UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2023-008
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 27/02/2023 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Fernando Kaway CARVALHO OTA
Born on 18 September 1984 in Santos/SP, Brazil

SECURE ARCHITECTURES FOR MOBILE FINANCIAL APPLICATIONS

Dissertation defence committee

Dr Radu STATE, dissertation supervisor
Professor, Université du Luxembourg

Dr Omar CHERKAOUI
Professor, Université du Québec à Montréal

Dr Raphael FRANK, Chairman
Professor, Université du Luxembourg

Dr Thibault CHOLEZ
Professor, Université de Lorraine

Dr Jean HILGER, Vice Chairman
Head of SnT Finnovation Hub, Université du Luxembourg

Dedicated to my family, whose unwavering support has been the cornerstone of my learning journey. To my mother, Jane, whose tireless efforts ensured I received the finest education possible. To my father, Tadashi, a beacon of motivation, inspiring me to embrace creativity. To my sister, Larissa, who opened my eyes to the diverse facets of life. To my partner, Bruna, who bolstered my spirit even in moments of self-doubt. And to my daughter, Elisa, who has shown me the most profound and fulfilling form of love one can experience.

Index

List of Figures

List of Tables

Chapter 1

Introduction

Over the last decades, the popularization of the Internet has changed the needs and ways that users interact with software, forcing a complete shift in software engineering paradigms. Indeed, more than half of the world's population is now connected to the Internet. Moreover, smartphones are now generating the majority of the Internet traffic, with more than 52% of the total data volume[1]. One direct reason is the increased usage and availability of mobile applications, namely apps. The two most known mobile application store platforms (Apple mobile application Store and Google Play), offer over 2 million apps. People depend on apps to carry out daily activities including browsing the web, purchasing or checking bank accounts. These activities have attracted attention of different sectors.

Several industries have transformed their retail channels by electing apps as the primary means of communication between clients and organizations. This forced an increase on the amount of services available on mobile applications. This breadth of services, that are usually served to the apps through Representational state transfer (REST) Application Programming Interfaces (APIs), broadens the attack surface to actors that can reach the back-end servers via broad open internet.

Recurrently, events such as data breaches[2] put the user's privacy on the spotlight. At business level, evolution of data protection regulations (e.g. GDPR in Europe) are pressuring

[1] https://www.statista.com
[2] https://www.privacyrights.org/data-breaches

service providers but there is an urge. Some previous leakages of worldwide players, such as Facebook, show that the trade-off between security and convenience seems to always end in big losses. If leaking personal conversations or pictures can be considered big issues, the problem become even more critical when using sensitive personal data.

Even a properly built secure communication channel between the mobile application and backend, using Transport Layer Security (TLS) [58] as example, can still be exploited if a potential attacker understands how to create the API requests that explores poorly built business logic. As instance, if an endpoint that checks the balance of a customer by taking an account number as parameter would be easily exploited by replacing it with account numbers of other customers. Even though proper secure software development life cycle is in place, as the range of more complex services increases, the chance of failing in building secure business logic raises as well.

In fact, APIs have recently became a major concern as regulations are obliging financial institutions to open their set of services to third party access, also known as Open Banking. This enforcement have forced a fast paced development of such APIs, exposing those institutions to new risks as they have to rapidly develop REST endpoints integrating their transactional environment.

A generic architecture of a mobile application is illustrated in Figure 1.1, presenting the interactions between the entities. The action starts with user interacting with the mobile application. When the user needs to be identified and authenticated, the most common approach is to use an external authentication service that will provide accesses tokens. Those tokens grant access to the backend server that hosts the APIs, interfacing the database that contains the user data. By using those tokens with user consent, third party applications can also hit the same backend server and consume the data in the same manner the mobile application is consuming.

This great set of interactions forces data to flow into many directions, increasing the risk of exposing sensitive data. Once the data leaves owner's possession, it is at risk while in transit, at rest and when processing. This means that the data life cycle requires proper protection to keep a high standard of privacy. In other words, security assurance has to be

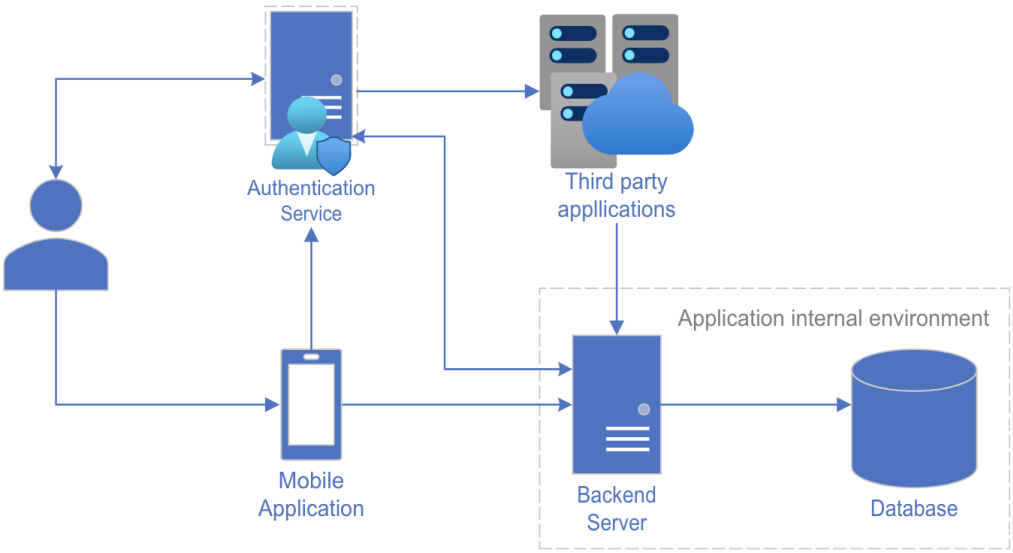


Figure 1.1: Generic mobile application ecosystem

enforced in every single communication, storage and process of every data holder and/or processor.

The communication channel between the mobile application could be considered secure if both of the parties can recognize each other. By pinning TLS certificates in the code, the mobile application can guarantee that the data is delivered to the trusted backend server but the other way around is a problem, leading to the first research question:

How can the backend server guarantee that is receiving data from a legit instance of the mobile application?

The next concern starts when the data reaches the backend server. Companies are increasingly relying on public cloud infrastructure to process the data. Thus, the risk of leaking data to those providers, through side channel attacks [111], has to be considered. Even if the data is processed on premises, this risk is also imminent from the infrastructure management staff. This raises another research question:

Can the data processing be protected from those with access to the processing platforms in the cloud?

As demonstrated before, in recent mobile applications ecosystems data is processed by third parties. Those third parties can be not only curious but hostile. For some use cases an analysis of anonymous data would be enough, but for other ones, sharing sensitive data would be required. In the latter, there is an ever growing risk of the user lose the control over his data as it might be shared many times among several entities. By knowing who, when, how and why those entities collected and processed data, the user can ensure that the ecosystem has a proper governance over his sensitive information. In order to reach this level of assurance, the following question has to be answered.

Can an user share data ensuring proper data governance?

Even though those questions are broad with many potential outcomes deriving from them, a central hypothesis can be reached from those issues: **In an open API exposure scenario, if the consumers are properly identified and their data exchange is controlled, then the providers retain a level of control that can guarantee data security and privacy.**

By leveraging recent advances on privacy enhancing technologies, the work of this thesis aims to provide solutions to those questions and support the hypothesis by raising security standards on mobile applications even if stakeholders are openly connected. To this end, a Trusted Execution Environment (TEE) available in the cloud providers is used as root of trust to extract, transform and distribute any personal sensitive data, allowing data to be analysed even anonymously when necessary. Novel methods to understand the behaviour of the customer and detect anomalies are also presented to show the feasibility of gathering knowledge without knowing the identity. Finally, by combining the TEE with Blockchain technology, a fully decentralized super mobile application is created as an ultimate goal of preserving privacy in an open highly connected environment. In summary this thesis presents the following contributions:

1. Secure channel between the mobile mobile application and a trusted execution environment: one attacker man-in-the-middle trying to retrieve information from the channel between the mobile application and an enclaved backend will face the mutual authenticated end-to-end encryption channel described in the chapter 3, published [91].

2. Interest detection of task-oriented mobile apps: the behaviour of the customer is analysed in chapter 4, published at [90], without disclosing any kind of personal identifiable information while delivering knowledge that could be used for real-time offers.

3. Anomaly detection of mobile application behaviour based on API calls: applications connecting to the backend have their behaviour analysed to detect any kind of anomaly in chapter 5, in press from a conference, indicating potential attacker to the APIs.

4. Privacy Preserving Decentralised Super App: chapter 6 presents an approach of a super mobile application in order to correctly identify, authenticate and secure third party applications, maintaining traceability and auditability of the users' data, with 2 publications: [92] and [21].

Explanation in details about each of those mechanisms are depicted according to the nature of potential attackers. The next chapter presents the state-of-the-art found during the research of those contributions, followed by the chapters demonstrating how they are achieved. In sequence, all of the contributions are discussed in light of showing how this could be applied in real world. The last chapter concludes the work and proposes future work that can extend this thesis.

Chapter 2

State-of-the-art

This chapter is divided in three sections. The first one intends to elicit the threat model, followed by an introduction of the technologies used to build the security and analysis mechanisms. The last part present all the related work found in the literature in regards to the proposed approaches.

2.1 Threat model

Financial mobile application architectures may differ depending on various factors. One fact that is usually determinant is the technologies available during its conception. As example, a recent financial technology company (Fintech) may tend to adopt more cloud technologies than a large bank that adapted legacy infrastructure to provide a customer-facing mobile application. The reasoning for the choices has to balance trade-off between cost and security versus institutional risk appetite. Therefore, covering all potential data flows is unfeasible.

In order to provide an updated reference of security mechanisms commonly placed in a modern banking mobile application, Figure 2.1 presents the data flow among its potential components. It also presents the attention hosts where issues could happen even with all the security layers to protect the main core, the Financial Processing system.

Considering that the bank system has to provide a set of Open Banking APIs, which is the case for Europe with PSD2 as instance, both mobile application and third parties

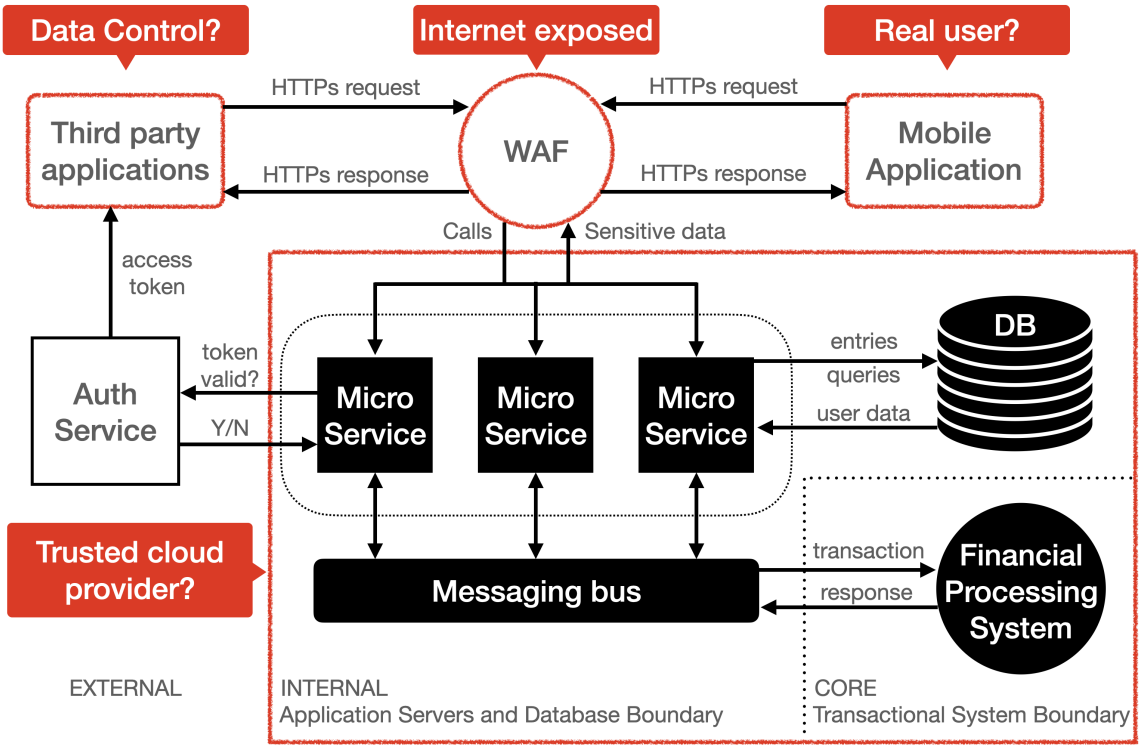


Figure 2.1: Data flow and its attention points (red)

would consume the same set of APIs. Current best practices [7] recommends that a Web Application Firewall (WAF) should be placed in front of all APIs, which becomes the entry point exposed to all internet.

WAF is a security solution that sits in front of a web application and monitors incoming traffic for malicious requests. It checks each request against a set of rules or a rule-based system to determine if the request is legitimate or if it represents an attempted attack. If the WAF determines that a request is malicious, it can block the request and prevent it from reaching the web application. WAFs can also provide other security features such as logging, intrusion detection, and blocking of known malicious IPs.

Web Application Firewall (WAF) typically includes the following features:

- Request filtering: WAFs examine each incoming request and block those that do not comply with the set of rules or that match known attack patterns.

- Common web application vulnerabilities protection: modern WAFs can protect against common vulnerabilities that are defined in the OWASP ModSecurity Core Rule Set (CRS), providing a first line of defense to the service layer .
- IP blocking: WAFs can block traffic from known malicious IP addresses.
- DDoS protection: Distributed Denial of Service (DDoS) attacks can overload a web application with a large amount of traffic. WAFs can detect and block such traffic to protect the web application.
- Encryption and SSL offloading: WAFs can handle SSL encryption and decryption, which can offload this process from the micro services.
- Logging and reporting: WAFs can log all incoming requests and generate reports that provide information on the number of blocked requests, the types of attacks, and other security-related statistics.
- Integration with other security solutions: Some WAFs can integrate with other security solutions, such as intrusion detection and prevention systems (IDPS) and security information and event management (SIEM) systems, to provide a more comprehensive security solution.

Even with all of those security mechanisms, the WAF can be freely hit by any accessible host. While origin of requests coming from third party applications can be validated by Public Key Infrastructure (PKI), methods to prove that calls are coming from the legit mobile application can be bypassed, such as hard-coded instance side certificate. The later refers to inserting a credential on the client application to be trusted by the server when sending calls. However, this credential could be easily retrieved with reverse engineering of the mobile application.

The first contribution of this work focus specifically on this problem. Chapter 3 provides a dynamic method of enrolling new instances of the mobile application, thus, legitimating the origin and encrypting calls end-to-end. It raises the wall by increasing the complexity

on counterfeiting calls to the backend, that can be deployed in the cloud protected against side-channel attacks from the cloud provider.

Notwithstanding, it is still not guaranteed that it is a real user operating the mobile application, using screen overlays in an emulator as example. The later can be used to automate attacks when the second authentication factor is a one-time password. By receiving relayed credentials collected in a fake application, a screen overlay application running in an emulator can insert this credential in the legit mobile application in near real-time, granting access to the attacker.

Granting that a real user is behind the instance of the mobile application can mitigate this risk. In this work, the efforts were directed in two ways: Section 2.3.2 presents a survey on life recognition methods for mobile application to ensure a human is operating the device; Chapter 4 describes another contribution with a data-driven approach that can be used to detect user behaviour on the server side.

None of the security mechanisms provided have a deterministic approach to guarantee perfect security of the mobile application, which by itself is an utopia. Consequently, anomalies will happen and should be detected. Chapter 5 updates the state-of-the-art of such task with a model that combines feature engineering approaches to detect deviations on the behaviour of the mobile application.

While proprietary mobile application can leverage from authentication system with liveness detection and behaviour analysis, third party applications may have to comply with certificate authorities, such as the eIDAS (electronic IDentification, Authentication and trust Services) certificates in Payment Services Directive 2 (PSD2) context.

eIDAS certificates are used to authenticate the identity of the parties involved in the transaction, such as the Payment Service Providers and the consumers. This authentication is done through the use of qualified electronic signatures and qualified certificates for electronic seals.

These certificates are issued by Qualified Trust Service Providers (QTSPs) that are recognized by the EU and are a key element of the EU's trust framework for electronic transactions. This framework allows for secure and reliable electronic transactions between EU

countries and ensures that the parties involved in the transaction are who they claim to be, but only if previous trust is centrally established, meaning that the data exchange is centrally governed.

One of the contributions of this work is the usage of Blockchain technology in a super app use case, presented in Chapter 6, decentralizing data exchange by leveraging of a distributed ledger that, when combined with Trusted Execution Environment (TEE), allows multiple applications to share and access data in a secure and auditable manner. Also the ability in providing attestation of the code shifts the trust from the entities to the code being executed as it provides transparency of data acquisition and movement.

To formally map the threat vectors, Figure 2.2 present the attack graph based on the MITRE ATT&CK framework, assessed in [50]. It is a widely recognized and commonly used tool for understanding and analyzing cyberattacks, covering both mobile and cloud environments. It provides a comprehensive, data-driven model of the tactics, techniques, and procedures used by adversaries. The assets already identified are isolated in the attack graph and its potential attack techniques are mapped based on the open research questions.

Starting from the perspective of the mobile applications, phishing and drive-by compromise, constituting social engineering technologies, aims to retrieve valid accounts. This also means that a legit user could start an attack. Along with compromising the mobile application with the techniques that varies from corrupting the binaries with reverse engineering, hooking the processes with Frida [80] and exploiting the public APIs, the attacker can inject processes by learning and sending requests directly to the WAF.

Third party applications could be compromised, therefore, their application behaviour could not be trusted. Current calls authentication mechanisms, based on the consent of the user, could also fall in phishing attacks, leading to potential malicious requests sent to the WAF, as well.

The trend of cloud also brings another potential attacker to attention: the cloud provider. With unrestricted access to the infrastructure, the cloud provider could either sniff the network to retrieve information or place side-channel attacks to try collecting data from the storage or memory at runtime.

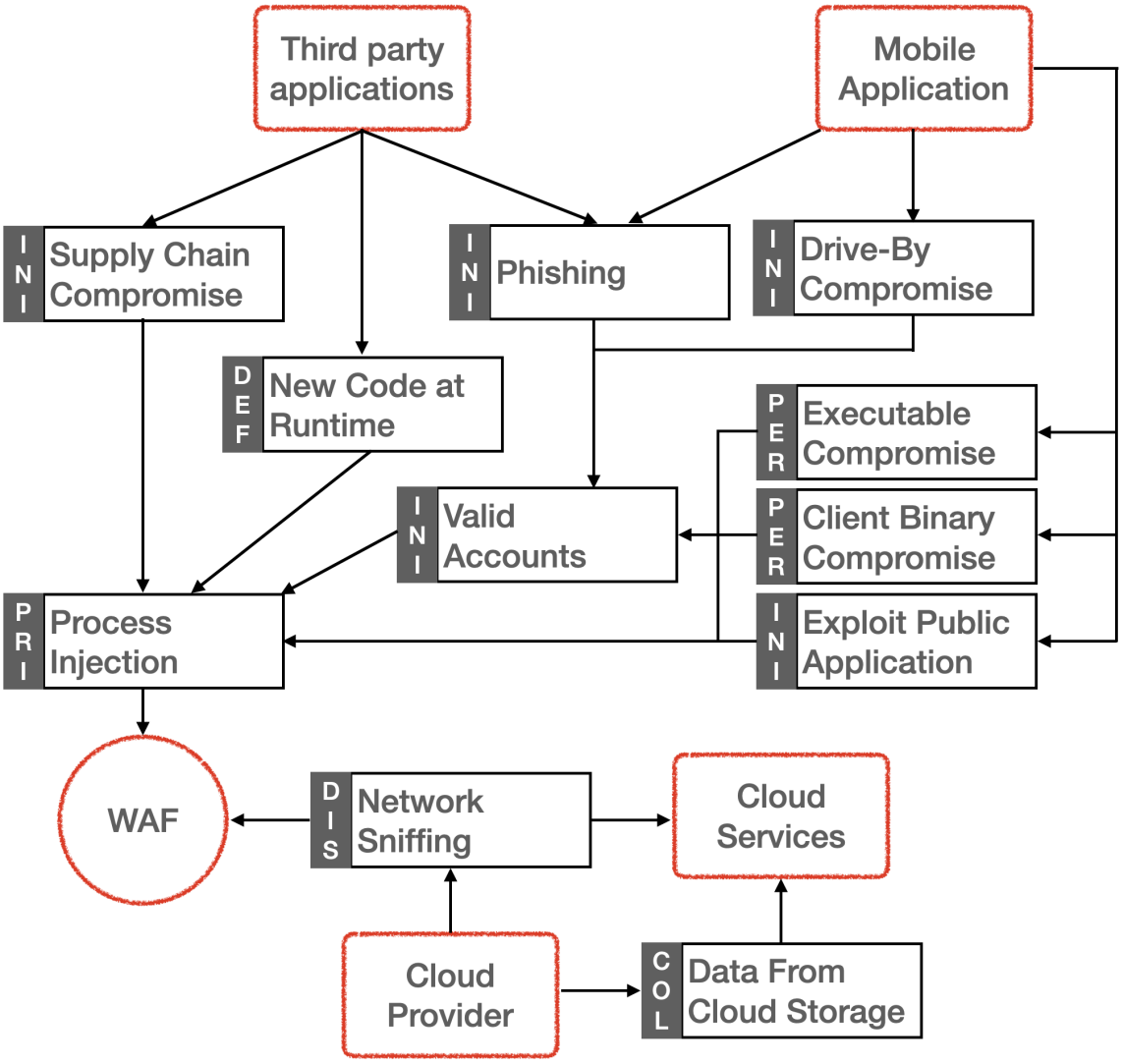


Figure 2.2: Attack graph based on MITRE ATT&CK techniques

The main focus of this work is to provide strategies to mitigate those potential attacks. The technologies that are necessary to create them are explained in sequence.

2.2 Enabling technologies

The state of the art of the technologies presented in this section allowed the design of the proposed systems. Although some of them come from more than 20 years of knowledge in the field, recent implementations and specifications provide the required knowledge to build the whole system.

2.2.1 Intel Software Guard Extensions (SGX)

Intel SGX is a trusted execution enviroment built in the brand's processors since 2015, ranging from PC up to server use. It has been widely adopted by main cloud services provider, such as Microsoft, IBM and Alibaba.

Its set of security instructions allows isolated processing of code and data in so called enclaves. The protected code to be executed into the enclaves should be partitioned and is called by a host app, the untrusted part of the application.

The design of this system counts on 3 cryptographic features available on SGX:

- Sealing: based on a fused key inside the processor, SGX has a set of instructions to derive a key in order to encrypt all the data to be stored at rest outside the enclave. SGX has some flexibility in terms of the elements that can be used to derive the key. We are interested in binding the key derivation to the UPM, AM and PEM instances so we can seal users's hash table as well as any user sensitive data that needs to persist.

- Attestation: once the enclave is instantiated, SGX allows a challenger to attest authenticity of the code running inside the instance. The aforementioned auditing process relies on this attestation process. External auditors (not SAMS owner) can challenge any SAMS components, while internal auditors (SAMS owner) can check members apps enclaves, including the source of the code as SGX provides code measurement mechanisms.

- Enclave Encrypted Paging: the fused memory of SGX is limited, usually to 128 MB. However, in case the enclave requires additional memory, SGX encrypts the portions

of RAM in the paging process. This is useful to the proposed system as authentication can be maintained in protected memory during an open session, avoiding excessive I/O operations even with increasing volume of data.

This SGX overview concentrates on features planned to be applied to our use case. A in depth explanation is provided in [30].

