminating the need to transmit the keystrokes over insecure channels, the solution may be more secure. However, further research revealed that it would be very difficult to encrypt locally stored keystrokes without diminishing the user experience. Users would need to remember a separate key to encrypt locally (such as a password).

13.2 Design Decisions

13.2.1 System Architecture

The chosen design synchronizes every change in state through a client service and the web server. This is not the most efficient design as additional requests are made between the client and server, but it allows the solution to realize a number of advantages.

First, the use of services and the observer pattern allows BioKey to be extensible and maintainable. New services can be added easily; extending BioKey with a new service is as easy as writing business logic that subscribes to changes to the client state and makes requests to the client state controller. Furthermore, existing services can be maintained and updated with minimal impact on other modules. This design choice makes BioKey easy to maintain and extend in the future.

The technical team has also chosen to make challenge strategies extensible by using the strategy design pattern. This offers two advantages. First, challenge strategies can easily be added and maintained. Second, it provides BioKey customers with a degree of customization that may make the solution more valuable. This design choice trades efficiency for maintainability, extensibility, and user value.

Furthermore, the technical team has chosen to route all requests through the web server and schedule all client state updates through an SQS server. This design choice was necessary for two reasons. First, it essential to have the client and server agree on the state to have a single point of truth. The server and the client should never disagree on the authentication status of the user. This design choice guarantees agreement. Second, it allows an admin user to constantly monitor the activity of a user (through the stream of requests to the web server) and make changes to their machine as necessary (by enqueuing jobs on the SQS server).

13.2.2 Data Handling

The design forgoes resources to save a history of client states. This is a beneficial trade-off because it is imperative that the solution continues to work when there is no connection with the server. By keeping a local history, the client can inform the server of what happened while the connection was lost. This ensures continuity and improves security.

13.3 Deployment Considerations

Deployment consideration had to be made for both the client module and the web-based components of the app.

The client module must be as simple to distribute and install as possible. Therefore, the client module was bundled into a simple package that can be executed for BioKey installation. A detailed walkthrough of this process can be found in *Section 18: User Manual* in the BioKey Final Report.

The BioKey web server and admin portal should be accessible from anywhere and must be able to scale in response to increase demand. Furthermore, it must be fault tolerant and fail-resistant due to its nature as a security application. To meet these needs, BioKey's web-based components are hosted on Heroku. This all but guarantees the ubiquity of access and reliability that BioKey requires.

14. Implementation

14.1 Research & Experimentation

Continuous Authentication through keystroke dynamics is a field of ongoing academic research with no established best practices and limited production solutions on the market. Due to this, BioKey performed significant research and experimentation to develop an algorithm that could accurately verify a user's identity through their keystrokes. Methods outlined in research included statistical approaches, heuristic algorithms, and traditional machine learning classifiers. Due to the lack of recent research, however, only a limited number of papers had begun to explore the use of large and deep neural networks. Our experiments focused on recreating the top performing papers as well as taking influence from them to devise neural network architectures in an effort to extract more information from the keystrokes.

Due to the fact that keystroke events are measured to the millisecond level, early attempts to represent the duration and intervals of raw key events resulted in very large and highly sparse feature vectors (these were represented by a binary 2-D matrix of keys by time series, resulting in a matrix of mostly 0's). Attempts to interpret these feature vectors using various neural network architectures were made, including Convolutional Neural Networks, Long Short-Term Memory (LSTM) Networks, and Phased LSTM Networks. The input sparsity caused large challenges in reaching convergence when training and it was concluded that the use of feature engineering would need to be explored.

One of the top performing papers in the field used a novel heuristic approach that looked at the degree of disorder between durations of n-graphs. n-graphs are sequences of keys (e.g. E-R or E-R-A) that have been typed consecutively. While the paper achieved strong accuracy scores on keystrokes captured during controlled typing sessions, the results were much worse when we tested it on daily computer usage. The paper did, however, provide inspiration for the n-graph feature selection used in the final implementation.

Another paper that influenced the final model also used n-graph durations as features. This implementation devised a Gaussian profile for each user representing the mean and standard deviation of the duration of each distinct n-graph. These values could then be used to score future typing samples through the use of a Gaussian Probability Density function, and then taking the average score found within the sample. BioKey found that while the model was able to distinguish between users, accuracy scores were too low to reasonably provide consistent continuous authentication. The Gaussian scoring, however, provided the foundation for the selected feature engineering strategy implemented in the final algorithm.

14.2 Final Implementation

14.2.1 Generating Gaussian Profiles

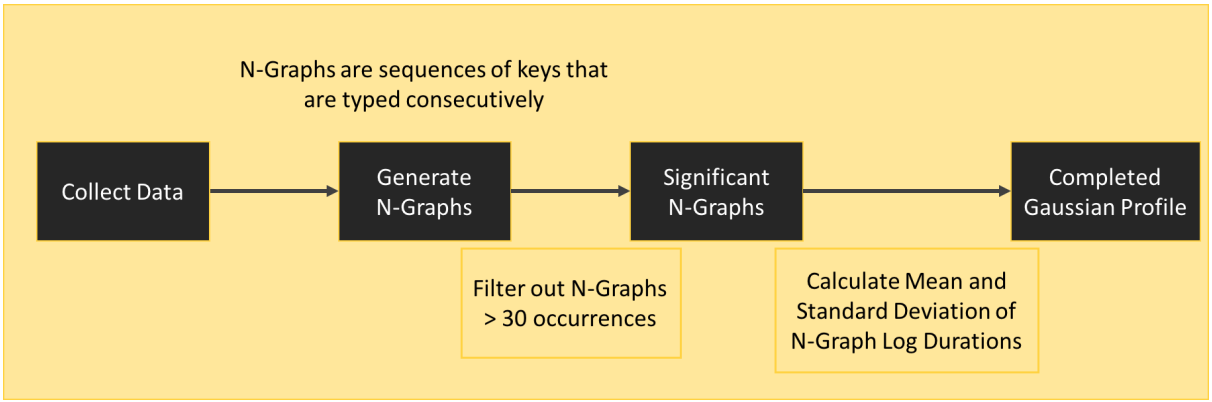


Figure 28: Gaussian profile generation

The first step in creating a typing profile for a user is creating their Gaussian profile.

14.2.1.1 Generating N-Graphs

From the set of keystroke data that exists for a user, n-graphs are created, where the time between any two consecutive keys is within a certain threshold (150 ms). An individual key can be a part of many different n-graphs, provided that it is in the middle of a sequence of keys that have been typed consecutively within the threshold. For example, for the word "make", the n-graphs {M}, {M-A}, {M-A-K}, {M-A-K-E}, {A}, {A-K}, {A-K-E}, {K}, {K-E}, and {E} would be generated. Note that each of these n-graphs has an associated duration value from start to finish.

14.2.1.2 Filtering to Only Significant N-Graphs

After all n-graphs have been generated, they are grouped together. Any groups of n-graphs with less than 30 samples were discarded, as it was considered insignificant to the user's overall typing profile (the number 30 was chosen in keeping with the general principles of the Central Limit Theorem).

14.2.1.3 Completing the Gaussian Profile

For the remaining significant n-graph groups, a mean and standard deviation value are calculated. Based on an analysis of the data, combined with the fact that an n-graph duration cannot be negative, a lognormal distribution was chosen as opposed to a normal distribution. Thus, prior to calculating the mean and standard deviation for an n-graph group, the natural logarithm of the underlying durations is taken.