2.2.2 FIDO Authenticator

When the topic is authentication, FIDO specifications are the most prominent market standards. FIDO is actually an alliance of flagship companies, such as Google, Amazon and Apple. In fact, they have already implemented the two main specifications: Universal Second Factor (U2F) [110] and Universal Authentication Framework (UAF) [109]. U2F concerns a second factor of authentication, e.g. dongles or OTP keys, while UAF concentrates on bringing up a passwordless experience. In most of the cases, the whole authentication stack of FIDO is performed locally, but in our approach we intend to collect the credentials in the super app and make the perform authentication remotely in our AM.

Intel by itself provide a document describing how to implement a FIDO Authenticator in SGX [8]. Still, we did not find in the literature real experiments of this implementation.

2.2.3 Signal Messaging Protocol

The Signal protocol provides strong message encryption. It is the so called end-to-end encryption implemented by Whatsapp. It has proven security analysis [27] providing privacy, forward secrecy and cryptographic deniability, achieved by combining the following methods:

1. Key generation: a set of keys with different lifetimes are generated for each user.
2. Keys registration: those keys, except the private one, are packed in bundle and then sent to be registered in a key accumulator server.
3. Key exchange: each side of the communication retrieves the other's bundle from the server and performs a X3DH [83] shared secret key agreement based on the each

other's public keys, ending up with a secure communication session.

4. Messaging: messages are encrypted with a key derived from the shared secret and each other's bundle ephemeral key, creating a chain of messages.

New users are assumed to exist based on the first usage, which means a new fake user can be easily created with automated attacks. For this reason, it is not suitable for provisioning users of this proposal, as we want users coming from the real super app. That is why we developed the protocol in [10].

For the other message exchange between the super app and the enclave components, including the member app backend, Signal protocol should be implemented.

2.2.4 Mobile User Interface Renderers

In a super app environment, the business logic should not be implemented in the mobile app to avoid its reinstall every time a new feature is added by a member app. Instead, the mobile client should be implemented as an user interface renderer. This means that generic user interface components, like text fields or buttons, should be implemented in the client app, leaving the layout and endpoints calls triggers to be delivered in real-time by the app backend. Implementations of this user interface renderer are made easy by most mobile development frameworks, even the hybrid ones like React Native. The server side will only need to send layout files and the respective endpoints.This approach increases the network traffic, but in most of the cases, the layout files are small and should not affect performance of the app. On the other hand, this approach allows better auditing of collected data as the layout files can be checked without any risk of user data disclosure.

2.2.5 Life recognition

The use of biometrics for mobile user authentication is growing, especially in financial applications around the world. One of the biggest enablers has been the addition of sensors to smartphones, such as fingerprint and iris capture sensors. However, not all smartphones

have these sensors. Most of the time it is the operating system (OS) itself that controls the whole process of biometric recognition, imposing authentication mechanisms that reduce security in certain business models. In the case of the two sensors mentioned above, third-party applications only receive a positive response if biometrics have been enrolled. In practice, for more sensitive applications such as banking, this ends up allowing access to the account for all people who have registered the biometrics in the device, as well as the risk of modification of the OS decision response if it is compromised. The need to provide better authentication methods for those who do not have a device equipped with specific sensors for biometrics has motivated the use of widely available sensors, such as camera and microphone, for face detection, voice or even a combination of them. However, using these sensors for authentication implies effectively blocking fraud attempts, i.e. preventing an attacker trying to impersonate biometrics from gaining access to the system, a technique known as spoofing. The large-scale adoption of biometrics that do not need special sensors becomes especially complex when it comes to being able to properly detect spoofing. The best example is that of facial recognition in Android. When Google first deployed face recognition in 2011, it was possible to bypass the system just by presenting a photo of the person to the system. To prevent this kind of spoofing, user interaction recognition was implemented, asking the user to blink during the attempt to unlock the device. This new technique could also be spoofed by placing a video of the person blinking in front of the device. Given the difficulty of creating an offline system with spoofing detection, official Android distributions no longer use facial recognition for screen unlocking. The truth is that to build a robust spoofing detection system, several techniques and algorithms have to be combined. The goal is to be able to detect if the live user himself is trying to access the system. For such a task, the main approaches are:

1. User interaction: some action is requested to be performed such as moving the head or smiling;

2. Contextual information: other information extracted from the image, photo metadata or other sensors are added, such as a photo with the Eiffel Tower in the background

when the GPS point is sending location data in Brasilia;

3. Surface analysis: this technique consists in detecting texture patterns and depth on the surfaces of real images or photos, videos and even masks;

4. Multimodal biometrics: they combine multiple biometrics from different sensors.

In large scale, some of these tasks may become extremely expensive from the computational point of view, as in the example mentioned in item 2 above. In this case, trying to accurately extract objects and locations from a photo may make its implementation unfeasible. A simple example of increased complexity is distinguishing a real object from a picture of the object. This is also true for the algorithms that are chosen for each approach. For example, there are very good algorithms for detecting texture patterns using Deep Learning, but the amount of images and processing time prevent them from being implemented offline in Android screen unlocking because they would only rely on the user's photos and the smartphone's processing power. On the other hand, if the photo could be uploaded to Google's cloud, this application would become feasible. Several methods have been proposed to determine whether a biometrics replica is being presented to the sensor. However, when it comes to a restricted environment like smartphones, the ideal is to find a solution that is both discriminative enough to be considered secure and of acceptable computational cost. In the next section we will introduce the main threats when trying to model a biometric system.

2.2.6 Super App concept

A super app might be defined as an application, usually mobile first, offering a set of services from different stakeholders, namely member apps [92]. Although it is named super app, an instance of a super app is only one part of the necessary infrastructure. Actually, every member app needs to implement its business logic portion, while the super app community maintains the infrastructure.

The Figure 2.3 presents an overview of the interactions between the super app stakeholders, illustrating their roles as well as how they communicate to each other. Our goal is to

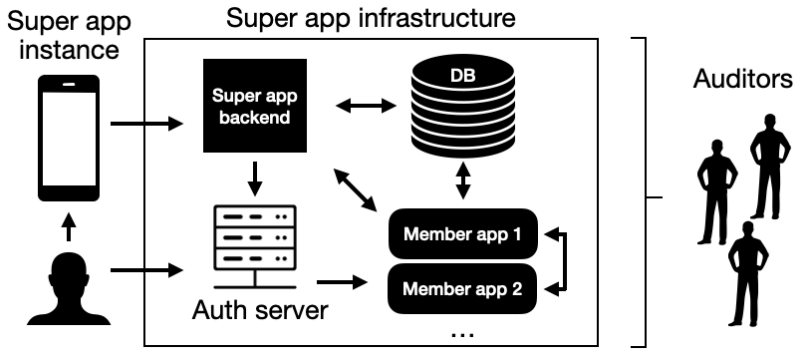


Figure 2.3: Super app interactions overview

facilitate understanding of each component in the architecture. To this end, the requirements are mapped as follows:

- User: a person in possession of a mobile device or a desktop, that downloads, installs and interacts with the app. The user must be uniquely identified, even when the device is replaced. User credentials must be linked to the entire infrastructure and not to the device or app instance.
- Super app instance: a legit instance running the app in the user's device. The instance installation should also be uniquely identifiable and consume services provided by the super app backend.
- Super app backend: An Application Programming Interface (API) layer must be provided in order to make services endpoints available to the super app instance.
- Auth server: the super app backend delegates access control to user data through an authentication / authorisation server. This server is responsible for authenticating and getting consent from users to issue tokens so a member app can access their data.
- Database: an ever-growing database of users and member apps needs to be properly optimized and protected. As the data tends to be highly heterogeneous, it is necessary to combine several approaches for different use cases.

- Member apps: each service provider will need to create its own infrastructure to connect to other members and to process the data for its own business end, including maintaining its own database. In the super app WeChat [49], the member apps are called mini-apps.

- Super app infrastructure: the whole infrastructure of the super app should provide provisioning of users and member apps. It is also desirable a software development framework to ease the deployment of the solutions. This infrastructure should also have means to be fully audited.

- Auditors: the auditing process might be done by several institutions to prevent abuse from any of the stakeholders. Internal auditors can focus on member apps, while external auditors, such as the government, might focus on the whole infrastructure.

2.2.7 Smart Contracts

Smart Contracts are immutable pieces of code executed in a distributed way using State Machine Replication (SMR) [71]. The immutable part of this statement relies on the principle that Smart Contracts are deployed using a Blockchain [87]. SMR uses a deterministic state machine to guarantee that all replicas of the executed code will have the same initial and final state, using consensus to ensure that all replicas receive requests in the same order [17]. This means that the code will be executed simultaneously in different peers and the majority of them must return the same results to guarantee that the code is reliable and has not been tampered.

In general terms, Smart Contracts are also defined as terms of agreements - or contractual clauses - between two or more parts written in the form of computer code, relying on the Blockchain technology to ensure three features: 1) *Observability* is the ability to prove the performance of some player on the contract; 2) *verifiability* verifies the contract auditability; 3) *privity and enforceability* means that the execution of the contract interests only involve the concerned parts [116].

In the context of the *Super App*, Smart Contracts are used to enforce auditable data

exchange between member apps and to manage the permissions a user gives to a member app to its data. Additionally, if a member app wants to become part of the super app, the majority of the parts involved must agree on the addition of this new peer on a token-curated registry of member apps. Token-curated registries use consensus between token holders - in this case, the member apps - to add and remove items from a list [52].

2.3 Related Work

This section presents a collection of the literature regarding the topics researched. During the research of life recognition, solutions that are already available to be used in mobile applications were found and recommendations to use in real mobile applications are given. For the other topics, the gaps presented justify a deeper research and are addressed in the next chapters.

2.3.1 Mobile Client-Server Secure Channels

The use of trusted zones, such as SGX enclaves, to process sensitive data is relatively new and the number of real use cases are not numerous. Researchers are now starting to put more efforts on this subject, driven by the recent threats and data breaches, as mentioned in session I and II. In the work presented in [106], the authors focus on analysing the biometric data from mobile devices by using enclaves. They implement and evaluate three SGX-compatible learning algorithms: Naive Bayes (NB) [63], k-Nearest Neighbour (kNN) [31] and Logistic Regression (LR) [32]. While the authors of this work concentrate their efforts on analysing the data safely, we focus on presenting a secure way of delivering the data from the mobile to the enclave. Then, in [76], the authors present a mechanism to defend against man-in-the-middle attacks by using SGX. They use sealing and attestation features to protect the user credentials. Nonetheless, this solution considers a self-owned cloud server, in private cloud environment. Next, in [102], the authors present a implicit authentication system. They propose a profile matching function by using statistics of features to accept or reject a new sample presented by a user. The data is encrypted and stored at the carrier

to avoid data leaks on the mobile side. After that, in [78], a software abstraction approach is proposed to offer trusted sensors to mobile applications. In fact, the aim is to give to the mobile application means to verify the authenticity of the data produced by the sensor. In short, the related work, mainly [106, 102, 78], is complementary to the one presented in chapter 3, once it deals with the same problem but from a different perspective.

2.3.2 Life recognition for mobile apps

One of the latest innovations in mobile facial recognition has been implemented in Apple's FaceID. Combining a regular camera, an infrared camera, a dot projector, a proximity sensor and an illumination sensor, the iPhone can capture a cloud of dots representing the user's face in 3 dimensions. This data is sent to an enclave in the Cortex A11 Bionic processor which matches it, returning the response to the app. Everything is controlled by the operating system, with third-party app developers not having access to the sensors, only the response. So users of iPhones with FaceID use the latest in mobile facial recognition. And is there a solution for other phones that do not have on-board anti-spoofing sensors? Without changing hardware it is not possible to get the same accuracy. But there are interesting alternatives in the literature, based on software, that can be implemented immediately in applications. One of the most promising transparent biometrics, which several banks already use in passwords is the typing pattern. One of the latest studies, [114] shows an accuracy of over 99% when the match is done in 1:1 and above 93% in 1:N comparisons when using deep-learning algorithms. In this study, together with the capture of the touches on the screen, the accelerometer data during typing was aggregated, which besides improving accuracy, adds an implicit life detection. Unless he is leaning on a surface, the user is holding the device while typing, generating micro movements that can be detected with the inertial sensors (accelerometer and gyroscope), differing from the data that would come from an emulator. Another caveat is that a set of data from the screen and accelerometer with a very high similarity index becomes suspect to be replicating the data, since it is practically impossible to replicate this combination of biometrics. Inertia also plays the role of life detection in the study [75], however this time combined with facial recognition. Fusing data

from the camera, the accelerometer and gyroscope, the method checks for consistency between the user's hand movement and the position of face elements while capturing a face video. The algorithm is also effective in different lighting conditions. The face reference point that presented the best accuracy was the mouth, reaching an Equal Error Rate (EER) of 7.2%. With this level of accuracy, the method can be used in commercial applications, however, it only detects fakes based on media (photo/video), i.e., masks are not detected. For mask detection, more complex algorithms must be used to try to detect patterns, such as different textures. When it comes to pattern recognition, such as for surface and texture analysis, deep-learning algorithms have shown the best results, however the training phase is computationally expensive. However, if computational power is not a hindrance, the solution presented in [108] detected spoofing on faces with accuracy higher than 95%. The test bases contained fake faces with masks and photos to prove the effectiveness of the method. The convenience of face recognition in contrast to the ease of spoofing has made life detection much studied. The article [127] presents a survey on face life detection methods. The methods found were: facial expression based, eye and mouth motion detection, optical flow based using the difference between two images, micro textures of face parts, light diffusion on the surface, multimodal and multi-attribute biometrics. The study points out as one of the biggest problems the different lighting conditions. Another problem is facial expressions that are easily deceived. Another sensor present in all smartphones is the microphone. Voice recognition, which is already present in many commercial applications, had recent articles with very interesting solutions. VoiceLive [132] takes advantage of the unique characteristics of the user's vocal system to detect life by measuring the difference in arrival time (DTC) of phonemes between stereo microphones. Replay attacks cannot achieve the unique dynamics of DTC. The method reached 99% accuracy for detection with less than 1% EER. The author claims that no additional hardware is required as stereo microphones can be emulated virtually. Another advance of the same research group of the previous article also for voice was the creation of a method of life detection through the user's mouth gesture while playing a sentence [131]. This detection is performed by listening to the reflection of a high-frequency acoustic sound while the user speaks, that is, it acts as a Doppler radar. This

method achieved similar results to the previous study and also does not require additional hardware. Explicit biometrics, such as face and voice, tend to be more easily falsifiable. More robust detection systems tend to combine other data which makes replication very complex. Thus, a line of research that has been gaining strength is the detection of user behaviour when using the smartphone. The study [37] proposes a method that combines data from app usage events, precise location, Wifi connection, accelerometer and gyroscope to trace a user's behavioural profile regarding their interaction. The model even assigned a score of 92.87% to a user but the authors do not fully trust it because the study needs to be done with more individuals. On the other hand, a previous study [107] that combined user movement data (walking or sitting), device orientation and device holding manner with typing pattern and screen touch, achieved an EER of 8.53%. The concept of continuous authentication is employed in this model, i.e. the user has their data captured constantly to assess whether it is the user who is using the device. While this seems like an ideal scenario, where an app would have greater authentication certainty coupled with the difficulty of spoofing, the potential battery drain is a concern for putting this method fully into practice.

Among the researched articles, the ones that can present the most results in a short term are the ones that use implicit behavioural biometrics. In this line, gyroscope and accelerometer sensors are the most recommended since their data are more complex to forge when a proper encrypted tunnel transmits end-to-end the data between the sensor and the server. Additionally, collecting the data from these sensors is easy to implement and they can contribute to more than one biometric. The following are suggestions for biometrics with lifetime sensing methods that can be implemented in the short term:

1. **Inertial typing pattern:** by inserting a keyboard that captures data from the screen sensor during password entry, it is possible to match a user's typing pattern. At the same time, data from the other two sensors is captured to make the same detections, avoiding emulators and replay attacks. It is also feasible to derive attributes from the set of these 3 sensors to know, for example, if the user is right or left-handed. A good challenge to be imposed to the customer is to ask him to type his social security number randomly, serving both for authentication and to detect probable people that do

not know the social security number in decor, and may even be a warning of potential opening of impostor digital accounts.

2. **Micro facial video with inertial sensors:** well-known facial recognition algorithms use video frames for matching. The gyroscope data is used to know the inclination of the device during the collection of the micro video and the accelerometer detecting small movements common to humans, like the one presented in the article [75]. This ends up reducing the access of imposters using emulators, which will have difficulty emulating data that varies constantly. In a second moment (medium to long term) it is possible to try to make a triangulation of images to reveal the depth of the face. Using the accelerometer it is possible to predict the displacement of the device, and consequently enabling the mathematical calculations to model a nose and a mouth. Even if the modelling is not perfect due to the inaccuracies of the sensors, a video or a photo would be easily detected because the calculations would reveal an almost flat surface. It is very likely that the implementation of these two methods already presents very satisfactory results. However, increasing the number of biometric and life detection methods strengthens biometric systems. In order to make a potential identity theft attack more complex, in the medium term, the following methods may be implemented:

3. **Context detection:** the goal in this case is to draw a risk profile, similar to the one presented in [56], based on GPS data, proximity of bluetooth devices, cellular network triangulation and wifi connections. Implicit life detection in this case is done by checking displacement and whether the user is close to certain devices. This analysis will also allow predicting in which context the user is inserted, for example, he is at work because the visibility of the building connections (wifi and mobile data), his colleagues' devices and GPS location provide enough data.

4. **Contact matching:** another implicit life detection can be built by making comparisons between the client's frequent calls and their contact book, as done in the user profiling of article [73]. An emulator has a high chance of not containing any phonebook

or never having made a call. Most of the time, this method is resistant to switching devices since both Android and iOS download the user's phonebook as soon as the smartphone is used for the first time, by logging in with their Google Play or Apple Store account. Software engineering for methods 3 and 4 must take into account some restrictions. The first of them is about privacy because the information that will be collected reveals much about the user, consequently infringing some regulations such as the General Data Protection Regulation (GDPR). However, there are emerging technologies (e.g., homomorphic encryption) that will allow analysis of this data without the need for the analyst to know the content. The second restriction is the amount of data. If the server has to store the address book of all clients or all visible wifi and bluetooth devices, the database may become unfeasible. On the other hand, if all the data is processed in the client application, there is a risk of battery drain and attacks that circumvent the decision system. There are ways around these problems, such as controlling the time frame of the analyses by processing part of the data on the smartphone. Another consideration is that these two methods should not be deterministic, that is, a confidence score will be assigned to contribute to the creation of alerts for transactions that deviate from normal patterns. The implementation of these 4 methods cited above will already make the attacker's life quite difficult forcing them to look for more vulnerable systems. However, valuable targets can still be focused for large scams. And the tendency of a robust authentication system is to create a high level of trust in authentications, which can be dangerous since sophisticated attacks tend to be discredited by security and risk analysts. Thus, a good plan is to continue to insert additional methods, especially for high-risk transactions. In the long term, the following suggestions should further strengthen the system:

5. **Voice recognition and inertia:** the researched articles showed that the new methods for life and spoofing detection in voice recognition have high hit levels. Moreover, the microphone is the most uniformly present hardware in smartphones. With well-known noise reduction and recognition methods [93], the other sensors detect inertia during voice capture, reducing the attack surface considerably. This method can also be

implemented during the micro video capture of method 1.

6. **Active challenges for voice recognition:** the more complex methods in articles [132] and [131] could be implemented making the system well protected against spoofing attacks. Although the papers describe well how the method was created, considerable time would be required to reconstruct the algorithms.

   A reasonable assumption for voice is that the user doing high-value transactions would be in a more controlled noise environment, including being explicitly asked to submit to more silence. In this way, voice would be a more sophisticated challenge and only used to achieve a higher degree of confidence.

7. **Ensemble Learning [133]:** the combination of models that does life detection strengthens the system once it tends to make it more complex for the attacker to forge several types of biometrics and different behaviors in which several learning models are applied. This is also a countermeasure to avoid attacks based on Generative Adversarial Nets (GANs) [53] in which attackers use artificial intelligence to generate false data in an attempt to bypass the biometric system.

2.3.3 Interest Detection

The main goal of interest detection is to find frequent sequential patterns in the session and identify the interesting ones for the business. In this sense, many researchers have focused on developing models for frequent sequential patterns. The research papers [103][13] used the Markov Chains model to identify users' navigation paths on websites. The model focused on analyzing the user's favorite web page without considering a more sophisticated analysis of user behavior. In [16], the researchers propose modeling user trails on the websites. The model obtained promising results in terms of accuracy and mean reciprocal rank, however, the model did not implement event timing modeling to help generate a non-stationary process in the analysis of random shipping lanes. In [70], they analyze the navigational click-stream data, but to social commerce platforms, which diverges from oriented-tasks app. In [62], they develop a user navigation behavior on a website model. This model fo-

cuses on visualizing web data without implementing user navigation path analysis. In [104], they present a framework for modeling the sequential data capturing pathways. In [60], they propose a model for discovering users' navigation. In both investigations, they implement a reduced sample size, affecting the analysis of user behavior and the model's scalability.

An unsupervised clustering method for analysing behavior based timed k-grams approach was proposed in [124]. Their k-grams construction is based on the event name interpolated with a category of a time interval. Then, their clusters are based on the frequencies of the k-grams in the user sessions. The approach was employed to clickstream datasets from Whisper and Renren social network apps, which contain 33 and 55 types of events, respectively. To those amount of event's types, this approach suits well, but it tends to become more computationally expensive as the k-grams construction exponentially increases when the app has a wider spectrum of events. As instance, the bank app we analysed had 311 events in the period we analysed, resulting in over a million 3-gram possibilities to count their frequencies per session, disregarded the time intervals categories. Therefore, this approach becomes unpractical to our use case in terms of processing and human comprehensible cluster visualisation.

One inspiration to our sequence discovery and subsequently graph modelling is found in [125]. In their effort to mine path in web sessions, they created the concept of a via-link to reduce the noise by removing the bigrams and 3-grams that does not reach a minimum occurrence count. To our use case, filtering off the bigrams or 3-grams that does not reach a probability threshold globally would result in removing also real events' transitions as some of them are likely to appear few times in the dataset. For example, some products that are not commonly purchased would disappear in the modelling with the global threshold proposed in this latter paper.

The main contributions of this thesis in this topic can be summarized as follows: First, a new technique is proposed for mapping the whole mobile app by defining the source node based on the user's first action. Second, a tree is built verifying all the possible routes without going through the same path determining the HUE. Finally, a cluster is created to group the behavior of the users in a behavioural perspective.

2.3.4 Application Behaviour Anomaly Detection

In recent years, research has been proposed for the detection of behavioral anomalies in API Calls including statistical techniques, machine learning and deep learning. System calls were used in several studies to detect malware in mobile devices. As this work is also trying to detect patterns in calls, the techniques for detecting patterns were inspired by this line of research.

In [79] presented a statistical pattern method based on feature extraction techniques using frequency sequences and n-grams on system call traces for anomaly detection. Using the n-gram approach by itself and calculating the statistical patterns proved not achieve high anomaly detection scores in the work, leading to many false positives and negative when applied to real world use cases. The reason is that the n-grams can only capture fixed patterns and when this is the only feature, it tends to create models with patterns that are actually noisy sequences. Our approach of calculating the levenshtein distance to the n-grams is an effort to minimize the noise by calculating the effort of transforming a sequence into another. This means that if a noisy sequence is being matched against the already extracted sequence pattern, the levenshtein distance will be smaller if the error is minimal but significant when a potential attacker is behaving completely anomalously.

A novel system based on feature selection and supervised machine-learning algorithms was proposed in [4] for detecting patterns on mobile malware. The complexity of the model affected the performance rate of the system.

A malware detection approach using the Hidden Markov Model based on partial analysis of API call sequences was presented in [112]. They did not implement data pre-processing techniques, affecting the efficiency of suspicious event detection from streams of API calls.

In [126] proposed a malware detection system based on Long Short-Term Memory (LSTM) using API call sequences. The pre-processing techniques were not implemented obtaining noise in the data.

The work of [95] presented a malware detection technique based on Deep learning using API call graph embedding. Setting the hyper-parameters of the deep learning algorithm is computationally expensive.

A malware detection task was developed in [45], modeling the malicious behaviors by API call sequence alignment and Markov chains. Only malicious behaviors common among malware belonging to the same family are considered in the API call sequence alignment algorithm.

A framework for malicious behavior detection in android apps through API calls and permission uses using Support Vector Machine (SVM) model was described in [128]. The model only identified 71% of the malware samples.

A static analysis behavioral API for malware detection using markov chain was designed in [11]. In the research they infer that the proposed system outperforms existing malware detection systems without a demonstration.

The research of [67] suggested a novel approach to detect malware using sequence alignment algorithm based on API call. The model could not detect all the malware patterns defined in that study.

Another line of research that could be compared to this one is detection of attacks using web browsing behaviour. Recent findings demonstrate the feasibility of automating feature extraction with auto encoders [121] and Word2Vec [74].

A comparison of deep learning and n-grams approach is presented in [82]. Their work also confirm our finding that is the LSTM is computationally expensive to this use case.

To the best of knowledge, the approach of this thesis in the implementation of several techniques for Behaviour Anomaly Detection using API Calls based on mobile applications is unique in the literature.

2.3.5 Super Apps

Although SGX could be replaced by a tailored Hardware Security Module (HSM), the system would not be able to run in cloud enviroments. The current most used super app WeChat was not implemented with privacy by design. This means that our approach could make a super app feasible in the face of privacy regulations, such as european GDPR.

A previous work using SGX to preserve privacy has already been proposed in [14]. In this work, the authors propose three approaches to monitor the data flow and enforce the

use-based privacy, which means disallowing any usage of data that harms privacy. In our proposal, we work with requiring user consent, as for some applications users would like to provide sensitive data in order to gain some business advantage or even to make transactions feasible. Nonetheless, we can learn from the experience of creating privacy monitors from this article to implement our member app backend auditing.

Another insightful privacy data driven work can be found in [69]. This exploratory study employs SGX to create a trustyworthy remote entity in a many-party applications environment. They aim to preserve privacy by computing certain functions across several peers. Their work implementing privacy enhancing techniques in SGX could be extended to some of the use cases where the super app would struggle, such as location-based services.

A recently published work [55], related to the AM, applied the Signal protocol to send authentication results. Instead, we plan to implement the authentication logic on the server side in order to have more control of the credentials, use more powerful computational resources if necessary, and also combine different types of telemetries.

Some super apps have been deployed worldwide, however, all of them count on a single central authority to control the whole ecosystem. As WeChat is the major deployed superapp, the majority of the contributions in the topic are related to it. including the its impact in China [**giudice**]. A description of WeChat from an outsider research perspective is presented in [49], as well as an exploratory study to elicit the gaps of such a multi use platform [25]. In another work [81], the authors propose a deployment of WeChat API in a platform as a service cloud environment to reduce development and deployment burdens. In [88], the efforts are concentrated in the software engineering perspective of a generic super app architecture. Indeed, this work is complementary to ours and might support the development of our work.

With the deployment of solutions in different industries, some new applications could leverage the rich customers' data environment provide by the super apps, such as the works with multi source credit risk modelling [97], graph based fraud detection [2], credit card fraud [1], financial inclusion [98] and income estimation [113].

Finally, the challenges of deploying a country wide super app solution in India is shown

in [66], in which the authors intend to highlight the social adoption issue and how to address them. None of these contributions are focusing on the data privacy aspects of a super app. In this sense, the work of this thesis is intended to extend them into the perspective of privacy and decentralisation.

Chapter 3

Mobile App to SGX Enclave Secure Channel

There is no doubt that biometric, implicit and continuous authentication [5], improve security of mobile applications. However it is paramount to take into consideration the threats that a potential exposition of this data might cause.

In fact, a biometric system is a good example to explain potential vulnerabilities that can arise from processing sensitive data. In general, a biometric system is composed of the following components:

- Biometric sensor: collects biometric data from users;
- Features Extractor: transforms data collected from the sensor into a features vector (biometric template) that can be interpreted by the biometric recognition algorithms;
- Matcher: compares the previously saved template with a new one, generating a score that reflects the degree of similarity between them;
- Database: stores users' biometric templates;
- Decision-making system: makes the decision to accept or not the new template presented.

Depending on how the system is built, all components may contain vulnerabilities that can end up accepting synthesized, fake or even replicated templates. In Figure 3.1 the components are illustrated, to which attacks they are subject (extracted from [1]) and their descriptions are presented below:

1. Fake biometrics: the attacker can use a cast, mask, photo, video or any other fake artifact to try to present himself to the system as a legitimate user;
2. Replicate biometrics: an attacker with control of the channel between the sensor and feature extractor can resubmit a biometric template and gain new access;
3. Bypass the extractor: if the attacker is controlling the feature extractor, he can modify all input biometrics;
4. Synthesize the feature vector: knowing how the biometric template is constructed, the attacker can synthesize vectors until the decision system accepts it.
5. Bypassing the matcher: With control of the matcher, the attacker generates recognition scores to either accept imposters or block genuine ones, or attempt to learn about the decision system to discover its acceptance threshold (minimum score required to accept the template);
6. Modify the template: in the database it is possible to modify the registered templates of users, impacting all future access attempts;
7. Intercept the channel: the channel between the database and the matcher is extremely sensitive, once the attacker in possession of this channel can send any type of template to the matcher, consequently falsifying all decisions.
8. Bypassing the decision: some systems are built providing only the answer of whether the biometrics are accepted or not. In this case, if the attacker manages to modify this response, the whole biometric system has been compromised.

The main manufacturers of smartphones have implemented the extractor, matcher, database and decision system embedded in the operational system to enable collection of biometric

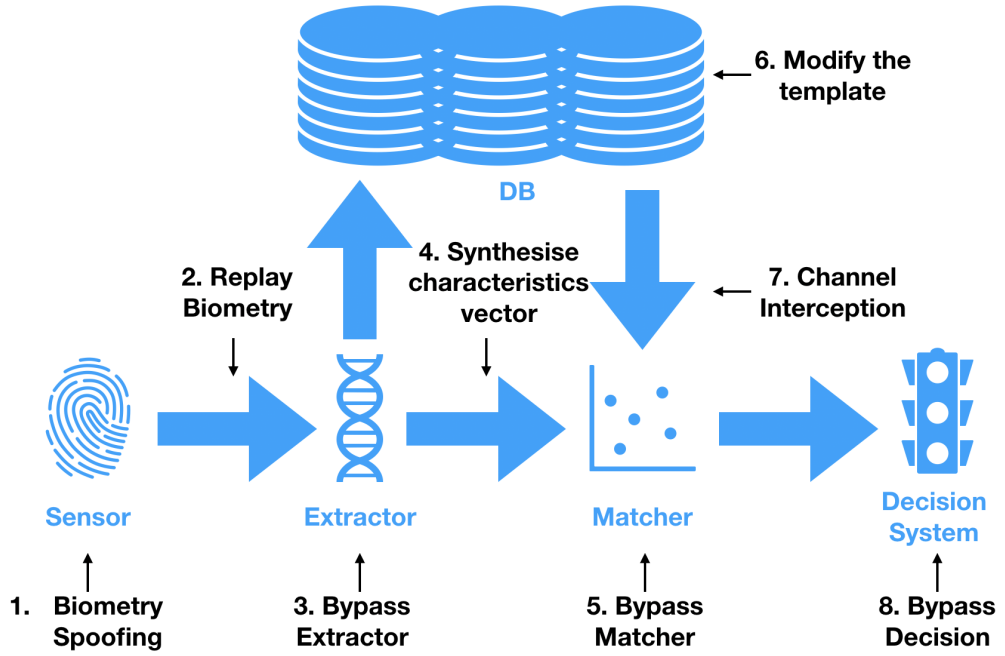


Figure 3.1: Attack surface of a biometric system

data with the hardware sensor shipped within the device. In most of the cases, the embedded authentication method works fine. However, in case that a more complex process is required, the biometric data must be transferred from the device to an external server, which raises several concerns related to potential data breaches.

By pinning the certificates into the mobile app, the developers can assure that an original instance installation is communicating with the proper server. However, the server has no guarantee that incoming requests are from the original app. An attacker capable of modifying the source code can repack the app to redirect connections to a malicious server or even start to send forged requests directly to the server APIs. While distributing a fake app is an heavy attack requiring skilled engineers, sending direct requests to the APIs will only demand an attacker to extract the certificate from the mobile app. In the former scenario, securing the app distribution by hashing and signing the binary is possible but require user awareness. In the later scenario, once the attacker knows how to generate the credentials payloads, he can start to feed the system with fake biometric templates until the system

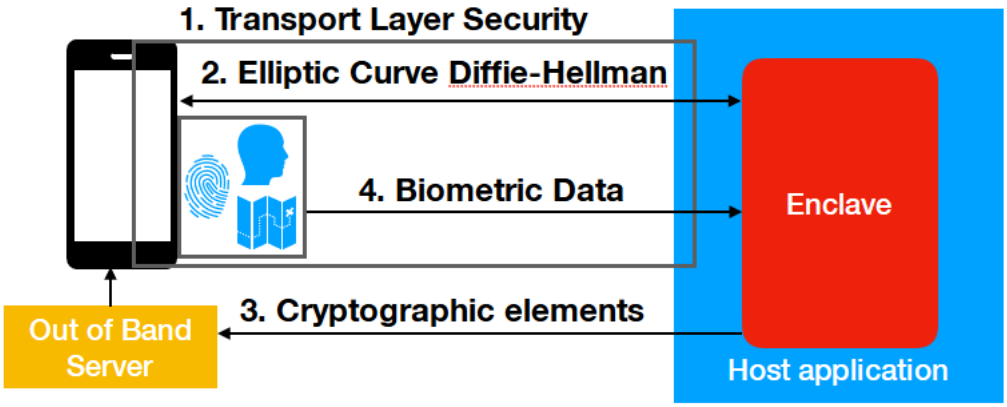


Figure 3.2: Architecture overview to transfer and process data

starts to accept it as a original one. The attack surface presented in Figure 3.1 [1]. Explicitly, the protocol presented in this chapter mitigates attacks targeting the communication channel and the client app side. The protocol also allows transfer of sensitive data (e.g. biometric data) from a mobile device to a trusted execution environment, in which the data can be processed securely, see Figure 3.2. In this context, the main security concerns are listed:

- Replay attack: An attacker with channel control between the sensor and feature extractor can resubmit a biometric template and gain new access;
- Bypassing the Extractor: If the attacker is controlling the feature extractor, it can modify any incoming biometrics;
- Synthesise vector of characteristics: by knowing how the biometric template is constructed, the attacker can synthesise the vectors to fool the *decision system.*

Furthermore, when it comes to biometric systems, vulnerabilities that could potentially be explored by adversarial machine learning [59] must be considere as the matcher is usually based on training models. What all these attacks have in common is that they are initiated from the data source because, like most of today web applications, it is expected that the certificate authority of the service provider is not compromised. Indeed, certificate based mechanisms are usually provided to ensure that the requests are coming from an

[1] This Figure is an adaptation from [3].

attested third party. However when they come directly from the mobile apps it is not possible to assure the origin of the requests. This mainly happens because the certificate, usually pinned in the app code, can be retrieved by a reverse engineering process. In possession of the certificate, an attacker can build his own app to send malicious requests to the bank API. With the enforcement of the Payments Services Directive 2 (PSD2), banks are opening their API to third party. This threat is serious because it impacts security measure on HTTP requests. PSD2 shall increase the HTTP requests coming from different third parties applications making more difficult to decide in a short time frame if the request is legitimate.

Then, if the data is processed in a non-reliable cloud environment, it is also necessary to protect the data from the provider in order to prevent unauthorized disclosure and tampering. Therefore, in order to provide a trusted execution environment, the Intel Software Guard Extensions (SGX) was chosen as the root of trust for the proposed protocol. In the server side, all operations are processed inside a SGX enclave to guarantee confidentiality and integrity.

The rest of this chapter is organized as follows. Next Section presents the threat model that is considered to develop our protocol. Section 6.1 presents the architecture and describes the protocol itself (subsection 3.1.3). In Section 3.2 a discussion of the implementation of the protocol and its results related to the overhead of using SGX are presented.

3.1 Architecture

Our main goal is to create a secure tunnel between the mobile app, more specifically the biometric sensor and the SGX enclave. As a SGX enclave can only be called by a host application, all the data coming from the mobile is forwarded to the enclave, but with an end-to-end encryption.

3.1.1 Software Guard Extensions (SGX)

Intel SGX isolates an execution environment by creating an abstraction called enclave with a protected physical memory. Enclave code and data have their access controlled by the CPU.

Thus, a host application can create an enclave but cannot access its data unless explicitly passed through function variables. Developers need to separate the untrusted code from the one processed in the Trusted Computing Base (TCB) [101]. This property allows running code even when the environment is not fully trusted such as a cloud provider.

Unfortunately, this benefit comes at a cost of memory and computation. The cache memory available to the TCB is 128 MB and part of it is used by itself, leaving only around 90 MB to the developers. The newer versions of SGX allows paging between cache and the main memory but to maintain the security, the pages are all encrypted, increasing the computation of the enclave code. A detailed explanation about SGX operation can be found in [30].

3.1.2 Properties

In order to cover the threats stated in the beginning of this chapter, the architecture offers the following properties:

1. Prevention to replay attacks: as the data leaves the application encrypted with an one time key, an attacker simply re-posting the message will not succeed.

2. Block upload of synthetic data (see Fig. 2 item 4): by closing a secure channel using an out-of-band server, it is unfeasible to insert synthetic data without tampering the mobile app and consequently modifying its signature.

3. Forward secrecy: even a future breach of SGX would block a Man in The Middle (MiTM) attacker of seeing previous data stored flowing through the channels or even inside the cloud provider.

3.1.3 Protocol

The Figure 3.3 presents a sequence diagram of the proposed protocol. A detailed description is provided below:

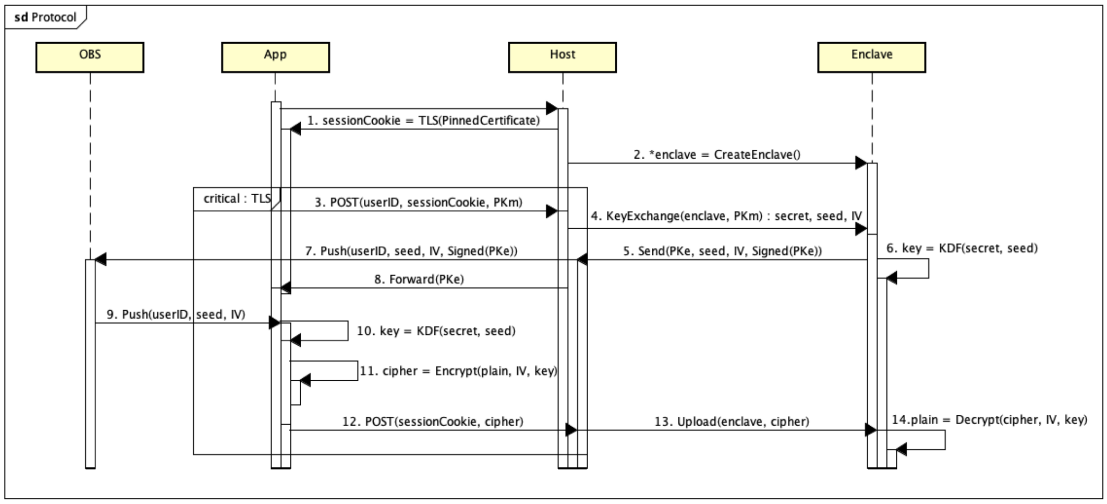


Figure 3.3: Protocol Sequence Diagram

- Session cookie establishment (step 1): The protocol starts by establishing a TLS channel from the mobile app to the host app. This procedure should result in a session cookie, allowing the host app to identify the following requests coming from the mobile app. By pinning the host certificate in its code, the mobile app guarantee the communication with the known host.

- Mobile app and enclave key agreement (steps 3, 4, 5 and 8): A shared secret is reached by performing a Elliptic-curve Diffie–Hellman (ECDH) [120] between the mobile app and directly the enclave by changing the public keys (PKm and PKe respectively). For auditing purposes or in case the mobile app developer does not trust the host app provider, an enclave remote attestation can be included to certify that the mobile app is communicating with the right enclave.

- Out of band cryptographic elements for encryption mode (steps 5, 7 and 9): The enclave generated initialization vector (IV) and seed for the KDF are sent through an out of band server (OBS) along with their enclave signed signatures.

- Key Derivation Function (KDF) (steps 6 and 10) The key to be used for encryption shall be derived on both sides combining the shared secret agreed on ECDH and the seed

coming from the OBS to generate the secret key (SK).

- Encryption of data (steps 11 to 14): This last step delivers the encrypted data to the enclave. For this purpose, the mobile app encrypts the data using symmetric encryption with the SK and the IV. The generated cipher is POSTed to the host app that uploads it to the enclave. The enclave is already in possession of the cryptographic elements necessary to decrypt.

In our implementation, a hardcoded seed and IV are used (see Section 3.2 for more details). Nevertheless, potential out of band servers are listed:

OTP devices: One Time Password would fit for generating the randomness to derive the cryptographic elements such as the seed for KDF. Unfortunately, this is already susceptible to social engineering, in special for bank apps, as it relies on the generated password that can be leaked by the user.

SMS: Cryptographic elements could be sent through SMS, nevertheless, countless attacks to this infrastructure have and are still being conducted all over the world as some financial institutions insist to send transaction authentication numbers (TAN) through this channel. The attacks include intercepting SMS at the Mobile Network Operator (MNO), device SMS leakage [86] and mobile phone line kidnapping. The latter consists on faking IDs of the owner and making the MNO to transfer the phone line to the fake customer, a very common attack in developing countries, such as Brazil.

ATM: Banking apps can count on their ATMs to agree on a first key or the first seed and/or IV. However, this would bind those cryptographic elements to an instance of the app installed on the device. If the user changes its device or even re-install the mobile app, then the process would need to be repeated. This reduces the user experience, but on the other hand provides high security mechanism.

Push notifications: In case of Android app, the IV can be sent through the Firebase push notification. As Firebase can validate the mobile app signature before delivering the message, an MiTM attacker would not receive the cryptographic elements necessary for the KDF and encryption steps. Therefore, the attacker will not be able to send data to the

enclave and this implicitly guarantees that the enclave will receive it from the mobile app. In terms of performance, Google claims to deliver 95% of the messages within 250 ms. Apple users would also count on the same sort of mechanism.

3.2 Experiments

In this section the experiments are presented to assess the overhead of our SGX enclave implementation over three different encryption modes (i.e., CBC, ECB and GCM). On the server side, the experiments have been conducted on the Confidential Computing platform of Azure providing SGX support. An 3.7GHz Intel XEON E-2176G backs these machines along with 8GB RAM running on Ubuntu 16.04. On the client side, and Android app was created and installed on Google Pixel XL with Android 9. The measurements are averaged on 10 runs.

3.2.1 The implementation

A host and an enclave applications were developed in C, using the Open Enclave framework. All the required cryptographic functions, namely ECDH, AES, ECB, CBC and GCM are included in the MBEDTLS library shipped with the framework.

In the host application a TLS 1.2 socket listens to the connection coming from the client. Once the TLS tunnel is created, the host application receives the first POST request containing a public key from the client to begin the ECDH. Then the host application starts the first servlet that creates the enclave and call the enclave key generation function. Upon completion of the function, the enclave public key is forwarded to the client. Meanwhile, the enclave completes its part of the ECDH and stores in its memory the secret. This secret is then used by the enclave when the host application calls the KDF, which consisting in applying SHA-512 8192 times along with a seed. For the tests the seed is hardcoded, however, that should go through the out-of-band channel along with the IV to be used in the encryption part. The source of random numbers for cryptographic elements shipped in the MBEDTLS

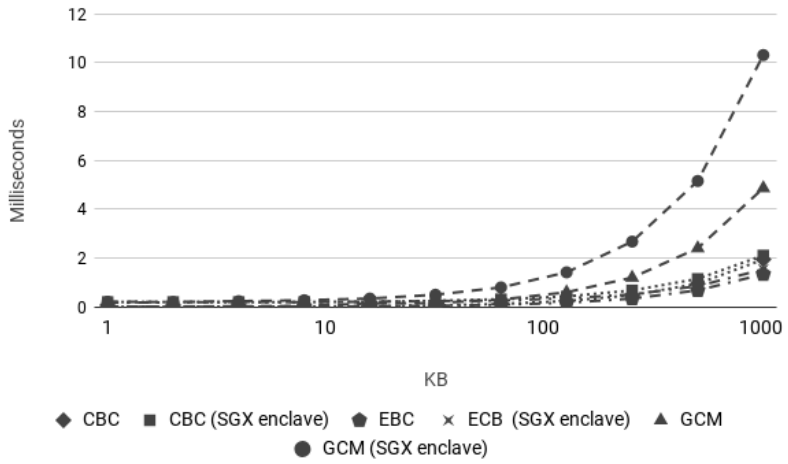


Figure 3.4: Execution of CBC, EBC and GCM Encryption with and without SGX enclave implementation.

version of the openenclave is insecure[2]. Instead, the function *sgx_read_rand* is used to call SGX Random Number Generator (RNG).

After answering the first POST with the PKm, the host application calls the KDF function to generate and expand the secret and the seed to the 256 bits SK. This implementation order allows the client app to proceed with their part of the key exchange without waiting for the enclave KDF processing. The enclave is kept alive until the host application receives the second POST containing the cipher, so that the secret never leaves the enclave. Finally, the host application calls the decryption function with the cipher. For testing purposes, the AES encryption and decryption were implemented in operation modes ECB, CBC and GCM. In AES/GCM three versions of the function are implemented: static, optimized and dynamic. The static receives two arguments, the plain text and the cipher text with a fixed length of 1 MB, while the optimized only receives one buffer, reducing the TCB to at most this 1 MB less. The dynamic allocates memory to an unique buffer.

Some operations modes do not accept to use the same buffer for the plain and cipher texts, nevertheless, as the encryption and decryption are applied for each 16 bytes, the unique buffer passed from the host application to the enclave is replaced block by block. By

[2]https://github.com/microsoft/openenclave/blob/master/3rdparty/ mbedtls/mbedtls/yotta/data/README.md

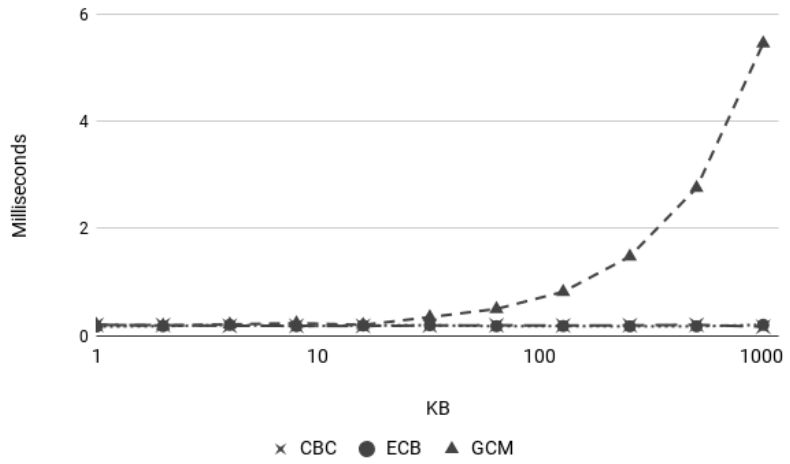


Figure 3.5: SGX implementation overhead

using the Native Development Kit (NDK) provided by Android, the client app implemented the same cryptographic functions in C from the MBEDTLS. Both POST messages (Key Exchange and Cipher Upload) were implemented as asynchronous tasks to allow the Android app to call them upon demand. In the mobile side, the SecureRandom class is used with a strong RNG provided by Android to generate the ECDH seed[3].