14.2.2 Training the Neural Network

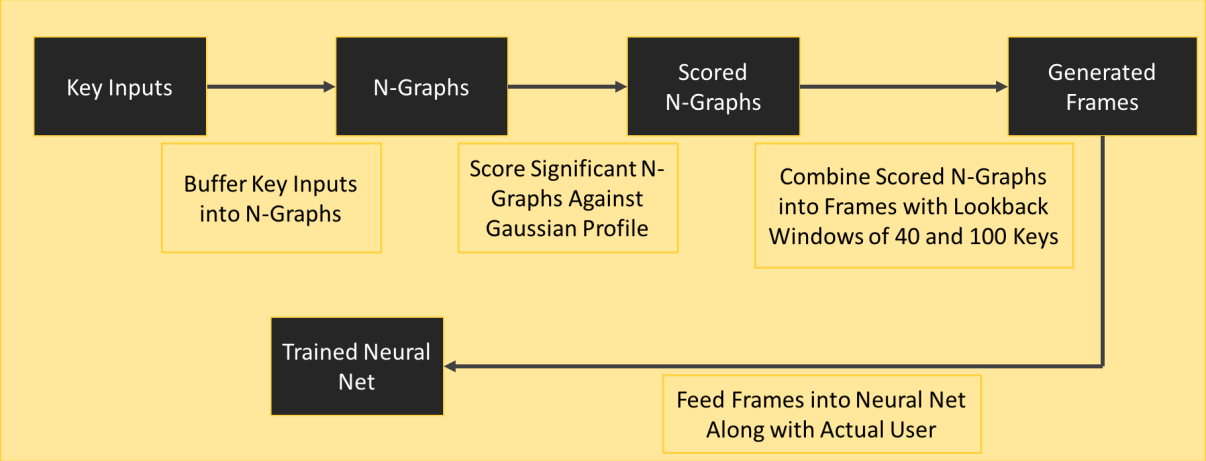


Figure 29: Training process of the neural network

14.2.2.1 Generating a Training and Testing Set

Before training a neural network to apply the Gaussian profile to keystrokes, a training and testing set must be built. Historical keystrokes are used to form training and testing sets for a given user that included simulated attacks by other user's keystrokes in the database. To ensure that simulated keystrokes by either a user or imposter are realistic, we ensured that subsequent keystrokes made by a user within one minute of another are included in the same training set; this will be called a session. Random sessions are then selected from both the user's history as well as other users and a corresponding label of 1 (valid user) or 0 (imposter) is assigned. The resulting training and test sets include sessions from both valid and imposter users and will be used to realistically simulate real world usage scenarios.

14.2.2.2 Interpreting Keystrokes

To overcome the sparsity challenges discussed above, feature engineering was performed to extract more information from keystrokes in a condensed format, via n-graph feature selection and Gaussian probability density scoring. Using the training and testing sets created in the previous section, keystrokes are buffered into n-graphs using the method discussed above. These n-graphs will form the basis of the inputs for the neural network. The network is trained with 3 input vectors, each providing a different time horizon; long term aggregation of 100 keys, medium term aggregation of 40 keys and short term raw log durations of n-graphs ending on the most recent key. Each of these vectors have a length equal to the number of n-graphs in the user's Gaussian profile. For context, an average user has around 500 features in their Gaussian profile.

The long and medium-term input vectors are formed by calculating the scores according to a Gaussian probability density function of all n-graphs that ended within the last 100 and 40 keys respectively. If an n-graph occurred more than once in the window, the mean of the scores is used. All n-graphs that did not appear within the window are assigned a score of 0.5, representing that they were neither typed similarly nor differently from the profile means.

The short-term input vector included all n-graphs that ended only on the most recent key. Instead of Gaussian scores, the raw log duration was used. For all features in the Gaussian profile that did not have an n-graph ending on the most recent key, a value of 0 was assigned.

14.2.3 Neural Network Architecture

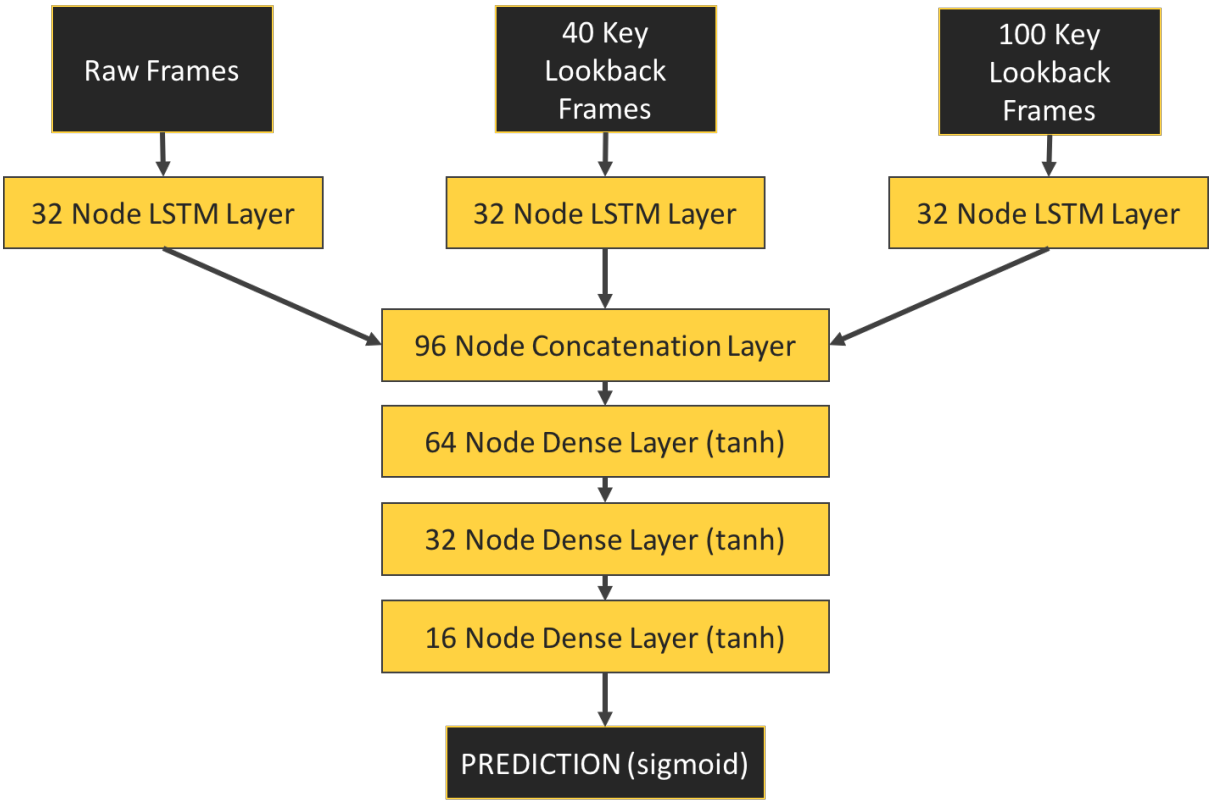


Figure 30: Neural network architecture

After n-graphs are selected and frames are generated from the user's Gaussian profile, a multilayer neural network is built. Each of the 3 input vectors is connected to its own 32 node LSTM layer, modeling the sequenced nature of keystrokes. LSTM layers are a form of Recurrent Neural Network that learn to remember and forget past predictions, enabling them to improve predictions on datasets that are sequenced in nature. Because keystrokes are not isolated events and rely on the events that come before and after, this type of network is well suited to the problem. Each LSTM layer is trained with a lookback window of 100 frames, allowing the cell to leverage memory from at most 100 historical key events and their corresponding frames.

The outputs of the LSTM layers are then concatenated together before being fed through three more fully connected feedforward layers of 64, 32, and 16 nodes to compress the information. Each of these layers uses a *hyperbolic tangent* $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ activation function in order to normalize outputs between -1 and 1. A final prediction layer of 1 node uses the *sigmoid* $S(x) = 1/(1 + e^{-x})$ activation function to output a binary prediction between 0 and 1. This prediction represents the model's belief of the percent likelihood that the current user typing is the valid user.

14.2.4 Optimizing Prediction Results

Given the nature of the application, having volatile prediction results and corresponding actions (lockouts) is not ideal. A single low prediction value from the model could cause a valid user to be unnecessarily locked out and force them to re-authenticate. In response to this problem, we decided to apply a *smoothing* function on model outputs in order to help avoid the impact of outlier predicted values.

The current implementation will lock out a user if at least five of the last ten prediction values are below a threshold, or if the average of the last 20 prediction values are below a different threshold. These thresholds are optimized by user to ensure a balance between user experience and system security. By following this method as opposed to determining lockouts based on a single prediction, the number of false rejections dropped significantly while having a negligible impact on true rejections.

14.2.5 Overall Model Results

Approximately 200,000 keystrokes through the model to simulate a series of user and imposter sessions, each 200 keys in length.

14.2.5.1 Individual Prediction Results

On average, the model predicted a value of 96.65% for actual users and 26.54% for imposters. Evaluating the model against all imposter results is a tough bar. The model often needs multiple keystrokes to determine when an imposter has started typing. When a grace period was added, the model performed significantly better. With a buffer period of 30 and 100 keys, average imposter scores dropped to 17.95% and 10.80% respectively.

The percent of predictions falling into various ranges (0%-10%, 10%-50% etc.) for valid users as well as imposters with various buffer lengths can be found below. As expected, it is clear that model produces significantly higher predictions for valid users than for invalid users.