3.2.2 Results

The figure 3.4 shows performances for encryption in the three operation modes of AES in the enclave and directly in the host application. It is possible to see that the SGX implementation always impose an overhead to the execution time. The overhead is constant for ECB and CBC cipher modes as seen in figure 3.5, but not for GCM. It was tested up to 1 MB as our use case is biometric templates and only for encryption as any major differences for decryption has not been observed. In any case, the overhead in encryption was never bigger than 6 ms per MB.

The overhead for creating the enclave was also tested, key exchange and a Key Derivation Function (KDF). The results presented in Figure 3.6 can be all taken into account when

[3]https://developer.android.com/reference/java/security/SecureRandom

using a SGX enclave to do a ECDH to reach a cryptographic key. In practical terms the KDF time is negligible on the server side, as its computation can be processed while waiting for the second POST. The key exchange function (item 4 of Fig. 3.3), namely, generating the random seed and calculating the shared secret using the public key from the mobile, has also a negligible overhead of less than 5 ms. The only considerable overhead is the enclave creation that takes an average of 50 ms, which can not be considered an issue for practical use.

3.2.3 Discussion

In this section, details of the potential vulnerabilities caused by compromised components in our proposed architecture are discussed.

The host application added enclave life-cycle functions to the servlet with TLS of the host application implemented in C. This first implementation of the servlet maintain the enclave state between key exchange and data upload requests. In order to make it stateless, an enclave binded function can seal the key and retrieve it when the ciphered data comes. However, this can lead to break the forward secrecy in case of an attacker saving the history in possession of a compromised sealing key.

The three cipher modes tested are common in mobile apps. The assumption was that a more complex cipher mode would increase the overhead of using SGX. However, the results showed that only the GCM would increase the overhead exponentially. By all means it is highly recommended to use a Authenticated Encryption with Associated Data (AEAD) mode provided by GCM over CBC [100] if the amount of data allows as GCM tends to rocket for large files. In comparison to the insecure ECB mode [100], the enclave overhead of using CBC is quite small even if the file increases too much. Therefore, developers should not choose to implement ECB instead CBC in any situation of this protocol as there are no considerable performance penalties. In addition, it is important to consider the possibility of each of the components present in the entire process being compromised. To this analysis, the most secure scenario using AES in GCM mode and the OBS as a push notification service are considered.

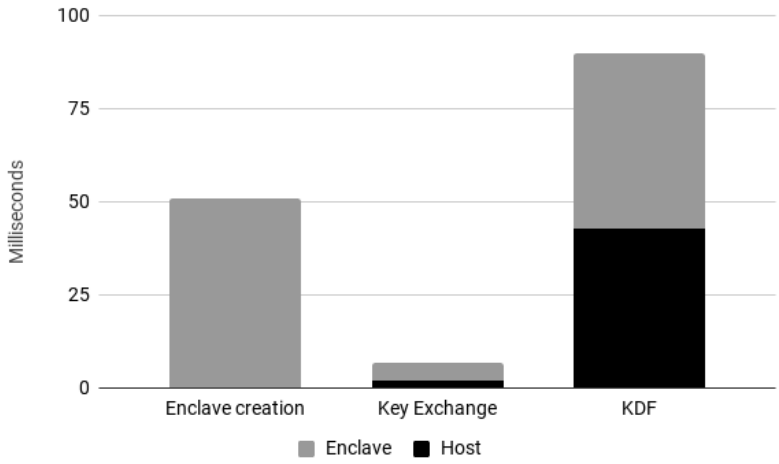


Figure 3.6: Key exchange overhead

- Compromised Mobile App: any tampering in mobile application code would modify its signature, disallowing the mobile application to request an user push notification ID. Therefore, by only attacking the mobile app, the attacker will not be able to deliver a message containing the seed and IV (item 9 - Figure 3.3) to an original instance of the app. So when the attacker build the cipher, the enclave will not decipher it correctly, the tag checking method of GCM would fail and the message can be safely discarded. Additionally, to act as MITM, the attacker would need to modify the pinned certificate in the code and redirect the connection to his server, consequently modifying the signature as well. Moreover, the distribution of the modified app would require social engineering techniques.

- Compromised Device: jailbreak or rooted devices poses as a challenge to major mobile apps. However in this case, it is not only a matter of owning the device but also the push notification services built in the operational systems of the Android and iOS. This would allow the attacker to retrieve the seed and IV for the encryption. Old versions of those services had abuses reported [39], but the new ones apparently remain intact.

- Compromised Host application: this piece of software is a natural MITM in the environment as forwards all the messages between all the other players. Yet, if a key not

generated by the enclave is sent by the host application during step 8 (see Figure 3.3), its signature cannot be forged if the mobile app is remotely attesting the enclave sign key, preventing the creation of the tunnel with the mobile app. This is efficient in guaranteeing the privacy of user's data. Unfortunately, it does not protect from the host application from uploading fake data to the enclave since it can exchange keys directly with the enclave. This can be mitigated by having the enclave checking the signature of the host app.

- Compromised Enclave: the enclave is the root of trust. In other words, if the enclave is corrupted during runtime, the whole setup fails. However, if a breach that allows to recover the keys of SGX is founded, the previous communications are not compromised as the shared key that encrypts sensitive data is immediately discarded after use upon enclave destruction.

- Compromised Out of Band Server: an attacker controlling the OBS can deny delivery of the messages and make the services unavailable.

Only by exploiting combined different infrastructures, that are nowadays reliable, an attack would succeed. For instance, if external attacker objective is uploading fake/synthetic data to the enclave, he would need to modify the mobile application, rooting/jailbreaking the device and bypassing push notification services of main mobile operational systems, which has not been done in the latest versions of the latter. In case the attacker is targeting to intercept data from the users, he would require to thrive in all those aforementioned three attacks inside the target user's device. In short, those attacks are unfeasible in practice using the current technology of the components implemented. Depending on the specifications of the Hardware Secured Model (HSM), the latter can replace the enclave, in other words, our protocol can be extended to current major banking infrastructure.

Chapter 4

Event-Driven Interest Detection for Task-Oriented Mobile Apps

Understanding the behavior of users when browsing mobile apps has increased interest in recent years to identify relevant products and make personalized offers [48]. This growing importance has driven focus of research on the analysis of task-oriented apps. This latter type of apps are defined to be when a user has a prior motivation in mind when starting a session [96], such as using the bank app to execute a wire transfer.

By analyzing navigation steps of the users' click events, the final user action can be identified, what is defined as High Utility Event (HUE) [43]. As instance, a user willing to check his bank balance has to navigate through the app until reaching the screen that presents his account. In this case, the HUE is the click event that contains the account balance. Click events consist of a series of ordered events triggered by user interactions when using the mobile app [18].

In this sense, many researchers have proposed various techniques to mine HUEs focused on finding the right candidates to the high utility (importance)[118]. However, the main challenge of those investigations was to find the correct sequence of the user's navigation steps to obtain the target action.

In this chapter, a novel approach proposed combining Markov Chain and graph theory

techniques to detect users' interest in HUEs a task-oriented mobile app. This approach mainly consists of mapping the whole mobile app by defining the source node based on the first event when the user starts the mobile app. The algorithm verifies all the possible routes, building a tree where the last nodes will be the events with high utility obtaining the target action or events of interest to the user. In the end, clusters are created to highlight the behavior of the users and the sessions by identifying the target action or the product of interest to the user, such as a purchase or a loan.

Problem Statement

Given an events' sequence of a mobile app, how to score the likelihood the user was to execute a certain target action in a session. As instance, a customer of a banking mobile application started to simulate a loan but did not submit a request. In this example, our goal is to identify how close the user was to finish the loan application.

For small apps, a list of target actions, namely HUEs, can be created by domain experts. However, more complex apps can have many candidates for target events, in other words, there can be several relevant actions that has to be filtered in order to provide real behavior knowledge. Additionally, complex apps from bigger enterprises are frequently changing, adding extra layer of manual mapping HUEs and its navigation paths. Usually, the search space is larger as the navigation events tend to appear in higher proportion than the targets.

Besides finding the right HUEs candidates, it is also necessary to detect the chain of events that lead to the execution of the target. By finding this sequence, it is possible to detect the intention of the user when he is navigating in the app in direction towards a certain action. Therefore, the sequential patterns of the app are also investigated. By merging those research topics, the term High Utility Event Mining is reached.

4.1 The Dataset

The analysis was performed on a click event dataset from an European retail bank app. Each button of the app is uniquely named representing click events that are captured through a

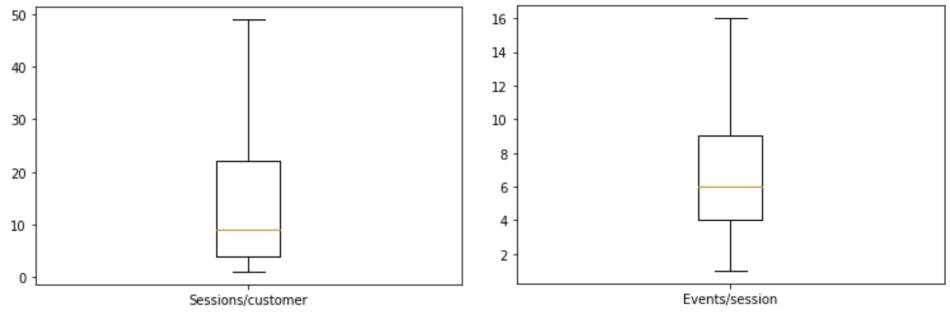


Figure 4.1: (a) Boxplot of sessions per customer (b) Boxplot of events per session

specific API HTTP call. This call is triggered at every click during user navigation to register the events with a timestamp and linked to an user and session identifiers.

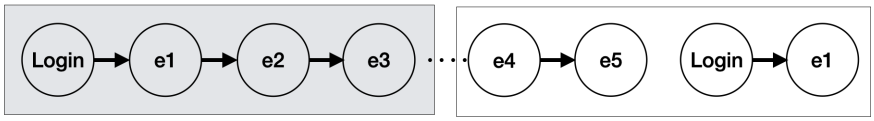
The dataset contains approximately 3.8 million sessions from over 210 thousand different users in a data collection from 31 sequential days from 2021. Those sessions have around 28 million events among 311 unique ones. Figure 4.1 (a) shows how many sessions a single customer started in the period and (b) presents how many events a single session usually have. For the purpose of keeping the confidentiality agreement with the institution, the names of the real events were replaced by ’e’ plus a number ranging from 0 to 311.

4.1.1 Dataset Issues

An important part of the modelling is data cleansing to avoid analysis on noisy data caused by the occurrence of the following errors. The strategies for addressing those issues are presented in the next sections.

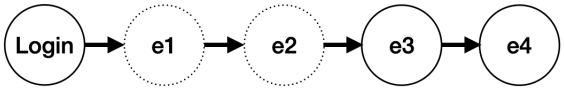
Late Event Registration due to Disconnections

The user can lose the internet connection and continue using the mobile app. In the bank app analysed, the strategy is sending the events when the user login again. In this case, the events from the previous session are registered under the current session id, resulting in a incorrect chain of events, as illustrated in Figure 4.2.



Session	Sequence
S01	Login,e1,e2,e3
S02	e4,e5,Login,e1

Figure 4.2: Late event registration error



Session	Sequence
S01	Login,e3,Logout

Figure 4.3: Losing events error

Lost Event Registration

There are some cases when the events are lost in the data pipeline. The reason for this varies from problems in the mobile app when sending the requests up to issues on the data flow from the servers to the bank mainframes. In any case, the dataset could contain samples of sequences with missing events as shown in Figure 4.3.

Insufficient Samples for Modelling

In a fully data-driven approach as ours, all the events and its transitions are taken from samples on the dataset. Consequently, if some of the events are not hit or the transitions from one event to another does not have any sample, the model is under represented as in Figure 4.4.

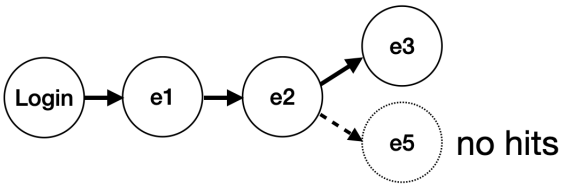


Figure 4.4: Lack of event's hits for modelling

4.2 Methodology

By combining techniques from graph theory, Markov chains, high utility pattern mining, cumulative distribution function scoring and clustering, the approach below is capable of detecting users' interest in a certain event during their navigation in a task-oriented mobile app. Figure 4.5 depicts the whole processing pipeline of our approach described in the following subsections.

4.2.1 Event Definitions

To the use case, events are the clicks on the bank app that are captured via API HTTP requests. Firstly, definitions and notations are given to be used throughout this chapter, and then more specific details about events are provided.

An event e is a labeled click register generated during the navigation of a user in the app. Let $E = \{e_1, e_2, \ldots, e_n\}$ be the set of events.
$S((uid_i, sid_j), T_s, T_e)$ denotes the click events sequence of the user uid_i in the session sid_j between the starting time T_s and the ending time T_e where $T_s < T_e$. $uid_i \in UID$ where

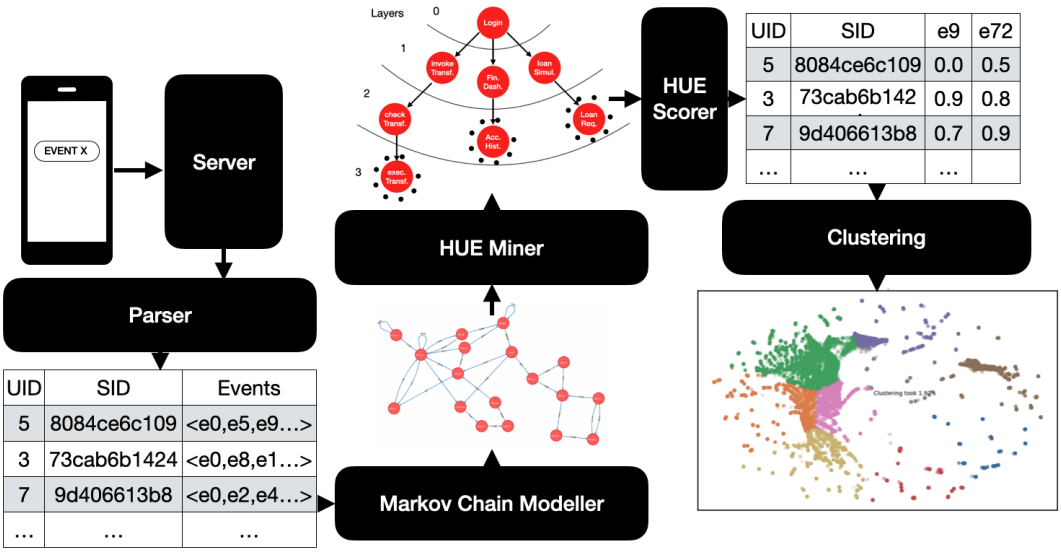


Figure 4.5: Overview of the processing pipeline

$UID = \{uid_1, uid_2, \ldots, uid_k\}$ is the set of users and k is the number of users.
$sid_j \in SID$ where $SID = \{(sid_1, T_{s_1}, T_{e_1}), (sid_2, T_{s_2}, T_{e_2}), \ldots, (sid_l, T_{s_l}, T_{e_l})\}$ is the set of sessions.
The click events sequence $S =< (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) >$, where $e_i \in E$, and $T_s \geq t_i \leq T_e$ such that $t_{i+1} > t_i$.

Table 4.1 provides an example of the dataset. In practice, the length click events sequences can be very long, and can reach up to a hundred of events per session. In order to mine and characterize causal relations between events, in the next step we introduce the notion of event graph model based on Markov chain as an intuitive graph representation for events.

4.2.2 Markov Chain Modeller

The model is based on *n-gram* sequence. An *n-gram* is a contiguous sequence of n events from a given click event sequence. The intuition of the *n-gram* is that instead of computing the probability of an event given its entire history of events, the transition probability of a n set of events occurring in sequence is computed.

Thus, to calculate those transitions, bigrams are extracted from the sequences to compute the conditional probability of original event e_o targeting the event e_t. In other words, it is an approximation with the probability $P(e_o \rightarrow e_t)$.

Given the events sequence $S_{i,j} =< (e_{1_{i,j}}, t_{1_{i,j}}), (e_{2_{i,j}}, t_{2_{i,j}}), \ldots, (e_{ni,j}, t_{ni,j}) >$ of any user i in any session j, the bigram of the sequence is computed. To compute a particular bigram probability of an event given a previous events, the count of the bigram $C(e_o e_t)$ is computed and normalized with the sum of all the bigrams that share the same first events:

User ID	Session ID	Click events sequence
U01	S01	<e1, e2, e3, e5, e6>
U02	S02	<e1, e2, e5>
U02	S03	<e1, e3, e5, e7, e3, e5>

Table 4.1: Click events dataset.

$$P(e_o \rightarrow e_t) = \frac{C(e_o e_t)}{\sum C(e_o)} \tag{4.1}$$

Let the following click events sequences of any user and any session from Table 4.1: $< e_1, e_2, e_3, e_5, e_6 >, < e_1, e_2, e_5 >, < e_1, e_3, e_5, e_7, e_3, e_5 >$. Here are the calculations for some of the bigram probabilities from these sequences.

$P(e_1 \rightarrow e_2) = \frac{2}{3} = 0.66 \mid P(e_2 \rightarrow e_3) = \frac{1}{2} = 0.5 \mid P(e_3 \rightarrow e_5) = \frac{3}{3} = 1$

A Markov Chain is constructed from multiple successive events and their occurrence conditional probability. In other way, it is a directed graph that represents successive transitions between events click events sequences through the sessions and users.

Specifically, each vertex in the graph represents an event e_i and each edge (e_i, e_j) indicates that there is a continuous bigram between e_i and e_j. The Markov Chain over a set of type of events V is a weighted directed graph $G = (Vertex, Edge, \gamma)$:

- $Vertex$ is the set of events from E ($Vertex = E$).
- $Edge$ is a set of edges in G. Let e_u and e_v be be two events in $Vertex$. There is an edge $(e_u, e_v) \in Edge$ if and only if there exists a probability with a dependency rule: $e_u \xrightarrow{l_{e_u,e_v}} e_v$.
- γ is the function from formula 4.1 that assigns a probability for each edge (e_u, e_v).

The current transitions between events represent an order of the click events and the behavior of users when they use the app, but also represents the errors presented in Section 4.1.1. In order to reduce the potential noises caused by losing events registration presented in Figure 4.3, some of the edges have to be removed. The first strategy for this removal is to exclude when they do not match a minimum transition probability.

Given the weighted directed graph $G = (Vertex, Edge, \gamma)$, and a minimum threshold $min\_support$, the filtered Markov Chain is a weighted directed graph

$G_{Reduced} = (Vertex_{Reduced}, Edge_{Reduced}, \beta)$, where:

- $Vertex_{Reduced}$ is the subset of events from E ($Vertex_{Reduced} \subseteq Vertex$).

MinSup Threshold > 0.05

Origin	Destination	Counts	Prob.
Login	e2	70	0.7
Login	e3	28	0.28
Login	e7	2	0.02
e2	e6	30	0.6
e2	e7	20	0.4
e3	e7	20	1

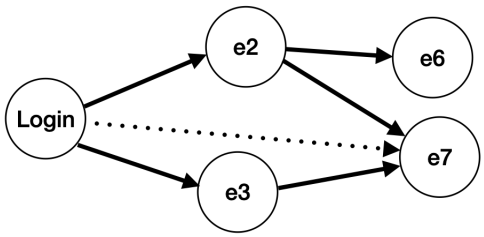


Figure 4.6: Markov Chain Noise Reduction

- $Edge_{Reduced}$ is the subset of edges in $G_{Reduced}$ where $Edge_{Reduced} \subseteq Edge$.
- β is a function that assigns for each edge $(e_u, e_v) \in Edge_{Reduced}$ the probability l'_{e_u,e_v} where $l'_{e_u,e_v} = l_{e_u,e_v} > min\_support$.

In in Figure 4.6, the minimum support threshold is set to 0.05 as an example. The transition probability from the Login to the event e7 is below this level, therefore, it is removed from the model.

4.2.3 High Utitily Event (HUE) Miner

Mining the high utility events can be complex in an app that has many events. Mostly because the definition of high utility varies depending on the desired outcomes, as instance, if it is business oriented, then product purchases would be relevant, or in a cybersecurity perspective, a password exchange event draw more attention.

Notwithstanding, task-oriented apps are likely to have navigation steps until reaching the event to execute the desired action. After executing this action, the user is usually taken back to the main menu or logout the app. If the app is designed in this fashion, consequently, it is possible to mine the HUE in a broad range by assuming the following:

Assumption : In a task-oriented app, HUEs are final states in unique sequences of events.

In this sense, by traversing the graph, the sequences of events can be outlined. However, finding the final states is challenging as it is hard to define the narrowing events of the

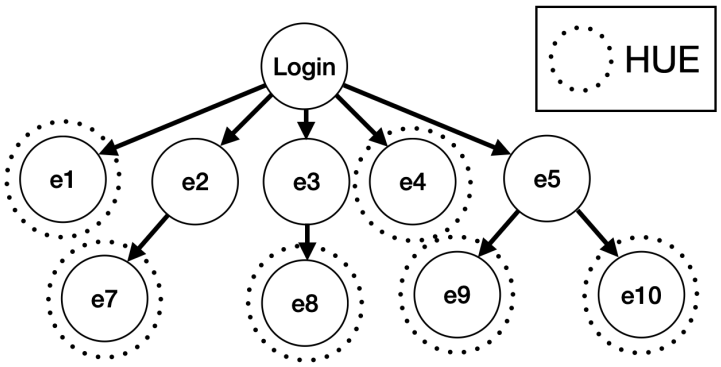


Figure 4.7: DFS-based HUE mining

sequence in an unsupervised way. Intuitively, it could assumed that the events before a logout or a return to the main menu could be the candidates for our HUEs. Unfortunately, this assumption fails because the user can logout at any part of the app or return the steps in the sequence until returning to the main menu. Therefore, an algorithm based on this intuition would lead to many imprecise HUEs candidates.

The solution is to search the graph from a source node, which is the first event when a user start the mobile app, and check on all the possible path are the nodes that can be reached without revisiting a node. Fortunately, the algorithms Depth-first search (DFS) [117] and Breadth-first search (BFS) [19] are techniques that can be employed to this end. Both algorithms traverses the graph and transform it into a tree containing the paths from the root node, the difference is that DFS tries to reach the deepest path while the BFS visit the node neighborhood before moving to the next depth level. To the use case, when a part of the app has multiple options, BFS tends to construct a layered tree where part of those leaves will not have descendants, consequently, outputting wrong candidates for HUEs. On the other hand, DFS group those options in the same level and constructs a tree where the leaves are the final states. By searching the filtered Markov Chain from the login event, DFS outputs a tree where the nodes with out degree equal 0 are our HUEs, as shown in Figure 4.7.

[t] Path Traversal Markov Chain Construction **Input:** A collection of app session traces S, η minimum support for links and κ minimum support for via-links

Output: Path Traversal Markov Chain [1]

unigram_dict ← *dict()*

link_dict ← *dict()*

via_link_dict ← *dict()*

s in S $length(s) < 3$ **break;** $i = 0; i < s.size() - 2; i++$ $v1 \leftarrow s[i]$ 1st vertex $v2 \leftarrow s[i+1]$ 2nd vertex $v3 \leftarrow s[i+2]$ 3rd vertex $v1 != v2$ *unigram_dict[v1]*$+= 1$ *link_dict[(v1,v2)]*$+= 1$ create new key in dict or increase by 1 $v3 != v2$ *via_link_dict[(v1,v2,v3)]*$+= 1$

(v1,v2,v3) in $via\_link\_dict.keys()$ *via_link_dict.get((v1,v2,v3)* / *link_dict.get((v1,v2)* $> \kappa$ *link_dict.get((v1,v2)* / *unigram_dict.get(v1)* $> \eta$ $G.setEdge(v1, v2)$ *link_dict.get((v2,v3)* / *unigram_dict.get(v2)* $> \eta$ $G.setEdge(v2, v3)$

$v = G.vertex()$ $!v.isConnected()$ $G.removeVertex(v)$

G

DFS would output a perfect list of HUEs if none of the events is lost during the collection. The cleaning process was done by the minimum support from the bigrams. However, a more frequent ”dirty” transition would still persist in the database if their occurrence is above the threshold. Therefore, another strategy was adopted by creating a path traversal Markov Chain. The path traversal Markov Chain is built by designing a new model of transition analysis, which incrementally creates links (bigrams) and via-links (trigrams), where the edges are weighted according to the relationship between connected nodes. In other words, the co-occurrence of nodes in a session, through a sequence of a mobile app events where a subsequence of size 2 and 3 is considered a link and a via-link of a session, respectively. By setting an additional minimum support for the via-links, the Algorithm (1) will remove the misplaced transitions as double event loss occurrence is less likely to happen then a single loss. The Algorithm (1) presents the graph construction and cleaning as a preprocessing phase of the DFS algorithm in order to discover the data-driven HUEs.

4.2.4 HUE Scorer

For HUE scorer computation, the Markov Chain is considered as a weighted directed graph (G, w) where $G = (V, E)$ is a directed graph, V is the set of events in the Markov Chain, and E is a set of edges in G where the edge are the connections between events in the Markov

Chain. $w : E \to \mathbb{R}_{\geq 0}$ is a non negative weight function. The weight represent the probability between two events in the Markov Chain.

The scoring method is derived from the definition of interest into a HUE. The closer the user is to hit a certain HUE in a session, the more interested he is into executing this action. To score the proximity, we used the Dijkstra algorithm [34] to calculate the distance of all the events to the HUEs in the Markov Chain.

This distance, weighted by the transition probability, represents the score of interest into an HUE. As the Dijkstra algorithm measures the distance to the target node, a lower score means a higher interest into the HUE.

Algorithm 4.2.4 represents the computation of the shortest distances of the events to the HUE. It is based on the Dijkstra algorithm on the weighted graph. The inputs are the HUEs and the weighed directed graph constructed from the Markov chain. The algorithm computes the shortest distances from each vertices to the HUE. The output is matrix of weighted distances to the HUEs.

[t] Distance of all events to the HUE. **Input:** (G, w) and v. (G, w) an edge-weighted graph (G, w) where $G = (V, E)$ is a graph and $w : E \to \mathbb{R}_{\geq 0}$ is a non negative weight function. v is the the target vertex. **Output:** $dist(u)$ the distance from u to v where v is the HUE. [1] $dist(v) \leftarrow 0$ and $dist(u) \leftarrow +\infty$ / $u \neq v$ Queue Q of all vertices in G using $dist$ as the key. $Q \neq \emptyset$ $u \leftarrow min\{dist(x)|x \in Q\}$ Remove $u \in Q$ z adjacent to u and in Q $dist(u) + w((u, z)) < dist(z)$ $dist(z) \leftarrow dist(u) + w(u, z)$ update $z \in Q$ dist

To achieve a score of interest a session in relation to a certain HUE, every session has to be mapped from the event to its distance. The minimum value of this map represents the closest the session was to the HUE.

4.2.5 Clustering

Clustering the sessions or users based directly on the distances will output improper clusters as bigger the distance is, the lower is his interest to the HUE. Hence, an inversion step is required before running the clustering algorithms. The final score can be calculate in a simple way by applying MinMax function in $score = 1 - MinMax(dist(u))$.

To decide over the clustering algorithms, the size of the app has to be considered. If the app is too big with many HUEs, the amount of cluster tends to rocket as the possibilities are 2 to the HUEs power. Normally, the domain experts are not looking to extract groups of all the mixed HUEs, instead, they should select a group of correlated HUEs.

Considering this and the most common clustering algorithms K-Means and DBSCAN, for this type of data, DBSCAN seems to be better as it is not required to specify the number of clusters, but K-Means will perform better if the amount of HUEs to be clustered is larger [22].

4.3 Experiments

4.3.1 Environment

The experiments were performed in a cluster with 4 nodes, 128 cores and 1 TB of RAM, running Spark [130]. The parser and the n-grams extraction were implemented in Scala 2.11. The clustering, algorithms 1 and 2 were coded in Python 3.

4.3.2 Parsing

The first effort of the work was parsing the API calls to extract the events and their timestamps. The calls persisting in HDFS RDDs consist of ”user id, session id, call name, timestamp of the call and message” fields. UID and SID can be retrieved directly by querying the table. However, the message fields from the calls that registers the events are the ones we are interested. By filtering those calls, we extract the messages' fields persisted in JSON files that are parsed to fetch the labels and click timestamp from the events, resulting in 2 new fields. Then, the resulting filtered dataset is grouped by the user and session ids. The events with the same user and session id are consolidated in a vector ordered by their timestamps. To fix the date issue 4.2 from the section 4.1.1, the first part of the events vector, trimmed at the login event, are concatenated to the previous session in the call timestamp order. The resulting parsed dataset with the columns of Table 4.1.

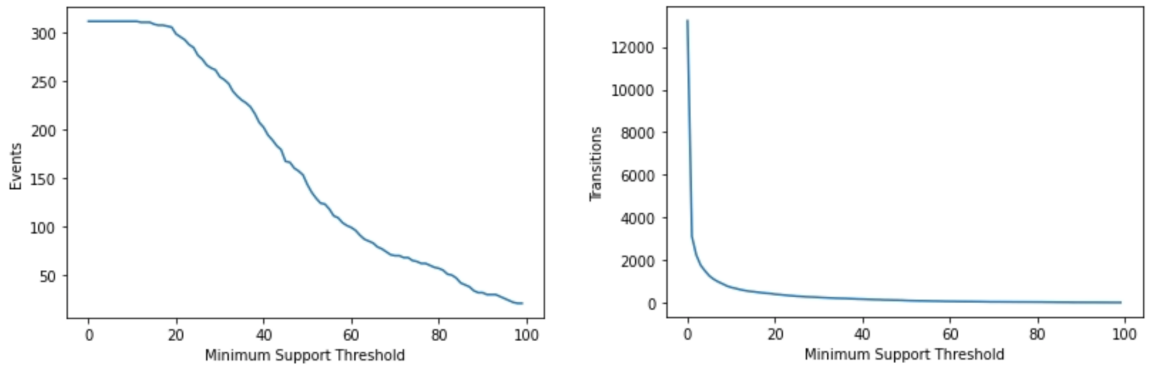


Figure 4.8: Threshold distribution

4.3.3 Modelling the app

By iterating over the vectors of events per session to extract the bi-grams and calculating the transitions probabilities with the formula provided in Section 4.2.2, a ”dirty” Markov Chain could be plotted. To clean the noisy transitions, a minimum support threshold is set based on their probabilities. However, as the threshold increases, events and transitions start to be lost, as seen in Figure 4.8. This poses a challenge on how to calibrate the threshold without eliminating the real transitions. It was expected that a visual indication would guide the estimation of the threshold.

At first glance on the events perspective, it is possible to see that 15% is a maximum threshold in order not to ban events from the model. But taking this 15% means that if the app has a menu with more than 10 possibilities, then at least one of this transitions would be less than 10%.

Another approach would be trying the elbow method [65] for the transitions. Even though, there is an elbow in the right Figure 4.8, the precise calibration is not yet straight forward. Actually, to determine this threshold, we had bank experts mapping the path for 7 HUEs, ranging from infrequent to frequent ones. By sliding the threshold in the range of the elbow, we stopped when we start to lose real transitions from the login to the HUE. The losing tendency is bigger to the events that are more infrequent as the transitions probabilities are lower until reaching those events. Finally, we reach a threshold of 2.75%, which seems to

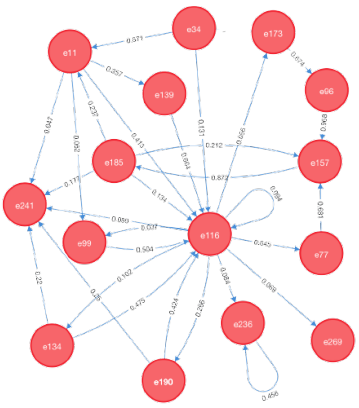


Figure 4.9: Frequent Markov Chain

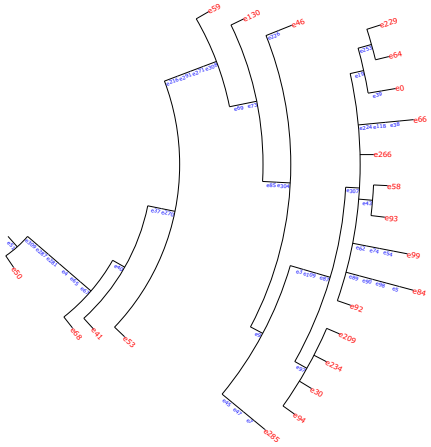


Figure 4.10: Mined HUE subtree

be the middle point of the elbow, and then the Markov Chain can be plotted, as the sample Figure 4.9

4.3.4 Mining the HUEs

The challenge of this task is again finding a minimum support threshold, but this time for the via-links required by Algorithm 1, besides the threshold already found. For this task, the via-links mapped were the 7 HUEs selected by the experts. By identifying the undesired via-links and their probabilities on those HUEs, we reached the second threshold of 18.6% that is able to remove them. After running the algorithm and applying the output to DFS, 58 HUEs were discovered on top of the 312 events.

For the sake of fitting the output of the algorithm with the DFS in this chapter, Figure 4.10 illustrates only a subtree containing 42 navigation events leading to 22 HUEs, representing approximately one sixth of the whole tree. Notice that the navigation events are not ordered by their ocurrence when in the same edge, but, it means that those are the multiple clicking options at a certain screen, such as choosing a type of loan.

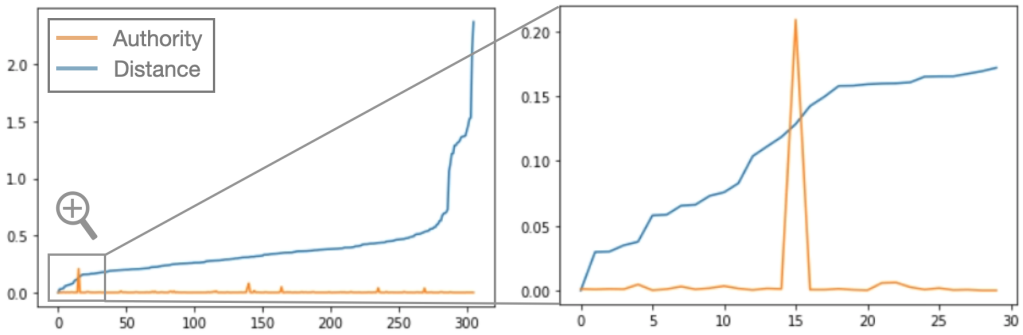


Figure 4.11: HUE distances vs HUE authority

4.3.5 Scoring HUEs

The HUEs are then submitted to the Algorithm 2, outputting a matrix with the weighted distance to all the events to the HUEs. In sequence, every column becomes a dictionary to translate the sessions events into the calculated distances. The minimum distance of each HUE for every session row becomes a new column, generating a frequency matrix for all the UIDs and SIDs.

This distance can be used as a score to highlight the interest of a user in a HUE. To this end, the sessions that have distances below a certain level for a given HUE are the ones that shows interest and when this distance is 0, the user has actually demanded the HUE. In the experiments, it was observed that the authority score from the hits algorithm [68] provides indication for determining the upper boundary for the HUE distance. In Figure 4.11, it is possible to visualise the maximum distance for personal loan interest that can be set when the authority score raises drastically.

By setting this boundary, the session rows with distance above the level can be filtered off, resulting in a dataset that represents all the sessions interested in a certain HUE. Figure 4.12 illustrates the distribution of the interest and the actual demand for one of the products in the period we analysed.

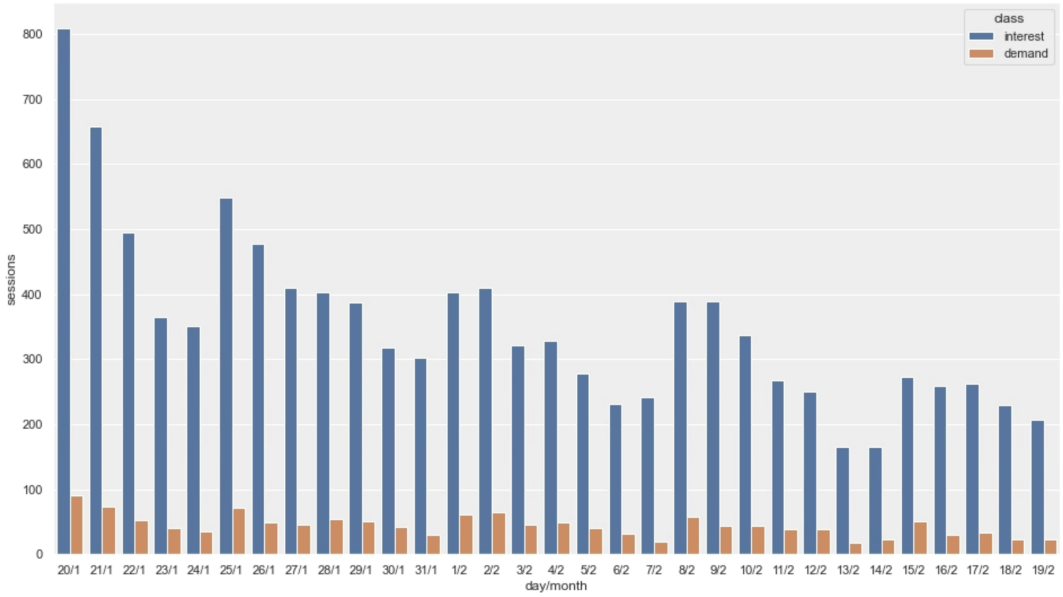


Figure 4.12: Specific product interest and demand

4.3.6 Clustering by HUEs

The clustering approach aims to identify the behavior of the customers for marketing purposes on the period analysed. The scores of the 7 previously selected HUEs were analysed for this task. In order not to disclose any information about the bank's customers, the numbers or the HUEs of the clusters are not presented. Still, to ease readers understanding of the clusters, the 7 HUEs were divided in 2 groups: (1) investment (i.e. stock market operation) with 4 products and (2) expenses with 3 products (i.e. loan).

First, the sessions of the user were grouped, keeping the maximum value of each HUE score. Second, Principal Component Analysis [94] reduced the dimensionality of the matrix. In sequence, 7 HUEs were submitted to the K-Means to try to identify the products. The labels of K-Means are used by a Decision Tree Classifier to identify which HUEs are the most significant to each cluster. Finally, DBSCAN was performed to try to extract the total amount of clusters.

Figure 4.13(a) presents the output of K-Means, where it is possible to see a clear separation between the investment and expenses products. In opposition, the outcome of DBSCAN

of Figure 4.13(b) does not gives any visual information. Yet, DBSCAN generated 98 clusters on top of $2^7 = 128$ potential HUEs combinations.

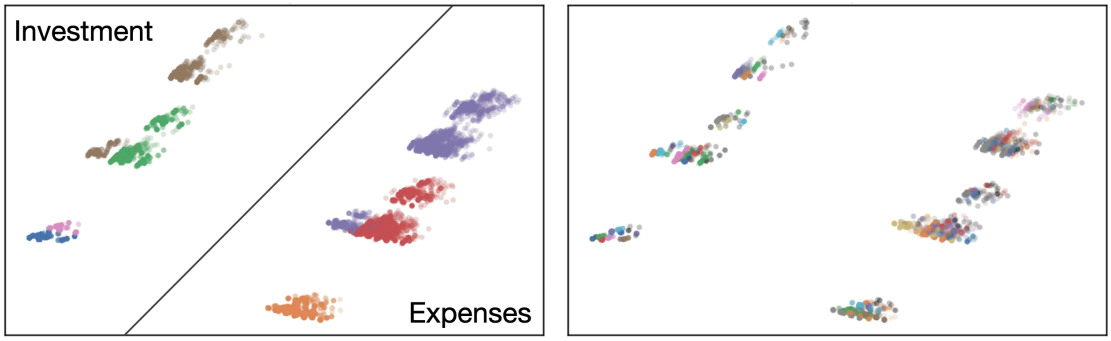


Figure 4.13: Clusters found by (a) K-Means and (b) DBSCAN

Comparing the methods, K-Means is more human explainable, while, DBSCAN resulted in better product agglutination, allowing visualizing the combination of products among the customers.

4.4 Limitations

The main bottleneck in this work is the scoring method with time complexity $O(|E|+|V|log|V|)$ [12]. The use of random or grid search does not scale well and the data size, while the influence of η and κ hyper-parameters is undoubtedly important and there is no guarantee to explore the hyper-parameter space without putting human-in-the-loop, experts.

On the other hand, computing clustering accuracy is a limitation as to perform this computation there is a need to have the bank experts labelling all the app events.

Chapter 5

Mobile Application Behaviour Anomaly Detection based on API Calls

Attackers trying to abuse the backend APIs have to understand the behaviour of the respective mobile application prior to their attempts. Some tools in the market have the capability to actively eavesdrop the TLS [33] channel by creating independent connections between the mobile application and the tool, and between the tool and the back-end server. The next step would be trying to create fake requests departing from the tool to the back-end. However, if they fail to recreate the behaviour of the application, the approach try to detect anomalies by taking advantage of the following:

Assumption: For every click event of the app there is a set of API calls that are performed to exchange data with the back-end server [95]. Those API calls respect an order that is bound to the logic the app was created. In other words, every event will trigger a set of ordered calls to the server that should be always the same if the app logic does not change [129].

Thus, any interference to this assumption characterises an anomalous user session. The main focus is to create an automatised process to detect those anomalies, that can be

classified as application failures, data losses or malicious attempts to access directly the APIs [54]. It is out of the scope of this work to classify the types of anomalies.

Every call, that is properly authenticated and reaches the server, receives a token to be presented by the app in the next call, thus, a chain of calls is created in a session that will be interrupted if this token verification fails. This means that a mistake in this chain is automatically detected by the bank system as an anomaly. Therefore, in order to abuse from this mechanism, an attacker, inside the Transport Layer Security (TLS) tunnel using Burp Suite [20] as instance, has to create a first legit call and receive this first token, that can be exchanged in further calls. However, if this attacker is 'studying' the system, he will repeat or even make calls in a different order than the coded logic of the app [15]. Therefore, one of the main goals of this work is to identify sessions that could be attackers abusing the APIs of the bank.

UID	SID	Timestamp	Call	Data
UA1	S01	07.05.2022-18:39.383668	POST /login	auth_data
UB2	S02	07.05.2022-18:43.687463	POST /login	auth_data
UA1	S01	07.05.2022-18:43.687596	GET /accounts	token: xxx1
UA1	S01	07.05.2022-18:47.383668	POST /event	loginButton

UID	SID	Timestamp	Call
UA1	S01	07.05.2022-18:39.383668	POST /login
UB2	S02	07.05.2022-18:43.687463	POST /login
UA1	S01	07.05.2022-18:43.687596	GET /accounts
UA1	S01	07.05.2022-18:47.383668	loginButton

Figure 5.1: Event registry call replacement by event name

5.1 The Dataset

The experiments were performed on a dataset from a mobile application (app) from an European retail bank. The dataset consists of API calls logs from requests made from the app to the bank back-end server. Each log data point contains an id a user identifier uid_i, a session identifier sid_j, a timestamp ts_t from the moment the event reached the server, the name of the call c_k and a data field df_l, where $uid_i \in UID$ which is a set of user identifiers of the customer database, $sid_j \in SID$ which is a set of session identifiers, $ts_t \in ts$ which is a set of timestamps, and $c_k \in C$ which is a set of calls. A specific call is used by the app to send click events through the data field $df_l \in DF$ which is a set of type of events.

This specific call was replaced by the name of the event and its column eliminated, resulting in a dataset similar to the illustrated in Figure 5.1. The reason for this replacement is to capture patterns that correlate the click events with the calls that are performed to execute the task desired by the user.

The dataset is formed of 40 days and over 4.6 million users' sessions from 2021. The sessions contains a mixture of 544 calls and events. To respect the privacy of the bank, the real name of the calls was substituted by the letter c_x followed by a random number $x \in \mathbb{N}$. The real events' names by the letter e_y followed by another random number $y \in \mathbb{N}$, resulting in 838 different combinations that uniquely identify either a call or an event.

In some circumstances, the calls and events are not stored in the logs. The reasons for this range from issues with the mobile app while submitting the calls to problems with data transfer from servers to the bank mainframes. In any case, the dataset may include examples of sequences that include missing calls.

Another issue that should be considered when implementing a full data-driven approach is that at least one sample of all calls and events should appear in a dataset. If the model built has not seen on click event in the training set, consequently, the evaluation of the session containing this unknown event will be automatically classified as anomalous. Therefore, it is necessary to train in a wide part of the dataset.

The APIs are made available to the bank's mobile and desktop application. The dataset does not include the desktop sessions. It is only formed by mobile and sessions that are not identified neither as mobile or desktop. Those unidentified sessions is the ground truth and classified as anomalous sessions to train the machine learning algorithms. The anomalous sessions are unbalanced representing around 1.3 % of the dataset.

5.2 Methodology

The approach consists of 5 steps to

1. transform and group the sequence of calls in every session

2. generate the features using approaches to capture different characteristics of the sequences
3. normalizing the features
4. balancing the data
5. feeding machine learning models to classify the sessions as anomalous or not.

This section describes the 5 steps of the pipeline illustrated in Figure 5.2.

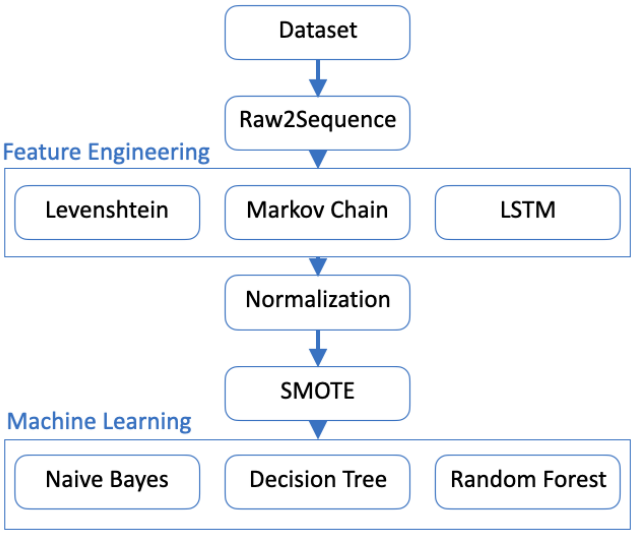


Figure 5.2: Pipeline

5.2.1 Data transformation: Raw2Sequence

The first step is to transform the data structures into a set of sequences. In fact, the sessions were grouped by the uid_i and sid_j, and merged with the calls and events ordered by the timestamp. This transformation reduces the size of the dataset by reducing user and session identifiers redundancies. From this stage on, the timestamps are eliminated as they were only necessary for ordering the sequence of calls, reducing even more the dataset.

The sequences show the dependencies among calls which define the behavior of a user when connecting to an account. Let $C = \{c_1, c_2, \ldots, c_n\}$ be the ordered list of calls, a

sequence S_i is built from a unique uid_i and sid_j, $uid_i \in \bigcup_{c_k \in C} uid_k$ and $sid_j \in \bigcup_{c_l \in C} sid_l$ such that $S_i =< v_i^0, v_i^1, \ldots, v_i^m >$ where the value $v_i^k \in C$ corresponds to the key (uid_i, sid_j) of the call c_k. This process is illustrated in Figure 5.3. By transforming the rows of calls into a sequence, it is possible to exactly track all the ordered steps of a session.