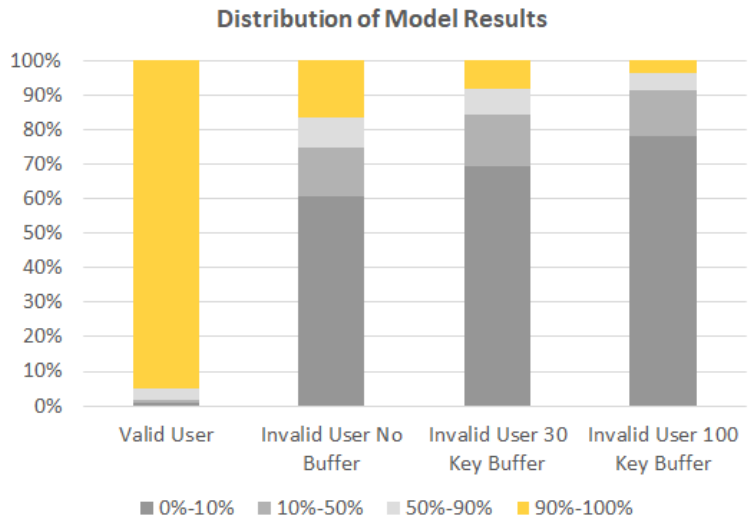


Figure 31: Distribution of analysis model results

14.2.5.2 Real World Results

The simulated 200 key sessions were designed to emulate real world conditions that the model may face – attempting to identify an imposter after a valid user was typing.

In our simulation, 98.8% of imposter sessions were locked out in an average of 47 keystrokes. About 9% of valid user sessions were locked out in an average of 120 keystrokes. A Monte Carlo simulation was used to determine the average number of keystrokes that a valid user would be able to type before being locked out. This simulation yielded a result of approximately 2,200 characters on average before a valid user would be erroneously locked out.

The percent of users of each type locked out as a function of number of keys typed is below. As expected, imposters are locked out far more significantly, and by about 120 keys typed, this line is asymptotically approaching 100%. Conversely, valid users are locked out far less often, and usually only after a significant number of keys. Future model improvements could keep the Valid User line even closer to the x-axis.

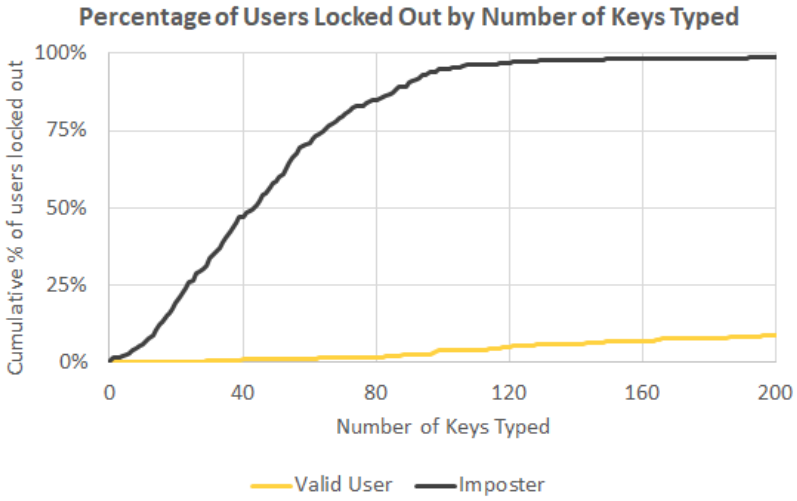


Figure 32: Percentage of users locked out by number of keys typed

14.2.6 Future Model Retraining

On a periodic basis, the same steps described above to build a model can be rerun for each user in order to build a more updated model. This would be done with a worker service designed to run external to the production server. Once the worker has finished, it can update the user's model in the database and alert the client of the change.

Each time the model retrains it well develop a new Gaussian profile as well as new weights in the neural net. As there will be new and increased amounts of data, it is possible that there will be new features in the Gaussian profile.

In order to account for potential changes in a user's typing patterns over time, we have also considered periodically removing older data when training the model as new data becomes available. This would keep the model continually accurate in response to any small changes in typing behaviour exhibited by a user.

15. Test Plan

BioKey is built as three independent modules that communicate with each other. Due to the nature of these systems, they do not work independently without these communications. Therefore, the usefulness of unit testing is limited because the integration between the modules is very important. The focus of this plan is therefore to explain the procedure and design of integration, system, and acceptance testing. These tests will closely mirror the functional requirements identified in the SRS. Test conducted on the prediction model is found in the Implementation section of this report.

15.1 Unit Testing

Unit tests were written for both the client module and backend server module. The development team used the JUnit / PowerMockito test driver and Mocha test driver for the client and backend respectively. Coveralls and Travis CI were used to evaluate the effectiveness of the tests continuously. Unit tests for the client module can be found at: https://github.com/BioKey/biokey-client/tree/master/src/test and for the backend server module at: https://github.com/BioKey/biokey-backend/tree/master/test. Some of the important unit tests are important for integration testing and will be described below.

15.2 Integration Testing

Integration tests on the client module focused on whether the communication between it and the backend server was successful. All test cases **PASSED**.

15.2.1 Test Case 1: Serialize Objects to JSON

Seq. #	Testing Activity	Pass Criteria
1	Serialize each of the Model Objects into JSON	Verify the JSON structure matches the objects the server expects

URL: *https://github.com/BioKey/biokey-client/blob/master/src/test/java/com/biokey/client/helpers/RequestBuilderHelperTest.java*

15.2.2 Test Case 2: Make General HTTP Requests

Seq. #	Testing Activity	Pass Criteria
1	Make a GET request to google.com	Verify that google.com responses with 200 OK

URL: *https://github.com/BioKey/biokey-client/blob/master/src/test/java/com/biokey/client/helpers/ServerRequestExecutorHelperIntegrationTest.java*

15.2.3 Test Case 3: Retrieve / State from System

Seq. #	Testing Activity	Pass Criteria
1	Save ClientStateModel using JPreferences	Saving to JPreferences does not throw any exceptions
2	Retrieve that ClientStateModel back from JPreferences	Retrieve ClientStateModel from JPreferences does not throw any exceptions and corresponds with the one that was saved

URL: *https://github.com/BioKey/biokey-client/blob/master/src/test/java/com/biokey/client/services/ClientInitServiceIntegrationTest.java*

15.2.4 Test Case 4: Make HTTP Requests to the Server and Receive Responses

Seq. #	Testing Activity	Pass Criteria
1	Make HTTP request to login	Server responds with 200 OK and provides the authorization token
2	Make HTTP request to retrieve client status	Server responds with 200 OK and provides the client status
3	Make HTTP request to POST keystrokes	Server responds with 200 OK
4	Make HTTP request to POST analysis results	Server responds with 200 OK
5	Make HTTP request to POST client status	Server responds with 200 OK
6	Make HTTP request to POST heartbeat	Server responds with 200 OK

URL: *https://github.com/BioKey/biokey-client/blob/master/src/test/java/com/biokey/client/controllers/ClientStateControllerIntegrationTest.java*

15.2.5 Test Case 5: Google Authentication

Seq. #	Testing Activity	Pass Criteria
1	Initialize new Google authentication key	Key is generated without throwing exception
2	Use Google authentication key to generate a one-time password	The generated one-time password passes the challenge

URL: *https://github.com/BioKey/biokey-client/blob/master/src/test/java/com/biokey/client/controllers/challenges/GoogleAuthStrategyIntegrationTest.java*

15.3 System Testing

System testing was conducted on the backend server module, client module, and admin portal combined. The objective was to treat the systems as black boxes and ensure that the functional and non-functional requirements are met. These requirements can be found in the Software Requirements Specification (SRS) Sections 3 and 4 of the BioKey Final Report. All test cases **PASSED**.