UID	SID	Timestamp	Call
UA1	S01	07.05.2022-18:39.383668	POST /login
UB2	S02	07.05.2022-18:43.687463	POST /login
UA1	S01	07.05.2022-18:43.687596	GET /accounts
UA1	S01	07.05.2022-18:47.383668	loginButton

UID	SID	Calls Sequence
UA1	S01	POST /login, GET /accounts, loginButton
UB2	S02	POST /login

Figure 5.3: Transformation of data points in sequence based sessions

5.2.2 Feature Engineering

One of the main challenges of detecting patterns in an application session is how to transform a sequence of calls of variable size in such a manner that is understandable to machine learning algorithms. Hence, a proper feature engineering is necessary to extract normalized frequencies of the same size and format that represents the sequence of behaviour in the application. To this end, the three selected approaches are explained in the following subsections.

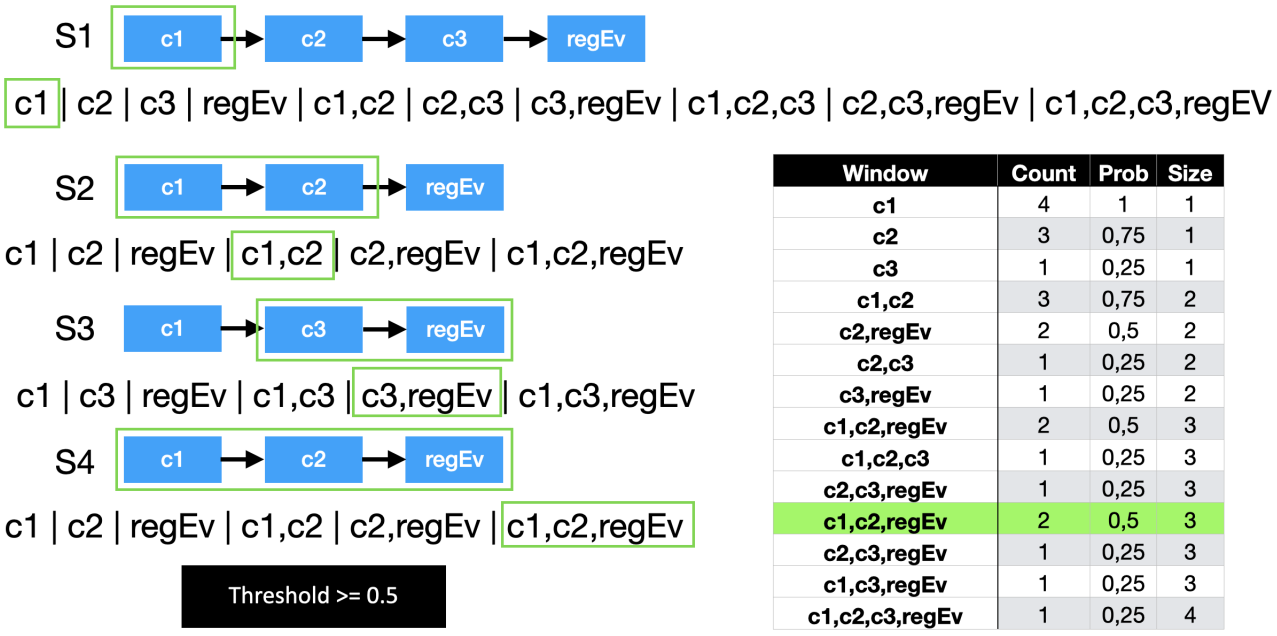
Levenshtein Distance

The first feature engineering approach, that could be considered the most intuitive one for this use case, applies the Levenshtein distance to measure the amount of editions to transform a sequence of characters into another sequence [72]. By detecting the patterns of calls for each click event, it is possible to compute the distance between a previously detected pattern and an incoming session. In this case, higher distances will indicate that an anomaly occurred in the event in question [6].

To detect the patterns of calls per click, each session calls sequence is trimmed by its events. Depending on the logic of the app, the event will mark the beginning or the end the

set of calls per click.

After retrieving all the sets of calls per click that appeared in the dataset for each event, it is necessary to elect what is the subset of call that is more representative. By sliding a window of increasing size, a table containing the occurrence count, the probability and the window size is created. Intuitively, the higher probability would be the right candidate for the pattern of calls per event. However, as the sliding window has to start with one for the purpose of better generalization, selecting only the higher probability would give us mostly candidates with one call. Instead, a better approach is defining a threshold and elect the bigger window size that surpass this threshold. Assuming that the *regEv* is the specific call that register the events of the app as mentioned in Section 5.1 and is the marked as the end of the set calls per click, Figure 5.4 illustate the whole process of electing a candidate of a pattern of calls for the certain event.



Window	Count	Prob	Size
c1	4	1	1
c2	3	0,75	1
c3	1	0,25	1
c1,c2	3	0,75	2
c2,regEv	2	0,5	2
c2,c3	1	0,25	2
c3,regEv	1	0,25	2
c1,c2,regEv	2	0,5	3
c1,c2,c3	1	0,25	3
c2,c3,regEv	1	0,25	3
c1,c2,regEv	2	0,5	3
c2,c3,regEv	1	0,25	3
c1,c3,regEv	1	0,25	3
c1,c2,c3,regEv	1	0,25	4

Figure 5.4: Calls per event pattern selection example.

In sequence, a dictionary with all the events and its calls is formed. This dictionary is used to compute the Levenshtein distance against all the ones retrieved from the testing sessions. This transforms all the sequences of calls in a sequence of Levenshtein distances

for every event present in a session.

Given two session calls sequences S_i, S_j, Levenshtein distance between S_i and S_j is:

$$lev(S_i, S_j) = \begin{cases} max(|S_i|, |S_j|) \text{ if } min(|S_i|, |S_j|) = 0, \\ min \begin{cases} lev(|S_{i-1}|, |S_j|) + 1 \\ lev(|S_i|, |S_{j-1}|) + 1 \\ lev(|S_{i-1}|, |S_{j-1}|) + k \end{cases} \text{otherwise} \end{cases} \tag{5.1}$$

where $K = 0$ if $(|S_i| = |S_j|)$ otherwise $K = 1$. The added values can be manipulated to reflect the error weight of substitutions $(lev(|S_{i-1}|, |S_{j-1}|))$, deletions $(lev(|S_i|, |S_{j-1}|))$ or insertions $(lev(|S_{i-1}|, |S_j|))$.

Given S_1 and S_2 two session calls sequences, where $S_1 = c1 \rightarrow c2 \rightarrow regEv$ and $S_2 = c1 \rightarrow c3 \rightarrow regEv$. The Levenshtein distance $lev(S_1, S_2) = 1$ where one substitution operation has been done.

Markov Chain

This second approach intends to detect the probability of the transition between an element, either a call or event, and the subsequent one. A Markov Chain is better visualised as a directed acyclic graph [46] with transitions probabilities. Applying to the use case, a session is a sequence of steps while walking through the graph that could be directly transformed to a sequence of the transition probability frequencies. The data-driven model computation of the Markov Chain consists in iterating through sessions, adding to a counter of occurrences and calculating the origin element divided by the sum of all the destination element which the origin is the same [89].

The model is based on $n - gram$ sequence. An $n - gram$ is a contiguous sequence of n events and m calls from a given click event sequence. The intuition of the $n - gram$ is that instead of computing the probability of an event and call given their entire histories, the transition probability of a set of events and calls occurring in a sequence are computed.

Thus, to calculate those transitions, bigrams are extracted from the sequences to compute the conditional probability of original call c_o targeting the call c_t. In other words, it is approximated with the probability $P(c_o \rightarrow c_t)$. Given the calls sequence $S_{i,j} =< c_1, c_2, \ldots, c_n >$ of the user i in any session j, the bigram of the sequence is computed. To compute a particular bigram probability of a call given a previous calls, the count of the bigram $B(c_o c_t)$ is computed and normalized it with the sum of all the bigrams that share the same first events:

$$P(c_o \rightarrow c_t) = \frac{B(c_o c_t)}{\sum B(c_o)} \tag{5.2}$$

A Markov Chain is constructed from multiple successive events and their occurrence conditional probability. In other way, it is a directed graph that represents successive transitions between click events sequences through the sessions and users. Specifically, each vertex in the graph represents either an event e_i or a call $c_{i'}$ and each edge (e_i, e_j) and $(e_i, c_{i'})$ indicate that there is a continuous bigram between e_i and e_j, and e_i and $c_{i'}$, respectively. The Markov Chain over a set of type of events and calls is a weighted directed graph $G = (V, E, \gamma)$:

- V is a set of events and calls.
- E is a set of edges in G. Let v_i and v_j be two events or calls in V. There is an edge $(v_i, v_j) \in E$ if and only if there exists a probability with a dependency rule: $v_i \xrightarrow{\gamma} v_j$.
- γ is the function from equation 5.2 that assigns a probability for each edge (v_i, v_j).

If the model is built without seeing at least one example of every possible interaction, further evaluation of a session with an instance of an unknown incoming legit transaction would lead to a detection of a false positive anomaly. On the other hand, application or data storage errors that miss transitions, but appear in modelling, would result in skipping anomalous sessions, namely false negatives.

Thus, the Markov Chain should be modelled with a substantial amount of data and a threshold filter is applied to support a minimum representation on the model. The more data is seen, the more accurate the threshold can be set as a trend of decrease in error

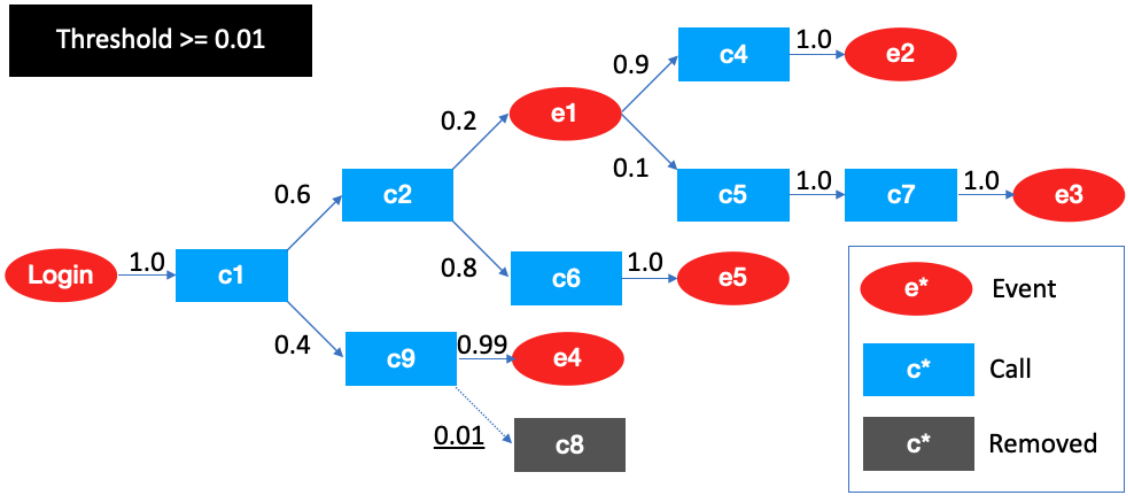


Figure 5.5: Markov Chain example with the removal of a call below threshold

representation is expected, except if the dataset contains over threshold recurrent errors. After setting the threshold, the outcome some be as shown in the modelling example of a small application in Figure 5.5, where the call c8 would be removed as instance.

The strategy for this filtering is to exclude when they do not match a minimum transition probability. Given the weighted directed graph $G = (V, E, \gamma)$ and a minimum threshold $min\_support$, the filtered Markov Chain is a weighted directed graph $G^{'} = (V^{'}, E^{'}, \beta)$, where:

- $V^{'}$ is the subset of events from V ($V^{'} \subseteq V$).
- $E^{'}$ is the subset of edges in $G^{'}$ where $E^{'} \subseteq E$.
- β is a function that assigns for each edge $(v_i, v_j) \in E^{'}$ the probability $P(v_i \rightarrow v_j)$ where $P(v_i \rightarrow v_j) \geq min\_support$.

In Figure 5.5, the minimum support threshold is set to 0.01 as an example. The transition probability from the call $c9$ to the call $c8$ is lower the the minimum support threshold. Therefore, it is removed from the model.

LSTM

This section describes the Long-Short Term Memory (LSTM) neural network encoder [57] model that operates at the API Calls level to generate a vector representation of sessions.

The encoder consists of the word vector dictionary known also by embedding layer [24]. It is used as dense encoding compared to one-hot encoding to extract features from session calls. LSTM layers are used to encode the sequence of calls. On the other hand, the head of the encoder consists of 2-LSTM layers, which predict the next call within a session at each iteration. Once the model is fine-tuned, the LSTM model extracts the sequence of features to represent a dense session vector.

Embedding layer: First, the dataset asset represents multiple sessions SID. Each S_i is comprised of a sequence of API calls. Each call c_i constitutes an integer vector representation of calls denoted as a binary valued vector $b_{c_i} \in \{0,1\}^k$, where k is the number of unique calls. The embedding layer allows to turn each call b_{c_i} into a fixed length vector of a predetermined size d, as a dense vector representation $w_i \in \mathbb{R}^d$, this constant length of calls vectors allows better modelling and representation of calls while also reducing their dimensionality. In the pipeline, the embedding layer is made up of two-dimensional input matrix, called *session matrix*, represented as $E_S \in \mathbb{R}^{m \times d}$, where d is the calls embedding dimension and m represent the length of session $S_i, |S_i| = m$, where $m > 0$.

Therefore, the result is a dense representation matrix rather than simply a sparse matrix. The E_S is made as an input the LSTM layer.

LSTM layer: A special kind of architecture called *many-to-one* LSTM model [51] is able to learn long dependencies in call sequences in unidirectional LSTM model given the sequential nature of the data. LSTM gates are selected due to their high performance in capturing long-distance dependency between word sequences in short texts [122], with the same analogy; calls are considered as words and sessions as sentences.

Many-to-one LSTM take into account only the previous only for this reason, it was used deep many-to-one LSTM of 3 layers to consider the information from the previous hidden states. Therefore, the model is able to predict the next call from a sequence of calls called session using the multi-class cross entropy as the loss function to train the model using SGD

as an optimizer method [99].

In the experiments, the focus is to investigate the effectiveness LSTM frequencies based on LSTM embedding fine-tuned using the next call prediction as machine learning task. Therefore, once the dense vector representation of each call $w_i \in \mathbb{R}^d$ is reached, the session matrix $E_S \in \mathbb{R}^{m \times d}$ is constructed to apply set of 5 statistical frequencies of top of E_S, as feature extraction approach to represent sessions.

5.2.3 Normalisation

When a session sequence of calls and events is transformed into frequencies with any of the 3 previous explained algorithms, the output is always a vector of values. For the Levenshtein distance, the vector varies according to the amount of events. The Markov Chain will output a value for every bi-gram encountered in the session. The most expensive one is the LSTM that will create a vector of the hyper-parameter size for each call or event increasing drastically the size of the dataset.

The Machine Learning algorithms cannot work with such heterogeneous sizes and formats frequencies. Hence, the features have to be normalized in order to produce the same value ranges $([0, 1])$, while keeping the properties that will enable the classification methods to distinguish between a normal and an anomalous session. For this purpose, the following five statistical methods were selected based on the values of each session:

1. Min: the minimum value.
2. Max: the maximum value.
3. Avg: the average of all the values.
4. Log: the sum of the logarithms of the values.
5. Std: the standard deviation between all the values.

After processing the five computations for each of the 3 feature engineering algorithms, the output is a dataset with 15 columns of a single numeric value for each session. The

reasoning for selecting these methods varies for each of the feature engineering algorithms. A high Levenshtein distance and Markov Chain probabilities are likely to be represented in normal session, while a high LSTM embedding distance would be a probable anomalous session. For the use case, the min for Levenshtein and Markov Chain and max for LSTM values would capture at least one anomaly appears in the session. The average and standard deviation are expected to be distinguishable from normal sessions if more discrepant values appear in a session. The sum of the logarithms would increase for sessions with more anomalous transitions.

5.2.4 Balancing the data: SMOTE

In this section, SMOTE [23] method is presented to solve the crucial issue with imbalanced API call sessions dataset.

Over the past decade, class imbalance learning is well known as a challenging task in supervised machine learning, two approaches exist in the literature and commonly used for machine learning models, over-sampling and under-sampling techniques.

In this work, the focus is on oversampling techniques and instead of using a naïve non-parametric oversampling technique such Random Over Sampling (ROS) [85], which leads to create duplicates in the dataset by simply extending the minority classes by drawing randomly and with replacement anomalous sessions. A parametric over-sampling techniques is opted and according to the review presented in [44], SMOTE is the most famous and efficient over-sampling technique, where the algorithm generates synthetic samples for one or more minority classes based on interpolation techniques, in the use case anomalous rare sessions.

As shown in [44], many SMOTE variant strategies have been proposed, however, the pipeline of this work use mainly SMOTE based on arbitrarily interpolates to generate new synthetic minority samples where each new sample must be between real samples of each specific minority class by following the K-nearest neighbors (KNN) algorithm, as described in [44].

As illustrated in Figure 2, once the dataset is normalized, the entire imbalanced dataset

is fed to the SMOTE algorithm to smooth out and over-sample the imbalanced properties of each class before the training phase to improve the performance of the model, which by fact SMOTE avoid the over-fitting compared to ROS by keeping the new synthetic samples within the boundaries of original data points [23].

5.2.5 Machine Learning

After the proper feature engineering, normalisation and balancing of the data, the machine learning task should be able to differentiate normal from anomalous session based on the behaviour frequencies extracted by the previous steps of the pipeline. In this sense, the main classification machine learning are expected to achieve good performance. The previous work [46] applied Naive Bayes, Decision Tree, Random Forest and Support Vector Machines to their use case, that can be considered similar in terms of the machine learning task.

5.3 Experiments

To run the experiments the dataset was partitioned in 3. One part for modelling the feature engineering calls per event pattern, the Markov Chain and the LSTM embeddings. After parsing the data as described in Section 5.3, the first challenge was to run the three feature engineering algorithms in sufficient data so the models created have representations of all the transitions. After a business reasoning, 30 days of data were elected to generate the models as it should contain almost, if not all, the click events and their respective calls. In order not to disclose any information regarding the banking daily transaction, there is no disclosure on how much data represents the 30 days. Also, the experiments were performed in the dataset with data from the same version of the mobile application, discarding the possibility of new events added in newer versions. From the remaining part of the data, 70% was used for training the machine learning algorithms and 30% for testing them.

5.3.1 Environment

The experiments run on a 4-node Spark 2.3.2 cluster with 18 GB of RAM and 4 cores each, and 1.5 TB of disk space, on the bank premises. The parsing and feature engineering was developed in Scala 2.11 and the remaining part of the pipeline in Python 3.

5.3.2 Extracting the features

Each feature engineering algorithm described in Section 5.2.2 was evaluated in the training dataset to extract the patterns of calls per event, and the transition probability and the embedding distances between the calls and events.

Calls per event patterns

To isolate the calls that are performed in relation to a particular event, first it is necessary to understand when the event is sent to the back-end. In the bank mobile application analysed, the call is the last one in the chain. In other words, when a button is pressed in this mobile app, all the data exchange calls are performed before sending the last call that will register the event. This results in a behaviour that is similar as illustrated in Figure 5.4.

By trimming the sessions by the event and collecting all the calls that appear before the event, all the sets are joined to create a dictionary containing all the calls that appeared for the events with the occurrences counted. At this stage, it was expected that the totality of events of the dataset would have multiple set of calls, however, only part of them trigger calls. This is not a problem for the use case because the main focus is in capturing the behaviour of the events and calls that could harm the system, videlicet the ones that can modify the state of the server, e.g. execute a wire transfer.

Then a sliding window from 1 to size of the set of calls outputted all the n-grams possible for each of the events. Those n-grams were then counted for every type of event and their probabilities calculated. By capturing the most recurrent one, a dictionary with 234 events and their respective calls is formed. The distribution of size of the set of calls versus the quantity of events is illustrated in 5.6. In the setup, this process took around 20 hours to

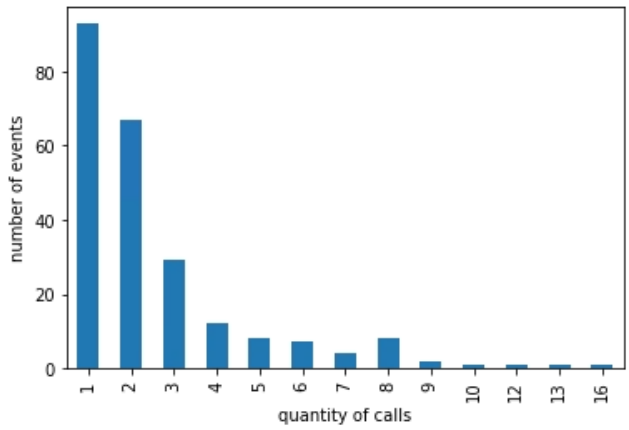


Figure 5.6: Distribution of quantity calls per number of events

finish for the whole 30 days.

The Levenshtein distance algorithm operates in character level. This means that the events and calls have to be replaced by a single character. Besides the 311 events, there are other 538 calls that have to be mapped to characters if this replacement is performed in the dictionary level, posing another issue as the encoding tables are 256 characters maximum, such as ASCII. To overcome this limitations, the dictionaries are kept with the full calls and events names and transformed into single characters while evaluating each session. None of the sessions surpassed the limit of characters.

Finally, each session was trimmed by the event and the respective calls, and the levenshtein computed against the dictionary value. The session are then transformed to the 5 statistical frequencies. After reducing the 5 dimensions to 2 with PCA, the testing dataset can be seen in Figure 5.7.

Transition probabilities

Transforming the sessions in the transitions probabilities was more straight forward than the calls per event patterns. To this end, all the sessions were transformed in bi-grams and then counted. The amount of one origin to a destination call divided by the sum of the all of the transitions with the same origin results in the transition probability. The threshold of

0.01 filtered off X events forming a bi-gram dictionary containing their respective probability. The whole computation of the Markov Chain based on 30 days of data took over 8 hours to complete.

The training and test dataset were mapped to their bi-grams transition probability distribution. Figure 5.8 illustrates the PCA reduction to 2 is again applied to the normalised sessions.

Embedding Euclidean Distance

The most computationally expensive algorithm to perform was the LSTM embeddings to transform the sessions in embeddings and then compute their distances, taking over 12 days to complete for the 30 days of data in the cluster.

Firstly, all the calls and events are transformed in a symbolic computation graph with an embedding of 64 positions. In sequence, the initial states for 512 hidden and 3 lstm layers are set. Those hyper parameters were selected based on the amount of transitions found in the Markov Chain.

Batches of 32 calls and events were taken in 20 epochs per day to generate the LSTM model for the 30 days of data. The training and test dataset were then transformed to embeddings, costing a lot of disk memory to the cluster. Only one day of data surpassed 150 GB. The distances are then calculated by measuring the Euclidean distance from the model embedding to the sessions embedding. The 5 normalisation statistics were computed and reduced to 2 dimensions with PCA to present the Figure 5.9.

5.3.3 Balancing the data

The training set was formed by 515,816 with 6602 anomalous sessions, representing around 1.28%. If the algorithms were directly trained using this set, there was a high probability of the machine learning algorithms generate false positives and false negatives. First experiment was attempted to train and test with this data and the best performance was 82.73% in the random forest with mixing the Levenshtein distances and Markov Chain probabilities.

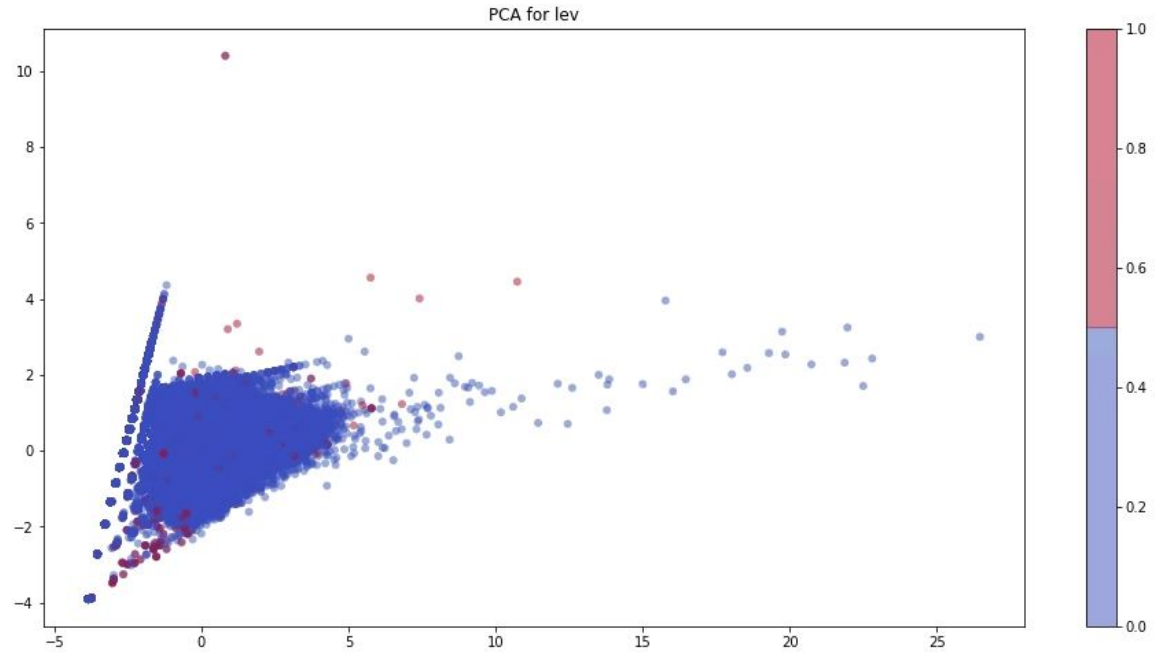


Figure 5.7: Principal Component Analysis of Levenshtein Distances

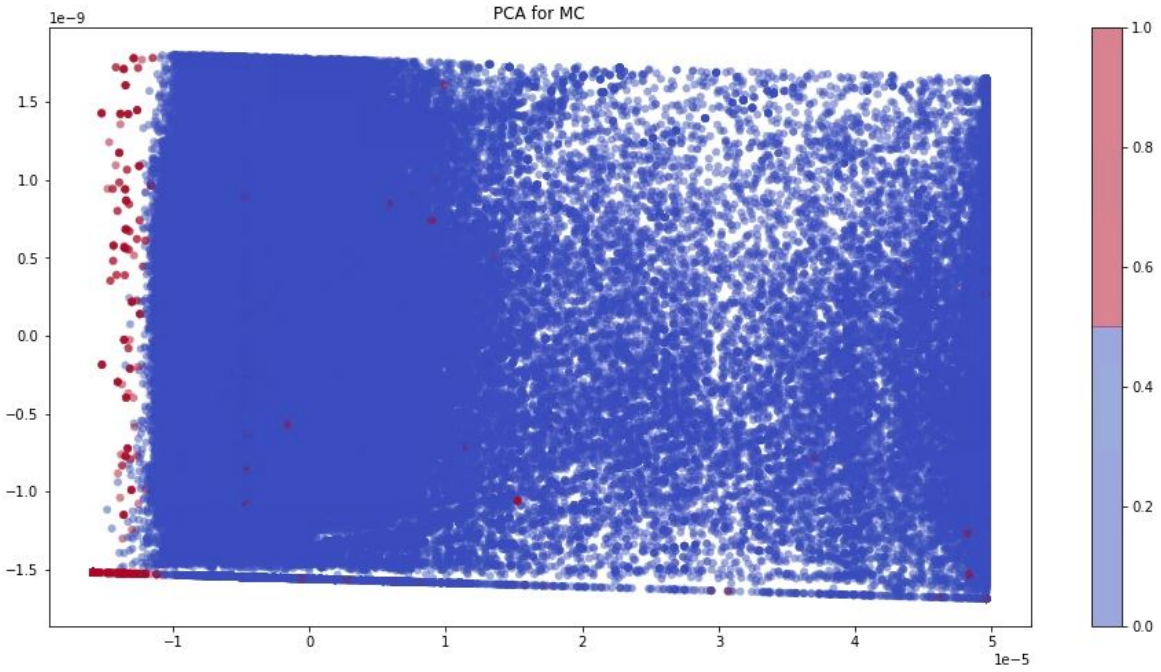


Figure 5.8: Principal Component Analysis of Markov Chain

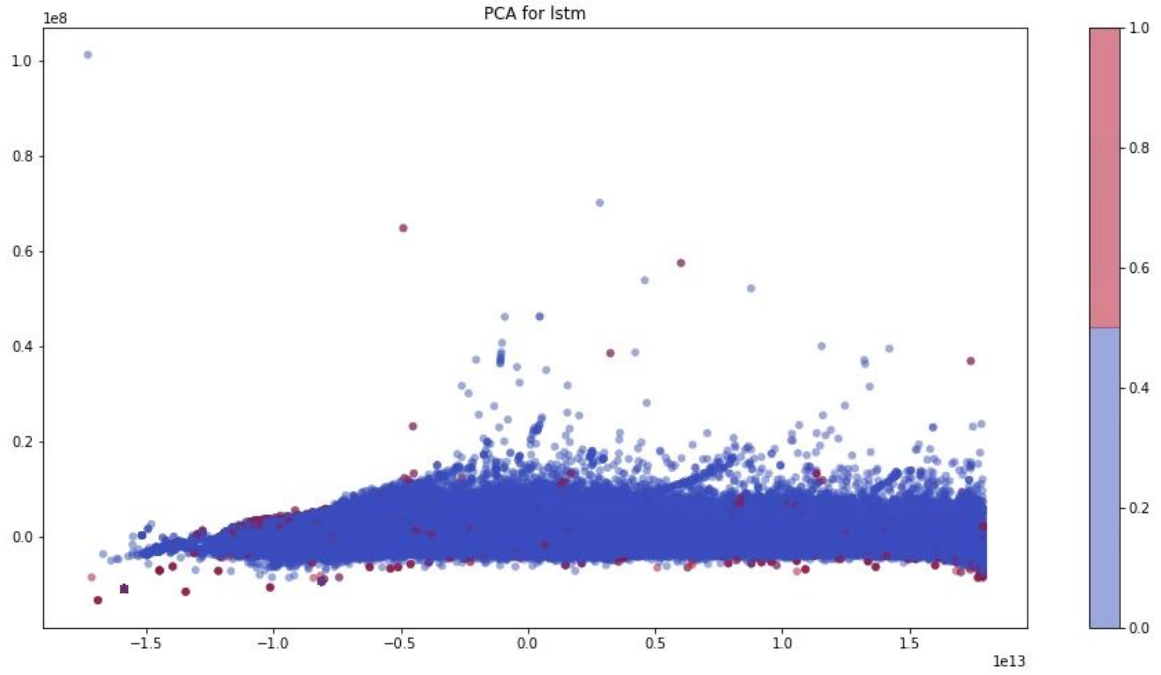


Figure 5.9: Principal Component Analysis of LSTM

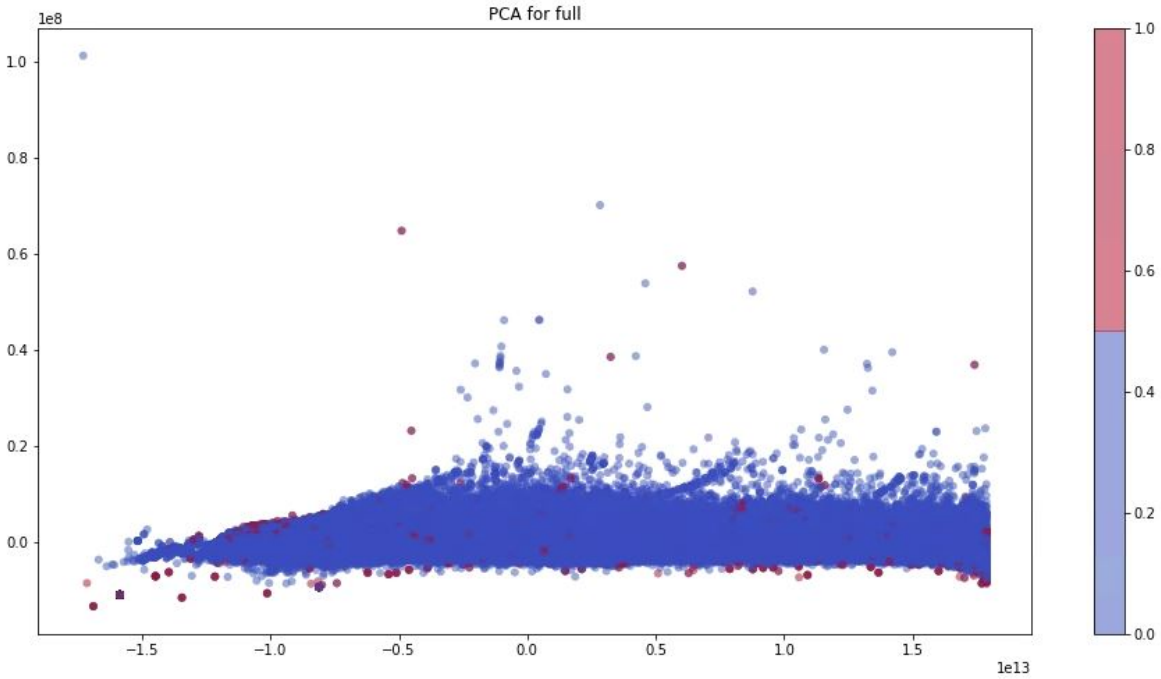


Figure 5.10: Principal Component Analysis of all features

By applying the SMOTE algorithm to balance the data, better results were achieved.

5.3.4 Results

The test set contains 154,745 in which 1,984 belongs to anomalous class. The first approach to the features was trying to see if a visual division of the classes using PCA could be observed. A first visual glance of all of the features, reduced to 2 dimension, can be seem in Figure 5.10. By comparing with the other figures, it is visually notable that LSTM-based features are the most prominent. But the graphical distribution of the Markov Chain features were more promising as the anomalous class are more dense in the left side. To evaluate the performance of the created features over the machine learning algorithms, each set of 5 features were evaluated for Levenshtein Distance (Lev), Markov Chain (MC), LSTM, and mixtures of them over 7 machine learning algorithms: Gaussian Naive Bayes, Decision Tree, Random Forest, Support Vector Machines, K-nearest neighbors, Logistic Regression and Support Vector Classification with kernel RBF. The results are report for the 3 best F-measure scores in table 5.1. The best performance achieved **96.61%** of F-measure when combining the Levenshtein distances with the Markov Chain probabilities. Notice that individually those algorithms could not perform good results. Adding the LSTM features degrades the performance.

Algorithm	**Gaussian NB**	**Decision Tree**	**Random Forest**
Lev	0.036084	0.017385	0.038651
MC	0.138052	0.000000	0.000000
LSTM	0.046971	0.110930	0.110706
LSTM + MC	0.046971	0.731652	0.764765
Lev + MC	0.045405	0.940885	**0.966179**
LSTM + Lev	0.046971	0.061807	0.102190
All	0.046971	0.912612	0.950993

Table 5.1: F-measure scores of Machine Learning algorithms using individual and mixture of features

Figure 5.11 illustrates the most relevant features taken from the Random Forest with all the features. Basically the logarithmic feature of the Markov Chain contributes most to

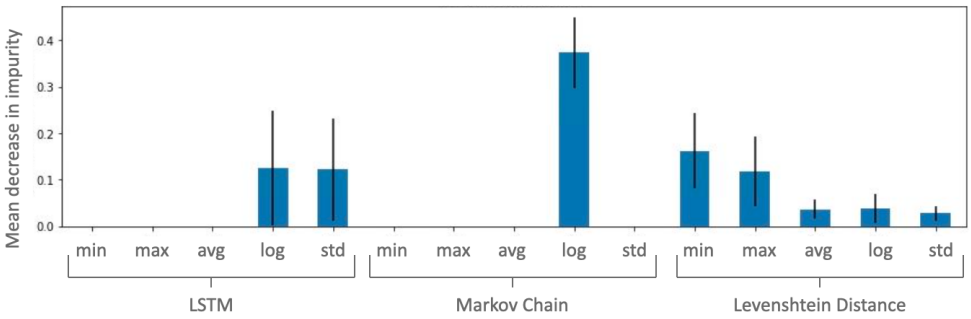


Figure 5.11: Complete Feature Importance

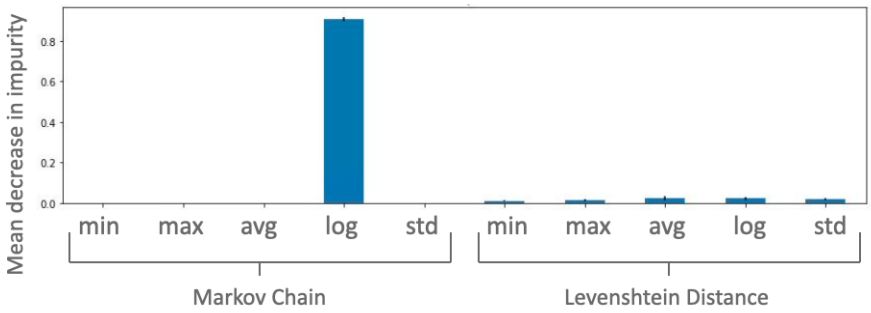


Figure 5.12: Feature importance for Levenshtein Distances and Markov Chain mixture

the performance but all the Levenshtein Distances when added help to increase the performance as seen in Figure 5.12.

The last thing to be comprehended is how many false positive and false negatives are raised by the approach. Figure 5.13 presents the 3 confusion matrices outputs of the machine learning models applied to the Markov Chain and Levenshtein Distances features. The best classifier to the use case, Random Forest, skips 70 anomalies on top of the 1984, representing 3.52% of false negatives, and misclassify 64 normal sessions or 3.23% of 1978 positive labels.

5.4 Limitations

Even though a statistical analysis can try to identify the threshold for Markov Chain, filtering should be manually set by domain experts and may lose real transitions when there is a

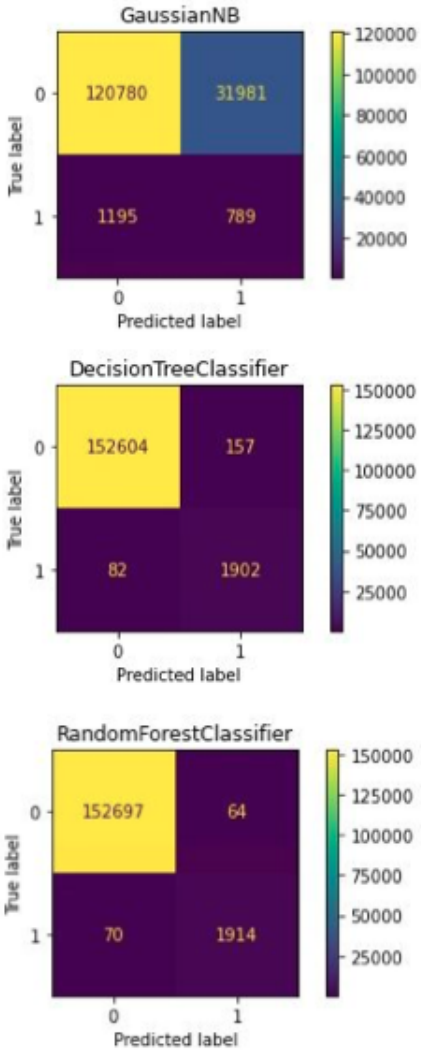


Figure 5.13: Confusion matrices for Levenshtein Distances and Markov Chain mixture

menu in the application with many items and some of them not accessed frequently. Therefore, for better performance, it is highly recommended that a map of the application should be created and the probabilities added to the map instead of trying to extract the transitions based on the data.

New versions of the mobile application requires a new modelling to add or remove events and respective calls transitions. As the modelling is fully data-driven, the model will only be complete when users click on every possible interactions. This means that a new released version would output false negatives and positives for some period while learning from the data.

The anomalies detected here could also contain errors in the data flow of the application, thus, it cannot be guaranteed that they are malicious attempts on the application. This means that the approach is good for generating alerts but they have to be manually analysed afterwards.

Chapter 6

Privacy Preserving Decentralised Super App

Another emerging concept of mobile development is the super app, an application, usually mobile first, offering a set of services from different stakeholders, namely member apps. In an unique platform, different providers can offer several types of services. Although it is named super app, an instance of a super mobile application is only one part of the necessary infrastructure. Actually, every member mobile application needs to implement its business logic portion, while the super mobile application owner, if there is one, maintains the infrastructure.

For instance, the popular Chinese WeChat app, initially released as a messaging service, has lately integrated a number of other services. As a result, it is now possible to order a taxi or make a doctor's appointment using the app. From a user experience point of view, a super mobile application is highly desirable as it eases access to different services. Nevertheless, it raises privacy concerns as the super mobile application owner retains high control of user data and information provided. As instance, recent new stories accused WeChat of censoring information related to the COVID-19 outbreak[1]. Indeed, data-driven applications are common target for a vast number of malicious attacks that can ultimately

[1] https://www.buzzfeednews.com/article/ryanhatesthis/china-tencent-wechat-yy-censorship-coronavirus

reveal individual private data from millions of users [35].

According to KPMG [2], the next trend for super apps is its adoption by large financial institutions. Such super apps could gain significant control over data describing user habits including lifestyle and purchasing behaviour. Financial institutions are likely to be the next industry to widely adopt this kind of digital ecosystem [42]. From the user experience perspective, the super mobile application facilitates access to broader services.

Solutions, such as the hailed end-to-end encryption implemented in WhatsApp, do not solve these types of problems and create a false perception of privacy[3]. Nonetheless, the Facebook group has shifted their platforms to enable addition of multiple services from outsiders. As instance, Brazilian banks use chatbots via Whatsapp to interact with customers in order to perform small transactions[4].

As the super mobile application is a highly coupled environment with many different stakeholders, a lack of a proper reasoning on privacy can expose several vulnerabilities. Taking into consideration the Brazilian bank example, it is possible to identify a conflict of interests between Whatsapp and the banks. Banks want to be assured that Whatsapp is not reading the transaction's messages from the phones. On the other hand, by having access to these messages, Whatsapp would be able to gain additional knowledge of their users. This is only one example of open issues to deploy solutions in such environment.

In fact, users can easily lose control of their data as exchanges between the stakeholders tend to increase when a wider range of service starts to be offered. This lack of control can happen in twofold: 1) the type of data collected; 2) with whom the information was shared. Usually, the super app owners control the whole users dataset, therefore, it completely depends on its goodwill to create mechanisms that guarantee control of data to the user. Indeed, data-driven applications are a common target for a vast number of malicious attacks that can ultimately reveal individual private data from millions of users[36].

In 2017, the European Union emphasized the importance of data marketplaces to make the EU more competitive in terms of computational intelligence [61]. Furthermore, data

[2]https://home.kpmg/xx/en/home/insights/2019/06/super-app-or-super-disruption.html
[3]https://theconversation.com/becoming-more-like-whatsapp-wont-solve-facebooks-woes-heres-why-113368
[4]https://iupana.com/2020/02/17/despega-la-banca-whatsapp-en-brasil/

marketplaces could be enhanced by API's and blockchain technology could help to the development of data-sharing models [28].

In addition, blockchain presents many advantages to develop the data marketplaces in EU, such as: there is no need of a trustworthy intermediary; it is offered transparency and the smart contract could reduce costs when the sharing of data is automatized [47].

In this context, the European Parliament report published in 2018 highlighted the importance of the blockchain technology to the development of solutions for the compliance with the principles of ensuring secured and self-governed data [84]. Thus, this article seeks to present alternatives to ensure that data subjects could have control over their personal data as regulated in Recital 7, Article 15 and Article 20 of the General Data Protection Regulation (GDPR).

According to [40], the connection between an offline storage with the blockchain would help data subjects to "control and own their personal data, while service providers are guests with delegated permissions.". Therefore, the use of blockchain technology in super apps will help to empower data subjects to decide which data will be provided and when the consent will be revoked in a transparent way.

Another rising issue when it comes to super apps is the fact that the business logic tends to be centrally controlled if there is a super app owner. As instance, a food marketplace that sells meals from several restaurants but with the application logic completely owned by the marketplace provider.

Considering those issues, a complete decentralised architecture is proposed in order to: 1) Prevent that a single owner of the infrastructure control the whole users' datasets; 2) Provide a full log of data access between the stakeholders; 3) Allow member apps to create their own interaction experience to be plotted at users' screens; 4) Publicly attest all the member apps programming interfaces within the super app.

This chapter presents a framework that enables the creation of a super app using privacy by design principles. The proposed framework presents the following main features and contributions:

- Endpoint security: the super app communication channels must always be imple-

mented using the Trusted Execution Environment (TEE) Intel SGX cryptographic properties. By properly implementing authenticated key exchange protocols between a super app instance and a SGX protected server, requests reaching endpoints are trusted to be arriving from a legit user instance of the app. In other words, an attacker will not be able to create a fake app or script to interact with the endpoint. The details describing how this can be achieved are found in Section 6.1.1.

- Privacy by design: the user's privacy is guaranteed by default. No action from the user is needed to maintain its data safe and used only when allowed. The proposed framework includes user's permissions management system.
- Protected data exchange between member apps: the system should orchestrate the creation of a secure tunnel between member apps to guarantee data encryption. This orchestration aims also to provide footprints in order to keep track of users's data flowing among members apps in an auditable manner.
- Cloud deployability: the correct implementation of components will protect against cloud provider insider attacks, as it is processed in a TEE.

The approach allows the service providers to publish a set of APIs or even creating their own user experience design, while publicly attesting the whole executed on their behalf. Any auditing or inspecting entity can request evidence to be presented about what data was requested from the user and who had access to it. To the best knowledge, this is the first approach to a Decentralised Super App designed to preserve privacy.

6.1 System Requirements

The core of the system is the application that will take care of users data, namely Super App Management Service (SAMS). It is designed to keep the privacy of the users by masking all sensitive data exposed to the member apps and providing strong authentication. To those ends, SGX was elected as the root of trust as it has the ability to process data without disclosing it even when an attacker is in control of the machine. This is a feature for mitigating

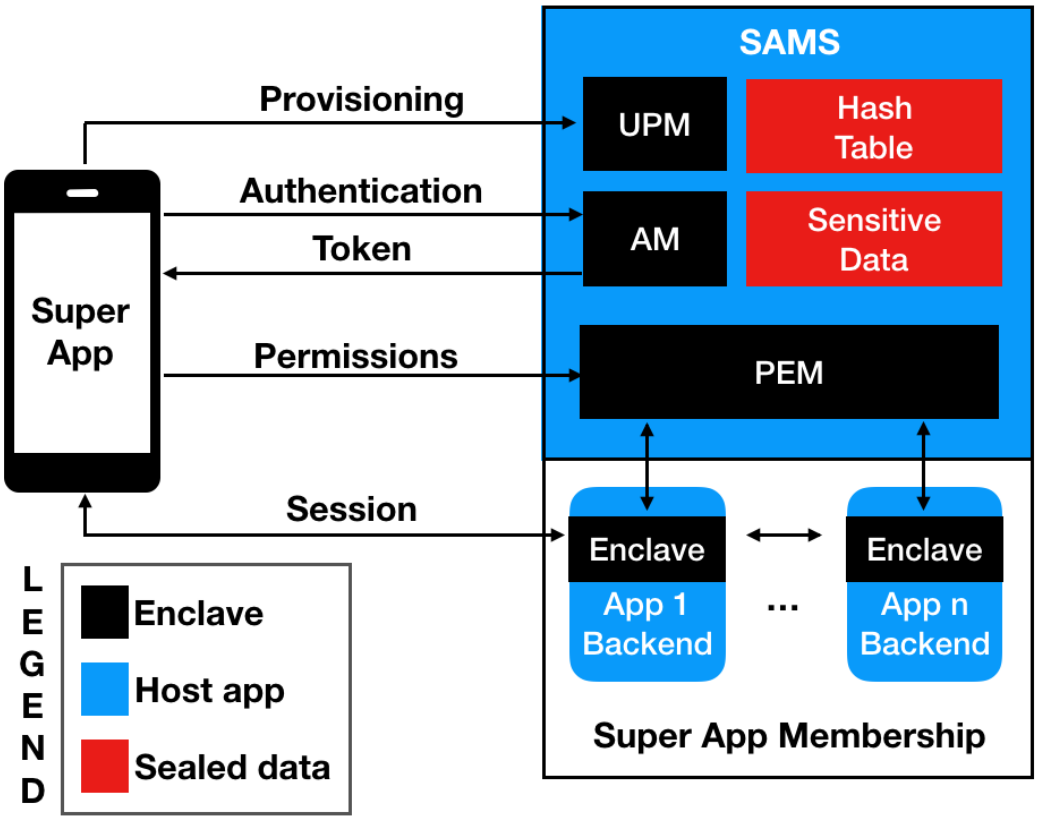


Figure 6.1: Architectural blueprint

the risk of data disclosure when running in a third party cloud environment. By adopting SGX cryptographic features the following components can be combined as shown in Fig. 6.1.