15.3.1 Test Case 1: Create an account and login

Seq. #	Testing Activity	Pass Criteria
1	Create an account with e-mail and password on the web portal	No errors with account creation
2	Login into client with the same credentials	Login is successful and personal activity can be viewed
3	Check the account information and the new typing profile that has been created	Account information matches credentials and page loaded in two seconds
4	Check if administrator receives a notification for a new user	Typing Profile created with the correct machine
5	Shutdown new user's computer	Administrator receives a notification within ten seconds
6	Restart computer	No criteria for restarting computer
7	Check if user is still logged in	User is still logged in after restart

15.3.2 Test Case 2: Typing and local locking and unlocking

Seq. #	Testing Activity	Pass Criteria
1	Using a new account, pass key logger 10,000 keystrokes of User A data without internet access	No criteria for passing keystrokes
2	Check if key data has been uploaded to web portal	Key data not uploaded to web portal
3	Enable internet access and perform activity 2 again	Key data uploaded to web portal within five seconds
4	Pass key logger 1000 30-character bursts of User A data and observe the probabilities returned by the analysis engine	FRR is in line with expectations
5	Prevent OS locking and pass key logger 1,000 30-character bursts of User B data and observe the probabilities returned by the analysis	FAR is in line with expectations.

	engine	
6	Re-enable OS locking and pass key logger User B data until computer becomes locked	Locking takes less than five seconds, observer that a variety of key commands such as ALT + TAB no longer have effect and explorer.exe is destroyed
7	Observe challenges shown	Challenges shown match what the administrator has set
8	Interact with challenge and provide the correct credentials	Challenges react with success in providing credentials
9	Observe OS lock status	Unlocking takes less than five seconds
10	Check if administrator receives a notification for a user challenge success	Administrator receives a notification within ten seconds
11	Check if user is unlocked following a challenge success	User is unlocked within ten seconds

15.3.3 Test Case 3: Remote locking and unlocking

Seq. #	Testing Activity	Pass Criteria
1	Setup the system to be in the state just after test case 2	No criteria for entering the state
2	Pass key logger User B data until computer becomes locked	Locking takes less than five seconds
3	Observe challenges shown	Challenges shown match what the administrator has set
4	Interact with challenge and purposely provide incorrect credentials until user runs out of attempts	Challenges react with failure to provide correct credentials
5	Observe OS lock status	OS is still locked
6	Check if administrator receives a notification for a user lock-out	Administrator receives a notification within ten seconds
7	Check if user receives a notification for a user lock-out	User receives a notification within ten seconds
8	Administrator sees notification in activity log in web portal	Administrator sees notification in activity log in web portal
9	Administrator navigates to the user's profile via a link in the notification	Administrator sees the link and the link navigates correctly to user's profile
10	Observe the 'locked' status in the client's data	Administrator sees that the profile is currently locked

11	Administrator edits the ‘locked’ status to ‘unlocked’	Administrator is able to change the status
12	Administrator refreshes the page after five seconds	Administrator sees that status remains changed after refreshing the page
13	Observe that the user’s OS has been unlocked	User resumes full control of the computer
14	Administrator edits the ‘locked’ status to ‘locked’	Administrator is able to change the status
15	Observe that the user’s OS has been locked	OS is locked

15.3.4 Test Case 4: Administrator actions in the web portal

Seq. #	Testing Activity	Pass Criteria
1	Create a new administrator account which will create a new organization	New organization created with administrator account
2	Using the administrator account assigned to the organization, assign additional administrators to the organization	New administrators are assigned to the organization
3	Using an administrator account assigned to the organization, add user accounts to the organization	New users are added to the organization
4	Using an administrator account assigned to the organization, delete a user account from the organization	Users are deleted from the organization
5	Using an administrator account assigned to the organization, assign a machine to a user within the organization	Machine is assigned to a user
6	Using an administrator account assigned to the organization, remove a machine from a user within the organization	Machine is no longer assigned to a user
7	Using an administrator account assigned to the organization, edit/reset a password for a user within the organization	User password is reset
8	Using an administrator account assigned to the organization, delete an existing typing profile and then create a new one for a user within the	User typing profile removed and replaced with new profile. Security threshold value changed

	organization. Then edit that typing profile by changing values such as the security threshold	
9	Using an administrator account assigned to the organization, edit the information of a user in the organization such as the authentication status	Authentication status of user changed
10	Using an administrator account assigned to the organization, change organization wide setting such as accepted challenge strategies	Organization wide settings changed

15.3.5 Test Case 5: Load testing

Seq. #	Testing Activity	Pass Criteria
1	Have user type at 100 words per minute	Observe no lag on the computer and a constant stream of prediction results being registered on the admin portal

15.3.6 Test Case 6: Usability testing

Seq. #	Testing Activity	Pass Criteria
1	Have a non-technical user read manual	Manual should be read within 15 minutes
2	Have non-technical user perform test case 4	Test case 4 should be completed successfully

15.3.7 Test Case 7: Installation testing

Seq. #	Testing Activity	Pass Criteria
1	Install BioKey	BioKey was successfully installed in one minute

15.4 Acceptance Testing

Acceptance testing was conducted on both the backend web server module and local client systems. The objective was to use to real data to test functional requirements from an end user perspective. These tests were not performed by the technical team like the system tests but by actual end users. For the purposes of BioKey, the tests were performed by classmates of the technical team with supervision from the technical team. As such, these tests were designed to be straightforward tasks related to the functional requirements. All test cases **PASSED**.

15.4.1 Client Module: Perform local, continuous keystroke analysis

Requirement	Test Cases
Record keystroke data (1)	● None; testing for this requirement will be done in conjunction with acceptance tests of requirement 4
Analyze keystroke data (2)	● Some testing for this requirement will be done in conjunction with acceptance tests of requirement 4 ● User types at least an hour in their ten most common applications *(experiences few false rejections switching between these applications)*

15.4.2 Client Module: Detect suspicious behaviour

Requirement	Test Cases
Assess analysis output (3)	● While typing, view the analysis output in the bottom corner changing in response to keystrokes ● If the prediction values fall too low, view the computer entering the locked state

15.4.3 Client Module: Lock the machine

Requirement	Test Cases
Lock the machine (4)	● After training your profile, have another tester type on your computer for at most 50 characters without internet access *(computer locks and user loses access to desktop)* ● After training your profile, have another tester type on your computer for at most 50 characters with internet access *(computer locks and user loses access to desktop)* ● Without internet access, reboot computer in locked state *(computer remains locked)* ● With internet access, reboot computer in locked state *(computer remains locked)*
Issue challenges and determine results (5)	● Without internet access, reboot computer in locked state *(computer remains locked)* ● With internet access, reboot computer in locked state *(computer remains locked)* ● Respond with incorrect credentials to a challenge without internet access *(computer remains locked for challenges that don't require internet access, computer asks for internet access for challenges that do)* ● With internet access, respond with incorrect credentials to a challenge *(computer remains locked)* ● Without internet access, respond with correct credentials to a challenge *(computer unlocks for challenges that don't require internet access, computer asks for internet access for challenges that do)* ● With internet access, respond with correct credentials to a challenge *(computer unlocks)*
Unlock the machine (6)	● None; testing for this requirement will be done in conjunction with

	acceptance tests of requirement 5

15.4.4 Client Module: Maintain client status

Requirement	Test Cases
Maintain login status of a user (7)	● Without internet access, reboot computer and check if logged in *(should succeed)* ● With internet access, reboot computer and check if logged in *(should succeed)*
Maintain state agreement with the server (8)	● None; testing for this requirement will be done in conjunction with acceptance tests of requirement 17 and 19
Respond to server messages from administrator (9)	● None; testing for this requirement will be done in conjunction with acceptance tests of requirement 13

15.4.5 Client Module: Recover from a lost connection

Requirement	Test Cases
Update state after connection is re-established (10)	● Go on airplane mode on the client, then lock remotely, turn off airplane mode and the computer should respond to change in state *(should lock the computer)*

15.4.6 Admin Portal: Allow administrators to view dashboard

Requirement	Test Cases
View activity logs for users within organization (11)	● As the administrator, view a user's activity log *(should show most recent activities)* ● As the administrator, view all users' activity log within the organization *(should show most recent activities)*
Notify administrator of important activity for users within organization (12)	● After a user gets locked out of their machine, observe their administrator's notification log *(should show lockout event)* ● After a user gets locked out of their machine, observe that administrator receives a text message *(text message should show lockout event)*

15.4.7 Admin Portal: Allow administrators to remotely lock or unlock machines

Requirement	Test Cases
Remotely lock or unlock machines (13)	● As an administrator, edit the user's lock status *(should reflect on user's laptop and should reflect on user's laptop)* ● As an administrator, edit the user's lock status then refresh the page *(changes should still be present)*

15.4.8 Admin Portal: Allow administrators to manage entities

Requirement	Test Cases
Create administrator account and organization (14)	● Create an administrator account and associated organization *(should succeed)*
Assign users within organization as administrators (15)	● As an administrator, assign admin permissions to a user *(should succeed and notify user)* ● As an administrator, assign admin permissions to a user then refresh the page *(change should still be present)*
Create or delete users (16)	● With internet access, create an account within an existing organization *(should succeed)* ● As an administrator, delete the user's account *(should succeed, navigate back to another page)*
Edit user accounts (17)	● As an administrator, edit the user's account information *(should be reflected on user's laptop and should notify user)* ● As an administrator, edit the user's account information then refresh the page *(change should still be present)*
Create typing profiles (18)	● As an administrator, create a new typing profile and associate it with a new machine *(should reflect on user's new machine when user logs in and should notify user)* ● As an administrator, create a new typing profile then refresh the page *(changes should still be present)*
Edit typing profiles (19)	● As an administrator, edit the user's typing profile's challenge strategies *(should reflect on user's laptop and should notify user)* ● As an administrator, edit the user's typing profile's challenge strategies then refresh the page *(change should still be present)*
Edit organization settings (20)	● As the administrator, make a change to the settings of an organization that you belong to *(should succeed and notify users)* ● After submitting changes, refresh the page *(changes should still be present)*

15.4.9 Admin Portal: Allow users to manage their own profiles

Requirement	Test Cases
Create regular user account using company email and a unique password (21)	● Open the link from administrator *(should show organization name)* ● Create an account within an existing organization *(should succeed)*
Regular users can edit their profile but cannot see their activity (22)	● Change the password, sign out, then login with new password *(should successfully login)* ● Cannot navigate to any other page except editing their profile *(should succeed)*

16. Conclusion and Recommendations

Our team has worked hard over the last six months and has overcome a number of challenges in developing BioKey. At its core, BioKey was about developing a unique typing profile for each user and then comparing that profile to the current user that is typing. If there is an inconsistency between these two data sets, the software should have the ability to completely lock the user out until an administrator intervenes or the user re-authenticates via SMS or Google Authenticator. While we could have left BioKey at this stage, we decided to take it one step further and develop an admin-facing portal that allows IT personnel in a company to manage all computers and users in the business, as well as being alerted immediately when something nefarious is suspected. We did this because from a business perspective, BioKey is targeted as an all-in-one security software for large organizations.

We as a team had to overcome a number of challenges in developing this software:

- ***Obscurity of the field***: There is no de facto industry standard for continuous authentication via keystrokes. Even after reading a number of academic papers, there was still no clear solution as to how we should develop the crux of our software. Therefore, our team had to become creative with our ideas in order to meet these challenges.
- ***Sparse data***: Recurrent neural nets often struggle to converge with highly sparse data.
- ***Processing large amounts of data***: To build the neural nets, we had to collect the keystroke information, and this resulted in an extremely large data set. Analyzing and processing these large data sets is not an easy task and forced us to be more conscious about our algorithmic efficiencies.
- ***High security expectations***: We had to consider the numerous ways in which BioKey can be circumvented. This can range from seemingly simple tasks such as precluding the use of opening task manager in the lock screen to opening another program altogether.
- ***Lack of Knowledge***: This project required us to conduct a lot of research in statistics and neural net implementation and use these tools to create our software.

We were able to overcome expectations and still meet the functional requirements as set out in the MVP. With that being said, given the constraints imposed on this project - notably time and budget - BioKey still has a long way to go before it can be used in a professional, business setting.

Further iterations of BioKey would work towards adding the following functionality:

- ***Continuous authentication via mouse movements***: It's quite clear that a big assumption in our implementation of our MVP of BioKey is that it assumes nefarious activity involves using the keyboard to a large degree. This is not necessarily true as one can do a lot of damage just through mouse movements and clicks. Therefore, using mouse movement profiles for users simultaneously with keystroke profiles will significantly improve the overall efficacy of BioKey.
- ***Implementation across other operating systems***: The current MVP is limited to computer systems that run Windows 7 or later. In order to increase the total addressable market of BioKey, it would be beneficial to ensure BioKey has full functionality on macOS and Linux.
- ***Disabling port input***: The current MVP fails to preclude a user's computer from reading information from input ports (e.g. USB keys) when the computer is locked. It's possible that an imposter could load a script on a USB key to override the lock screen and thus circumvent the software. This is clearly a significant threat to the efficacy of BioKey and should most certainly be addressed in future iterations.

With that being said, it is quite clear that continuous authentication provides a whole new level of security for user authentication and improves on a number of flaws with the mainstream, one-time user authentication implementations. Projects like BioKey are done to demonstrate a proof of concept, and we are excited that an application like ours can contribute to the cyber security and data privacy of large corporations.

17. Appendix

Note: all source code can be found on GitHub: https://github.com/BioKey.

Study	Data size	Latency	Input repetition	Input freedom	Method	FAR (%)	FRR (%)	EER (%)	Input sessions	Device freedom
[113]	33	FT	8	Yes	Mean, standard deviation	0.25	16.36	—	—	No
[61]	26	FT	30	Yes	Bayesian and minimum distance classifier	2.8	8.1	—	Once	No
[155]	24	FT	2	Yes	Perceptron algorithm	8	9	—	Yes	No
[9]	15	DT, FT	—	Yes	ART-2, RBFN, and LVQ	—	—	0	Yes	No
[103]	10	DT, FT	20	Yes	Inductive learning classifier	9	10	—	—	No
[84]	21	DT, FT	150–400	Yes	Autoassociative multilayer perceptron, SVM	0	0.814	—	—	—
[134]	100	DT	100	No	Genetic algorithm	—	—	95*	Yes	—
[21]	41	DT, FT	5	No	Random forest decision tree	—	—	2	Yes	Yes
[42]	30	DT, FT	10	No	Sequence alignment algorithms	0.15	0.2	0.35	Yes	—
[17]	21	DT, FT	—	Yes	K-means, euclidian	—	—	3.8	Yes	—
[86]	100	DT, FT	—	No	Multilayer perceptron	1	8	—	Once	—
[39]	41	DT, FT	30	Yes	Gaussian mixture modeling	4.3	4.8	4.4	Yes	Yes
[89]	100	DT, FT	6	No	Bayesian, Euclidean, hamming distance	—	—	6.96	Yes	No
[67]	1254	DT, FT	20	No	Mean, standard deviation	16	1	—	Once	Yes
[96]	16	DT, FT	5	No	Bayesian, Euclidean	—	—	4.28	Yes	—
[108]	25	DT, FT	30	Yes	Gauss, Parzen, K-NN, K-mean	—	—	1	—	—
[123]	51	DT, FT	50	No	Manhattan distance	—	—	7.1	Yes	No
[93]	100	DT, FT	12	No	Support vector machine	—	—	15.28	Yes	No
[3]	100	DT, FT	10	No	Gaussian PDF, direction similarity measure	—	—	1.401	Once	Yes
[138]	117	DT, FT	5	Yes	Support vector machine	—	—	11.83	Once	No

* Indicates performance measurement in terms of accuracy, similar but inverse to EER where value closer to 100% indicates better performance.

Exhibit 1: "Comparison between research works with static authentication mode (short inputs)."

Study	Data size	Maximum number of graphs	Freedom of input	Method	FAR (%)	FRR (%)	EER (%)	Platform
[20]	31	di-graph	Yes	Weighted mean, Standard deviation	—	—	90*	OS
[58]	63	di-graph	Yes	*k*-nearest neighbor	—	—	83.22–92.14*	OS
[59]	205	*n*-graph	Yes	Degree of disorder, mean, standard deviation	0.5	5	—	Web
[124]	22	di-graph	Yes	Bhattacharyya distance, goodness measure	—	—	86.47*	OS
[74]	61	*n*-graph	No	SVM and decision tree J48	14.5	1.78	—	OS
[105]	21	di-graph	Yes	Degree of disorder and histogram-based Density estimation	0.14	1.59	—	OS
[72]	10	di-graph	—	Random forest decision tree	0.41	0.63	0.53	—
[107]	21	di-graph	Yes	Random forest decision tree	3.47	0	1.73	Web
[75]	30	di-graph	No	*k*-nearest neighbor	—	—	0.5	Web
[22]	55	*n*-graph	Yes	Spearman's foot-rule distance-metric	2.02	1.84	—	Web
[76]	186	di-graph	Yes	Degree of disorder	1.65	2.75	—	Web

* Indicates performance measurement in terms of accuracy, similar but inverse to EER where value closer to 100% indicates better performance.

Exhibit 2: "Comparison between research works in dynamic authentication mode."

18. User Manual

18.1 System Overview

The BioKey system is made up of a **client** that is installed on user's machines and a web-based **administration** portal.

18.1.1 System Requirements

BioKey client supports Windows 7 or later with a minimum of 4GB of RAM and 500MB of free storage. Administrator portal requires any computer with a modern browser.

18.2 Getting Started

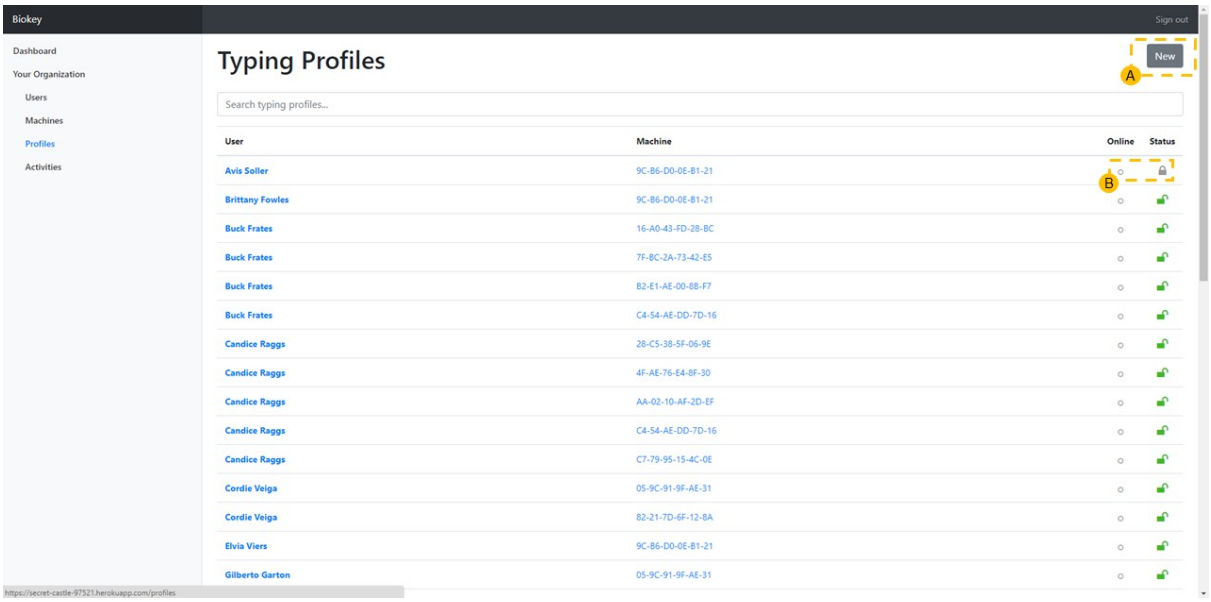
18.2.1 Part 1: Create Admin User & Organization

Take the following steps:

1. Navigate to https://secret-castle-97521.herokuapp.com/register
2. Register for a new account with BioKey

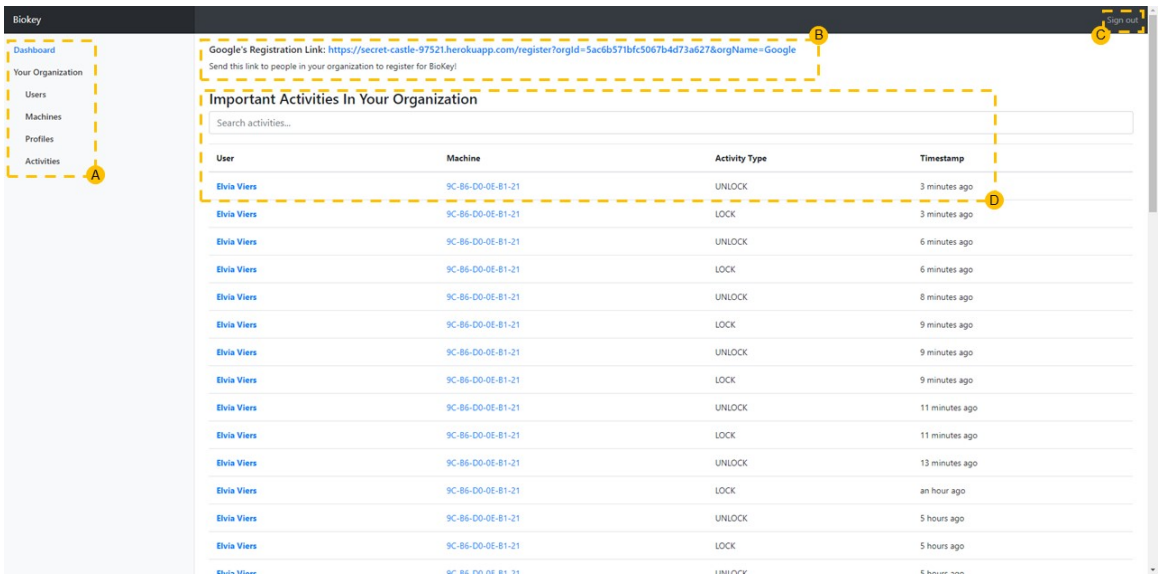
18.2.2 Part 2: Create a User Account

Invite users to the organization through one of two methods



Manual Figure 1: Creating a new user manually

1. Add them manually
    a. Navigate to the Users page through the left sidebar
    b. Click new in the top right corner (Manual Figure 1: A)
    c. Enter relevant details and click *save*



Manual Figure 2: Inviting a new user to sign up

2. Invite the new user to sign up
    a. Navigate to the Dashboard through the left sidebar (the picture above is the dashboard)
    b. Copy the link provided at the top of the page (Manual Figure 2: B)
    c. Share the link with users to sign up through the register page

18.2.3 Part 3: Install BioKey Client

Take the following steps:

1. Download the BioKey client
2. Run the installer to install BioKey and required files
3. Enter the administrator password to grant BioKey access to your system
4. Restart your computer to complete the install

18.2.4 Part 4: Setup BioKey Client

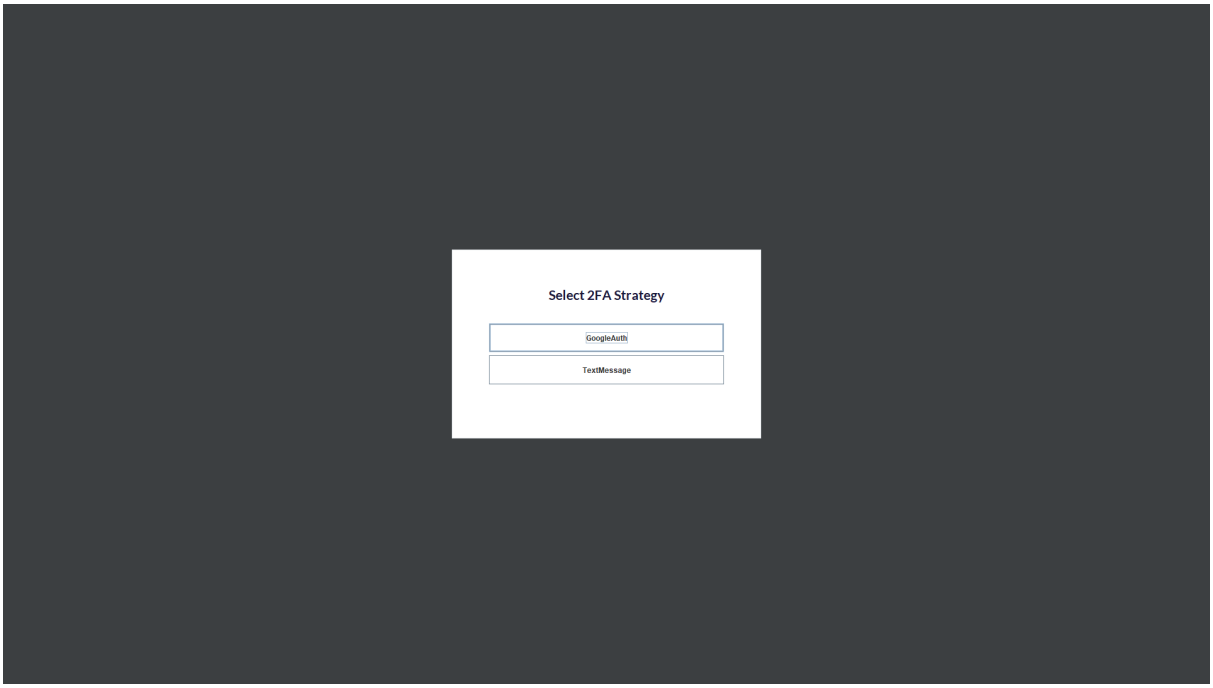
Take the following steps:

1. BioKey should launch with your system and ask you to login
2. Login with the email and password pair created above
3. BioKey will ask to setup two-factor authentication. Scan the barcode using a two-factor application (i.e. Google Authenticator) and enter the resulting code in to BioKey to create a challenge recovery strategy
4. BioKey is now running on your system!

18.2.5 Part 5: Re-Authenticate Through Challenge

Take the following steps:

1. In the event the system identifies a user as being fraudulent, the system will lock and display the screen below



Manual Figure 3: The BioKey 2FA selection screen

2. To re-authenticate, select a challenge strategy and follow the instructions outlined for your chosen strategy

18.2.6 Part 6: Re-Authenticate Through Admin

Admins can also remotely unlock the computer via the admin portal:

1. Navigate to the locked Typing Profile
2. Click the “Unlock Remotely” button in the top left of the screen

18.2.7 Part 7: Lock Remotely Through Admin

In the event the admin needs to remotely lock a computer, this can be done through the admin portal:

1. Navigate to the targeted Typing Profile
2. Click the “Lock Remotely” button in the top left of the screen

19. Curriculum Vitae

19.1 Connor Giles

Connor Giles

1158 Beechgrove Crescent, Oakville ON
cell : '905-808-1381' ; **email :** 'connorgiles@icloud.com' ; **web :** 'http://connorgiles.com' ;

Personal Projects

Sift - http://devpost.com/software/sift-6068p

Designed and developed an iOS application (Python backend) that consolidates news articles from different feeds and summarizes them using natural language processing. Built during the Hack Western Hackathon and won awards for "Best Designed App" and "Reinvent the Reading Experience".

Cryptocurrency Trading Algorithm

Developed a Node.js application that monitors a market's order book for arbitrage opportunities, executing live trades when an opportunity is discovered. Market data was accessed via the market's websocket API and processed in real time as orders enter the market for minimal latency.

Education

Bachelor of Engineering Science, Software Engineering
Western University, London, Ontario
Sept. 2013 - Present (Graduating year: 2018)
Honours Business Administration
Western University, Ivey Business School, London, Ontario
Sept. 2015 - Present (Graduating year: 2018)

Languages

JavaScript, Python, Swift, HTML, CSS, Node.js, PHP, C++, SQL, Salesforce Apex, VisualForce, Java, C#

Software

Sublime, Xcode, Visual Studio, Git, Sketch, Photoshop, Illustrator, Salesforce, Talend Data Integration

Employment

Co-Founder & Creative Lead (June 2015 - Present) @ SMPL Studios
SMPL studios is a team of freelance tech creatives that have worked on both private and partnership projects, applying our love for simplicity and elegance to our design and programming. (http://smplstudios.com)

Product Manager & Developer (June 2013 - August 2016) @ Dynamic Connections

- Managed a team of developers building products to increase value add for customers and streamline inefficient processes for internal transportation dispatchers using web technologies and Salesforce.
- Resulted in 12% workload reduction for dispatchers and was key factor in acquisition of company.
- Built dashboards using Excel, Salesforce and Python for detailed daily measurement that is transparent to all employees.
- Designed a standard design language that was applied to all communication templates.

Activities

Ivey Orientation Leader (2016)
Helped to assure a smooth transition to Ivey for students, providing support and resources throughout the various stages of their HBA1 year.
Provincial Rugby Player (2008 – 2013)
Toured throughout Canada and the United States, competing from club to international levels of rugby.

19.2 Brandon Kucera

BRANDON KUCERA

Tel: (226) 228-6485 | 16 Henderson Court, Ingersoll, ON, N5C 0A1 | Email: bkucera.hba2018@ivey.ca
ca.linkedin.com/in/brandonkucera

EDUCATION

Ivey School of Business, Western University (UWO), London, ON, Canada — Spring 2018
Candidate for Bachelor of Arts, Honors Business Administration (HBA), GPA: 83%

- HBA Dean's Honor List: HBA1 average in top 25% (2016)
- Mary and Barry Ullett 125th Anniversary OSOTF: Ivey Scholarship for academic excellence (2015)

Western University (UWO), London, ON, Canada — Spring 2018
Candidate for Bachelor of Engineering Science, Software Engineering, GPA: 84%

- Proficient in C++, C#, Java, HTML, and CSS programming languages and UML modelling language
- Understanding of JavaScript, jQuery, software development lifecycles and SQL database design
- 3x Engineering Dean's Honor List: maintained yearly average of above 80% (2013-2016)
- Western Scholarship of Excellence: incoming average of 90% or above (2013)

EXPERIENCE

BMO Financial Group, Toronto, ON, Canada — 2017 – Current
Analyst, Application Design and Configuration

- Evaluated two enterprise software platforms with thousands of internal users to recommend and implement usability and functionality improvements
- Standardized the testing protocols for supplier relationship management platform, saving hundreds of hours of development and configuration effort
- Analyzed internal buyer behaviour and made recommendations to drive cost savings

Ivey School of Business, London, ON, Canada — 2016 – Current
Research Assistant

- Researched strategic situations that defy traditional rules of economics in industries such as ecommerce; for example, performed a full industry analysis on drop shipping and its long-term sustainability
- Analysed findings to form hypotheses on the characteristics and long-term sustainability of industry trends
- Presented hypotheses and evidence to an Ivey strategy professor currently on exchange at Wharton Business school to provide possible material for cases to be used at top business schools

CAMI Assembly, Ingersoll, ON, Canada — 2016
Production Associate

- Constructed and inspected over 500 in-progress vehicles per day, resulting in efficient shift operations
- Performed vital welding quality control and ensured appropriate containment measures were activated

EXTRA-CURRICULAR ACTIVITIES

Ivey Technology Club, *Vice President Academic* — 2016 – 2017

- Prepared materials for Ivey students recruiting for technology-based firms
- Organized logistics for panels of speakers from Google, Shopify, and Facebook

Hack Western, London, ON, Canada — 2015 – 2017
Sponsorship and Transportation Lead

- Led Hack Western organizing team for both Hack Western 2015-16 and 2016-17; over two years, led volunteer recruitment, sponsorship, and transportation
- Initiated three sponsorship agreements to contribute to over $80,000 in funding and prizes,

Boston Consulting Group Case Competition, *Finalist* — 2016

- Analyzed business cases and presented recommendations in three rounds of competition, judged by Ivey faculty members and BCG consultants; placed in the top 10% of 80 teams

SunStang Solar Car Project, *Telemetry Team* — 2014-2015

- Designed and developed a telemetry system using Arduino microprocessors to collect data on critical components of the car, enabling competition in the Texas Formula Sun Grand Prix

19.3 Sankalp Hariharan

SANKALP HARIHARAN

(647) – 975 – 3282 | 3282 Paul Henderson Drive, Mississauga ON, L5M 0H3 | shariharan.hba2018@ivey.ca

EDUCATION

IVEY BUSINESS SCHOOL, WESTERN UNIVESITY **London, Ontario**
Honors Business Administration (HBA) and Software Engineering Dual Degree
- Awards: Dean's Honor List and Western's Scholarship of Excellence
- Relevant Coursework: Raising Capital in the Financial Markets, Finance, Applied Probability and Statistics for Engineers
- cGPA: 88.4%

EXPERIENCE

MEIR, HOROWITZ (MHI) CORPORATION **Toronto, Ontario**
Venture Capital, Summer Analyst Summer 2017
- Supported management in venture capital and venture debt investments on companies in the cryptocurrency industry; assisted with forecasting cash flow to estimate return on investment, built valuation models, and participated in due-diligence meetings with targets' management
- Educated General Partners (GPs) with the nuances of blockchain and cryptocurrencies through several authored internal documents and presentations

Selected Transaction Experience (ongoing)
- *$10 million Venture Debt Investment for One of Asia's Largest Cryptocurrency Companies*
  - Spearheaded an internal report on the cryptocurrency industry for a particular geography that analyzed comparable companies and helped the team better understand the technology; information was presented to the GPs as part of the initial screening process and successfully led to due diligence
  - Addressed various concerns and queries in several meetings with target's management team to further evaluate the target's business model and risk of investment
  - Constructed a levered free cash flow model by examining the target's financial statements and the provided guidance from management to estimate returns; built mechanisms in the financial model to support sensitivity and test various scenarios of debt provided; analysis was ultimately used to finalize the total investment amount
- *$5 million Capital Raising for an Initial Coin Offering (ICO)*
  - Researched market analysis for services provided by the cryptocurrency and synthesized it to create investor-ready presentations and other marketing material for potential investors; exceeded initial investment goal by $2.5 million

WESTERN INVESTMENT CLUB (WIC) **London, Ontario**
Researcher for Natural Resources and Technology, Media & Telecom November 2016 – April 2017
- Presented a sell-side stock pitch on Trecora Resources (TREC) for the Natural Resources Team
  - Gathered and analyzed information regarding the company and industry from equity research reports, company filings and other information such as futures on commodities from the Bloomberg Terminals to complete a business overview and create a DCF valuation model, built with various scenarios based on future revenue projections and prices of commodities
  - Collaborated with other analysts to formulate an investment thesis and internal analysis of the company and provided support to drafting the final deck
  - Delivered the final presentation to other researchers as well as junior and head analysts of WIC (~70 people); the pitch received a record amount of 'yes' votes from researchers/analysts and is now part of the club's $180,000 investment fund
- Compiled major industry-related events (e.g. M&A activity or IPOs), changes in economic outlook and relevant geopolitical events; presented and debated with 30-person researcher groups about its implications and consequence

INDEPENDENT INVESTING **September 2016 - present**
- Managed a portfolio of American and Canadian equity stocks; for each investment, applied the principles of value investing on target companies and performing valuation methodologies including DCF and comparable company analysis
- Located and gathered company-specific information from sources such as Thompson One, Bloomberg and FactSet
- Achieved YTD returns of ~12%; portfolio mainly consists of technology and natural resources companies

CARTE WEALTH MANAGEMENT **Toronto, Ontario**
Financial Advisor Summer 2016
- Formulated investment strategies for new clients that best matched their long-term financial goals by providing recommendations to invest in a combination of equities, bonds and mutual funds
- Independently reached out to four new clients over the summer for a total investment of over $50,000

ADDITIONAL INFORMATION

Certifications: Breaking into Wall Street Premium: covered relative and intrinsic valuation methodologies, M&A models and LBO models; Bloomberg Terminals: Bloomberg Market Concepts & Bloomberg Essentials Training
Languages: Fluent in English; Conversational proficiency in Tamil and French
Technical Skills: Object-oriented programming languages - Java & C++; Web-Development: Ember and Node.js
Interests: Learning new languages, traveling, competitive swimming, and technology

19.4 Joshua Weinstein

Josh Weinstein
524 Roselawn Avenue, Toronto, ON M5N 1J8
Tel: (647) 668-8628 Email: jweinst3@uwo.ca

EDUCATION

Ivey School of Business, Western University, London, ON, Canada 2018
Candidate for Honours Business Administration (HBA)

- Awarded 1 of 8 Leaders Scholarships (of 608 students) for exceptional potential in leadership, achievement and commitment – Jackson Family HBA Leaders Scholarship
- Awarded Ivey AEO Award of Excellence given to the top 1% of accepted Advanced Entry Opportunity applicants based on demonstrated record of achievement and passion to succeed
- Dean's Honor List (86.7% average) 2015-2016

Western University, London, ON, Canada 2018
Candidate for Bachelor of Engineering Science, Software Engineering

- Awarded 1 of 20 National Scholarships for outstanding academic performance, creative and innovative thought, achievement in extracurricular activities, and community service – President's Entrance Scholarship
- 1 of top 3 ranking students in Engineering (out of ~1600) – 2015 University of Western Ontario Faculty Association Scholarship
- Dean's Honor List (96.3% cumulative average) 2013-2014, 2014-2015, 2016-2017
- Top academically achieving varsity athlete – CIS Academic All-Canadian Award
- Purple Finger Award for outstanding spirit and contribution to the engineering community as chosen by peers

WORK EXPERIENCE

THE BOSTON CONSULTING GROUP Summer 2017
Summer Associate

- Defined the analytical approach and performed the analysis to understand the additional gross margin opportunity available for individual stores of a large national chain to launch new Stock Keeping Units more quickly. Analysis resulted in an identification of product groups for each store where specific attention should be paid to capture the largest percentage of the opportunity with the least effort.
- Developed and owned the integrated financial model which summarized impact across all BCG supported initiatives. Required collaboration with all team members to figure out timing of impacts and to identify potential double counting that need adjustments.
- Created a tool to help understand potential client pushback on the created clustering methodology. This allowed the team to better anticipate client questions and concerned and respond to them immediately.

CIBC World Markets, Toronto, ON, Canada Summer 2016
Sales and Trading Summer Analyst, Interest Rate Options Trading

- Created tools to perform principle component and regression analyses on swap rates; used to improve decision making for trading by providing a new level of analytics not previously available

CIBC World Markets, Toronto, ON, Canada Summer 2015
Sales and Trading Summer Analyst, Wealth Solutions Group, Structuring Pricing Desk

- Independently developed models to automate calculations of backtested performance for various financial products, significantly increasing accuracy and efficiency
- Directed and managed an internal technology team on the development of an improved website to display structured note information; effectively communicated business requirements and meticulously monitored work product, resulting in a timely deployment with accurate financial information

Electrical and Computer Engineering Department, Western University, London, ON, Canada Summer 2014
Research Assistant

- Independently researched the state of sensor network technology, procured components, and designed and built both hardware and software to create a working wireless sensor network
- Completed project to be used as a basis for current graduate students to conduct ongoing research

OTHER ACTIVITIES & INTERESTS

Western University Varsity Track and Field Team, *Athlete* 2013 – 2018

19.5 Tony Wu

TONY WU

175 Bamburgh Circle, Suite 707, Toronto, ON M1W3X8 | 647-668-2280 | TWu.HBA2018@ivey.ca

EDUCATION

Western University London, Canada
Candidate for Dual Degree, Honors Business Administration (HBA) and Software Engineering (B.E.Sc) *2018*

- *Cumulative GPA:* 3.92/4.00, Software Engineering Courses: 4.00/4.00 *(95.0%)*
- *Honors*: Dean's Honor List (2014, 2015, 2016), Claude Lamoureux HBA Scholarship (2015), EMCO Scholarship (2015), and General Motors Scholarship of Distinction in Engineering (2015)

Proficient in: Java (incl. Spring), Python, JavaScript (incl. MEAN stack), Testing, Git, SQL, HTML, CSS, VBA, Capital IQ

RELEVANT EXPERIENCE

Amazon Canada Inc. Toronto, Canada
Software Engineering Intern *Summer 2017*

- Improved seller experience across Amazon marketplaces by writing software that decreased the time required for sellers to estimate shipping costs for shipments that are Fulfilled By Amazon; resulted in a ten-fold improvement in estimate processing time and savings for Amazon sellers in the magnitude of tens of thousands of hours per year
- Contributed to the team's agile and lean development methodologies by making suggestions to optimize code reviews within the team and adding a workflow to alert the team when estimate processing time is above historical levels
- Presented technical specifications and design documents for the implementation and testing of the software that demonstrated how similar improvements can be made in other features of Fulfillment By Amazon

Qatalyst Partners LP San Francisco, U.S.
Investment Banking Summer Analyst *Summer 2016*

- Analyzed merger and acquisitions (M&A) opportunities by conducting valuation and due diligence to assist deal teams
- Independently drafted a full M&A pitch that was presented to partners to demonstrate learning over the summer internship; resulted in the highest merit among interns
- Worked on a potential ~$1.5Bn sale of Security Software company to a strategic partner or financial sponsor
  - Prepared comparable company analysis, market and operating analysis, parts of the merger model, and analyzed activist investor research, which resulted in participation in a strategic meeting with the CEO of company
  - Supported the deal team in soliciting deals by organizing the data room and helping put together the marketing materials that were sent to companies, which resulted in interest in acquiring the company

Alphabet Inc. (formerly Google Inc.) Kirkland, U.S.
Engineering Practicum Intern *Summer 2015*

- Created new workflow that visualized how changes in autocomplete algorithms impacted user satisfaction, which was approved to help Software Engineering (SWE) teams perform critical sanity checks of their work
- Provided useful data points for SWE teams in assessing autocomplete performance by implementing statistical analyses that allowed teams to analyze the performance improvements of their algorithms
- Presented design documents for the workflow and survey tool to SWE teams and used their feedback to update project deliverables, resulting in the seamless integration of the product with a beta launch of Google Hangouts

EXTRACURRICULAR EXPERIENCE

Western Investment Club, *Head Investment Analyst* *2015 - Present*

- Led weekly Researcher Meetings in the Technology, Media & Telecom group that discussed industry news and trends, as well as research investment pitches to add stocks to the club's $180,000 investment fund
- Engaged researchers to create comparables and discounted cash flow analyses, use financial database resources, and develop investment theses for the pitches of Microsoft Corp., Qualys, Inc. and Skyworks Solutions, Inc.

Investors of Tomorrow, *Co-President* *2014 - Present*

- Oversaw student executives in generating a multimedia curriculum and corresponding web presence to disseminate the curriculum with the goal of improving financial literacy for students
- Organized the annual conference and stock competition attended by students in eight Ontario universities, including writing the curriculum used to instruct students on investing fundamentals and presenting a Renewable Energy Workshop

Other Activities: Canadian Math Olympiad – Top 20 in 2013, Waterloo Computing Contest – Perfect Score in 2012, RCM Piano – Grade 10 with First Class Honors