6.1.1 User Provisioning Module (UPM)

This component is a SGX enclave that creates a new incoming user. By generating a new unique id and inserting it in a hash table that will map a different id for each member app, the system can guarantee that the identity of the user is only known by SAMS enclaves. The previous work on this topic [91] covers a protocol designed to assure that end-to-end encrypted messages are coming from the mobile app directly to the SGX enclave. This is perfectly adequate to ensure that communication with the UPM is actually coming from the super app. In addition, this module is also responsible for safely storing users credentials

using SGX sealing functions, which is already covered in [76].

6.1.2 Authenticator Module (AM)

Once the user is created and their credentials are stored, the user can start using the super app. When logging in, the user will have his credentials collected and sent to the Authenticator Module implemented in another set of enclaves. To implement this module, industry standard FIDO Alliance specifications can be followed. FIDO security requirements [77] classifies implementations of authenticators in trusted execution environments as level 3, being only considered less strong than an implementation in a dedicated secure element. After being authenticated, a session is open and a OAuth2 token is issued to the user super app instance, in a similar fashion as presented in [119].

6.1.3 Privacy Enhacing Module (PEM)

When interacting with a certain member app, the super app presents user's OAuth2 token to an enclave that communicates with its backend. This backend enclave interacts with the SAMS via PEM. This module checks if the session is open and returns the hash of the id of the user bound to that member app.

The permissions the user gives to that member app are also registered in the SAMS. PAM will take into considerations those permissions when the member app backend requests sensitive information from SAMS.

Another funcionality of PEM is to guarantee privacy to data exchange between member apps. This is one of trickiest functionalities of SAMS. As each backend knows a different ID for the same user, PEM will interface the data exchange by acting as the server in Extended Triple Diffie-Hellman (X3DH) [83] of Signal protocol. In this way, the communication ciphered tunnel between the member apps will only open if PEM verifies that the user granted access to that member app to request the data to the other one. A previous work covering group access control can be found in [29]

6.1.4 Super App Membership

Super app membership will consist of provisioning an enclave app to the new member. This process is crucial in order to enable establishment of X3DH tunnels between the other members. Additionally, if the super app is implemented as a renderer, this membership can provide all mobile user interface components.

There are two approaches to implement member enclaves. The first one is as a part of SAMS infrastructure. To be more precise, the enclave is then called by a SAMS host app. This would give greater control to the SAMS owner as the functions are all implemented by him. In this case, all the data coming from the super app would need to be forwarded to the member app backend, increasing the network time.

Conversely, the member app developer can implement the enclave as a part of his backend. This would give more freedom to customise the functions according to the business needs. By attesting member's enclaves, the necessary auditing task of verifying their codes becomes mandatory for SAMS and poses a delay to the developer. Every time the developer changes the code, another measurement would be generated and could immediately be detected by PEM and consequently block all the communication with the backend, as session establishment between super app and member apps is based on key derivations mediated by PEM. This approach would require a stronger software deployment process, yet, it makes SAMS code base lighter.

In both cases, the super app will always communicate with an enclave. This principle ensures that the endpoints are secure, if implemented in a similar manner as presented in [105].

6.2 Challenges

Even though the technologies presented in the last section solve problems of designing the proposed system, there are other issues that has to be solved. In this section, a set of concerns to be addressed are presented.

6.2.1 Scalability

As the borders of the apps are all built using SGX, horizontal scaling is not a simple matter of adding new processors. In fact, addition of new processors will require a process of provisioning the new SGX/enclave keys. However, if this is a weak process in the security perspective, the whole system can fail to provide the expected privacy. It is possible to summarize in one question:

How to securely add new enclaves to an existing group of enclaves?

6.2.2 Disaster recovery

Every time the data needs to be shared among different enclaves and sealed outside the enclave, there is a need to have a group cipher. In other words, all and only the allowed enclaves should have access to the sealed data. But this leads to a problem that would happen if all the processors in possession of the keys are lost, consequently, losing all the sealed data. For this problem the research question is:

Is it possible to recover data sealed by a lost group of enclaves?

6.3 Architecture

6.3.1 Design Principles

Smart contract as super app backend

Instead of a backend for the super app, a smart contract is designed. This smart contract is responsible for enrolling new member apps, serving a curated list of the active member apps and managing the permissions to share the data. When the super app instance starts, it will hit this smart contract to request all the list of member including their certificates, enclave and smart contract addresses.

Attested server driven rendering

A mobile app with a server driven user interface receives all the screens layouts from an API. Each file contains the content to be rendered, the components to collect data and the link to the next screens, creating a chain of layout files. By adopting this paradigm, the member apps are exempt from implementing client-side UI. In the approach, those layouts files are coming from member's enclaves, which will serve as the member app backend. The term attested server driven rendering is coined as the enclaves will sign all the layout files to attest what data is intended to be collected from the user and that the layout files are coming from a valid member.

Identification based on public key cryptography

After provisioning, users and members certificates are stored on the blockchain. Therefore, every time they need to prove their identity, asymmetric cryptography plays the role.

Enclave attestation report on chain registration

Member's enclaves register an attestation report in their smart contract. In this way, any modification to the enclave code is logged, easing auditing process. In other words, along with server driven rendering, it is completely clear what sort of data is collected from the user in each version of the enclave code.

Validation of app origin

The protocol for a "SGX enclave secure channel", developed in [10], might be used to add a validation origin step and to guarantee that the requests are coming from a legit super app instance. By applying this protocol in the communication between the instance and the member's enclaves, automatised attacks aiming to request user's data as well as submitting fake transactions are avoided.

Member apps smart contracts as data brokers

Members are allowed to exchange user's data. However, it is only feasible through a smart contract. An enclave in need of an specific data collected by other member app will hit the endpoint of this third party member to request an access token, leaving a data exchange audit trail.

User sensitive data is not stored on chain

User data is stored in his smartphone, in encrypted format when necessary, and in the member's databases. The blockchain will only maintain his certificate and the permissions gave to member's to collect and/or share his data. This action will help to ensure the right to be forgotten as defined by Article 17 GDPR and it will avoid the clash between DLT's characteristic of immutability and the GDPR, also following the recommendation of the French Data Protection Authority (CNIL) of decoupling sensitive data from the chain by using proper encryption methods to secure the data while keeping the exercise of erasure rights [26].

6.3.2 Super App Development

The decentralised approach assumes that the infrastructure is not owned by a single entity. However, the super app instances running into user's devices should have a centralised development as a single app will be deployed into the stores, whether it is a community or a group of stakeholders effort. The components of the super app instance are illustrated in Figure 6.2 and listed below:

User Interface

A common technique in mobile development is the dynamic rendering of screens layouts, which means that the mobile app embeds all the User Interface (UI) components and that the layout files comes from the server. By adopting this paradigm, the member apps are exempt from implementing client-side UI. The layouts files also include the endpoints to receive the data points. In other words, the screen coming from the member app will require data to

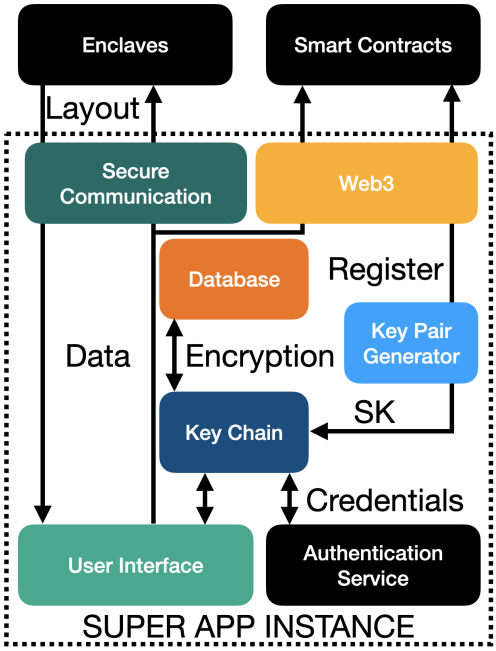


Figure 6.2: Super app instance components

be sent back via POST to an URL indicated in the layout file. Besides the layouts plotting and data collection, the user interface will also request permissions from the user to allow sharing data with and between the member apps. These permissions are registered in the super app smart contract.

Key Pair Generator

The identification of the user on the blockchain is based on his public key. Therefore, the super app developers are required to implement a key pair generation function. The generated public key should be registered in the super app smart contract.

Key Chain

It is crucial to the security of the system the protection of the private key or secret key (SK), at risk of personification by an attacker in case of the key stolen. User data is also required to be ciphered by the Key Chain methods. Both iOS and Android have methods for storing

this key safely in a keychain or keystore accessible only by the application sandbox.

Database

The amount of data to be stored inside the device can vary substantially. However, in most of the cases, the member apps can store the data associated with their usage, leaving a small portion to be stored locally and fully encrypted.

User Authentication

While protecting the private key can be easily implemented by using methods already available, authenticating the user to make use of the key can be tricky, specially when an user replaces his device, since there is no central repository for users credentials. Thus, the recommendation is to follow Strong Customer Authentication (SCA) requirements of PSD2. In this case, after authenticating, the app can use the private key bound to a single user to sign and exchange keys with the other hosts.

Secure Communication

The communication with the enclaves follows the protocol described in [91]. This protocol creates a secure channel to deliver sensitive data to the right enclave.

Web3

Interacting with smart contract requires an additional layer of communication standardised in the set of Web3 libraries [123].

6.3.3 API Design

In this section, the API of the super app smart contract, the member's smart contract and enclave are defined. Smart contracts can be reached by sending transactions or calls. Transactions change the state and pay a fee, while calls are read only computations in

the blockchain. The enclaves expose REST endpoints with HTTP methods. Below, each endpoint is described (see Figure 6.3) and its exposure to the other hosts:

Super App Smart Contract

I **Interfacing members smart contracts**

- TRANSACTION create member: Member app provisioning is exposed by this endpoint. The member app sends his public key, the address of the enclave and smart contract, and a signature of its enclave. In the background, the smart contract will contact Intel Attestation Service and check the authenticity of the enclave. A consensus mechanism to accept the newcomer member app, as well as, additional checks can be added, such as validating which data is requested by the member app. If these checks pass, the smart contract registers the app as valid in the blockchain.

- TRANSACTION delete member: The owner of the member app, which means the one in possession of the private key provisioned in the endpoint 6.3.3, can request deletion of his own member app. In fact, the super app smart contract will mark the member app as invalid after this request, but keeping the register for auditing purposes. The deletion of a member app can also be requested by a third member app, thus, requiring a consensus algorithm to decide whether to remove or not.

II **Interfacing mobile app**

- CALL list members: The instances of the super app hit this endpoint to get a list of valid member apps.

Member app Enclave

III **Interfacing mobile app**

- GET /layout: This endpoint provides the current layout file to be plotted in the super app instance. By identifying the user, the member can keep control of the current screen in use. The chain of screen layouts are kept into this file.

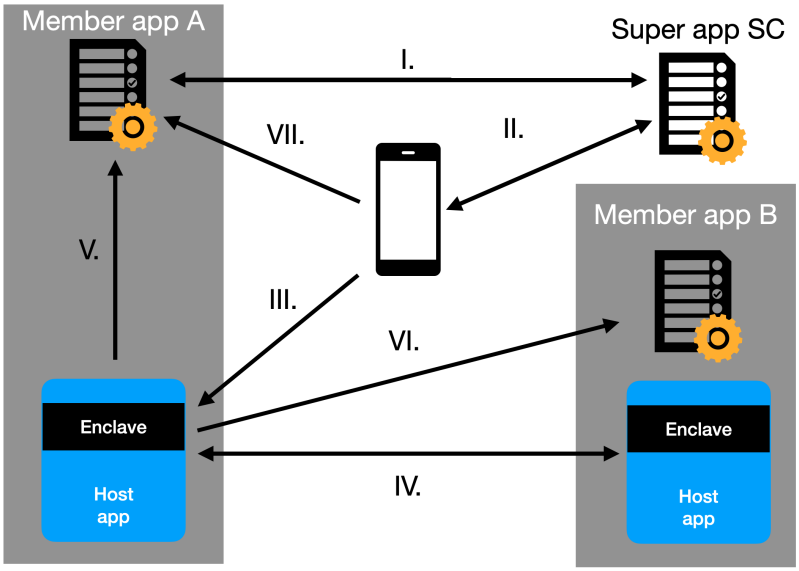


Figure 6.3: Endpoints overview

- POST /data/new: A single data entry point is exposed by this endpoint no matter what sort of data is coming. The Extraction, Transformation and Loading (ETL) is performed inside the enclave.

IV **Interfacing other member app enclave**

- GET /data: By presenting the token, the enclave hit this endpoint to retrieve user's data.

Member Apps Smart Contract

V **Interfacing own enclaves**

- TRANSACTION update report: Members willing to exchange the code must submit an attestation report of the current enclave state to this endpoint.

VI **Interfacing other enclaves**

- CALL list permissions: By hitting this endpoint, the member apps can retrieve a list of data access and sharing permissions granted by users to other member apps. The

member app sends the user identification (UID) and receives back the addresses of the member apps's smart contracts that acts as user's data broker.

- CALL create new token: A third party member hits this endpoint to retrieve a token to access user's data. The smart contract validates if the user has given permission to share his data and then issue a token that will be stored in the blockchain.

VII **Interfacing mobile app**

- TRANSACTION create new permission: Every time an user allows any member app to collect sensitive data, this endpoint registers the permissions.
- TRANSACTION modify permission: An user revoking the permissions will make the super app to hit this endpoint to confirm the deletion.
- DELETE /data/delete: The user request to wipe out his data hits this endpoint.

6.4 Experiments

Firstly, it was implemented the ID anonymisation function of PEM to test the feasibility of using SGX to this end. This function consists in unsealing (refer to 2.2.1) a file bound to a member app, containing its 2048-bit public key (PK) and a random salt of 16 bytes. Before unsealing the data, the enclave should be created and started. In sequence, this pair is hashed SHA-512 with user's original ID (i.e. 64-bit Android ID [5]) as follows:

$$ID_{anonymised} = SHA - 512(ID||PK||salt)$$

In sequence, the experiments focused on evaluating the overhead imposed methods by implementing the proposed architecture. The question to be answered is: What is the burden of adopting this approach when implementing a super app?

[5] https://developer.android.com/reference/android/provider/Settings.Secure.html

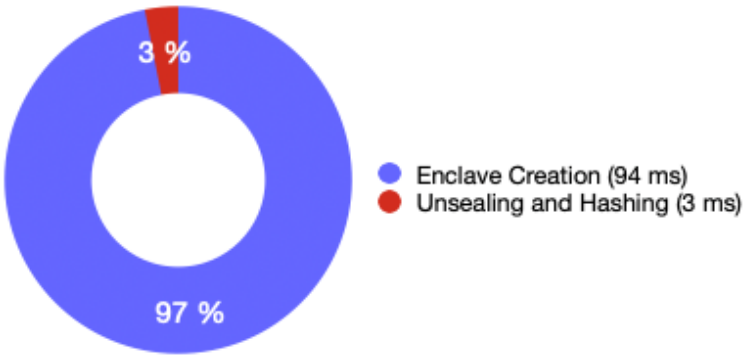


Figure 6.4: ID anonymisation average timings

6.4.1 Testbed

An environment containing the required resources was created in Microsoft Azure. The applications are hosted in Kubernetes cluster 1.23.12 with 2 auto scale node pools with Standard B2ms machines and Standard DC2s v2 machines backed by SGX to run the enclave applications. The enclave applications, developed in Golang and compiled with its version 1.18, are deployed using the Marblerun Framework [115], that provides access to the SGX hardware in a Kubernetes cluster. A Hyperledger Besu blockchain, tested in [41], was deployed in the cluster to reduce the network time and to give us full. The mobile app was developed in React Native version 0.69.3 and tested in an iPhone 12.

6.4.2 Results

By running the ID anonymisation tests 10 times, it could be verified that on average the enclave creation time is far superior than the unsealing and hash functions, yet, the total time can be considered negligible in practical terms, as presented in Figure 6.4. Therefore, it is considered that this part of PEM can be completely implemented inside a SGX enclave.

By evaluating the following bottlenecks, the following questions are addressed:

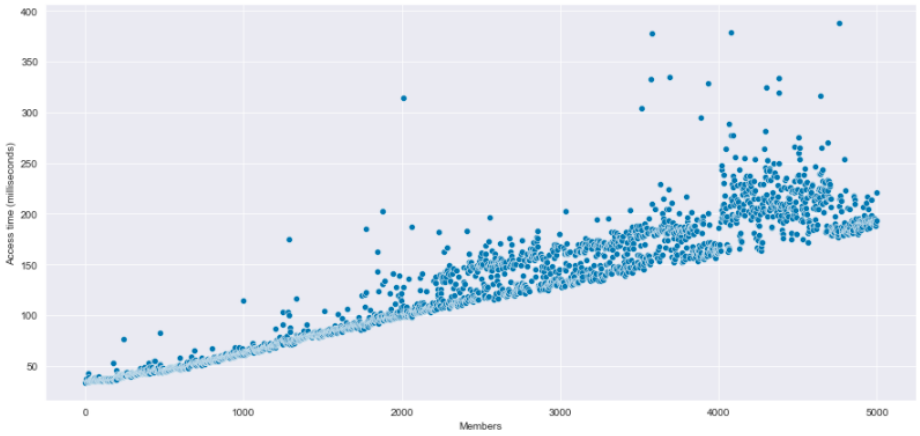


Figure 6.5: Time to retrieve the list with increasing number of members

(1) Retrieve the members list:

the super app instance has to make a Web3 call to the Super App smart contract in order to receive the list of member apps. To measure this call, an instance of the super app smart contract was deployed in Besu followed by the deployment of member apps. Even though it would very unlikely to have this number of member in a real world super app, 5,000 members' smart contracts were deployed and measured incrementally the access time to retrieve the accumulated list of members, illustrated in Figure 6.5. Each data point contains the mean of 10 round trip measures to make the CALL list members to super app smart contract.

(2) Validating the signature from the screen layout:

the main security step of the attested server driver rendering is confirming the signed layout files are issued by the enclave of the member app. The layout files are basically an xml structure to define the components and its disposal in the screen. Therefore, they are usually smaller than 100 kilobytes. However, depending on the complexity it could increase significantly. In this sense, the validation of the signature of up to 10 megabytes is implemented. The signature algorithm used is the ECDSA [64] with the curve secp256k1, which is the pattern for Ethereum that builds up the Besu blockchain. Figure 6.6 illustrates the

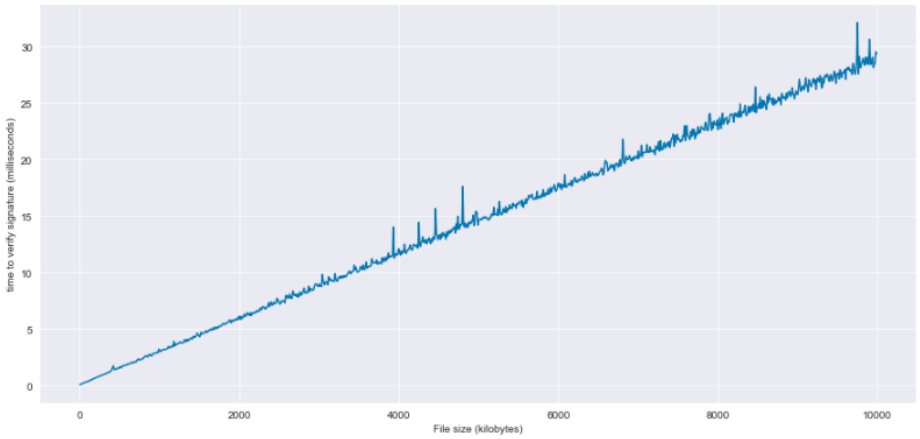


Figure 6.6: Time to validate the signature of the layout file based on its size

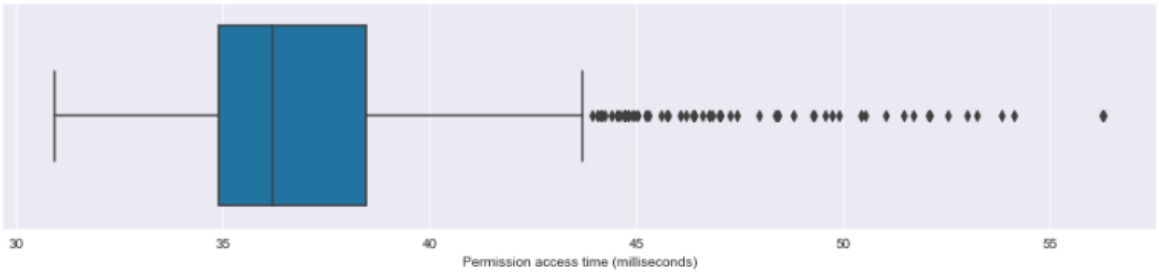


Figure 6.7: Boxplot of the time taken for a member to validate the users permission

raise in signature validation time when the layout file size increases.

(3) Checking users' permissions:

to see where the users' data reside, member apps have to check where his permissions were granted. By querying with the CALL list permissions with the address of each smart contract, a member will see the data already collected and available in other member apps. The aggregated time will vary on the business need of the member, in other words, if it needs data from several other members, then the time will increase as many calls will have to be performed. To understand how this bottleneck could impact the members, 1,000 single requests were measured and a boxplot is drawn in Figure 6.7.

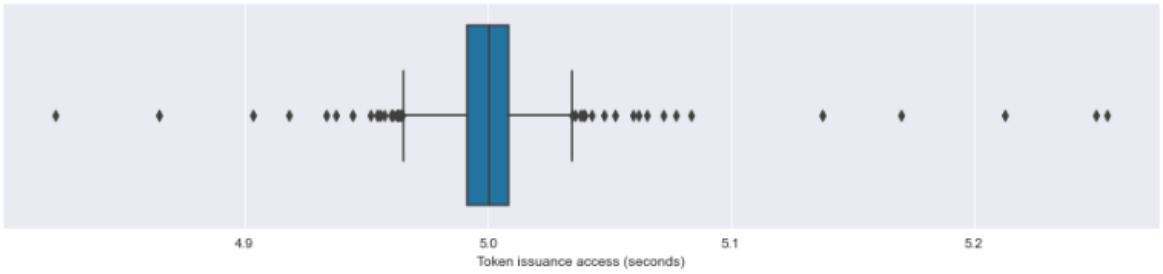


Figure 6.8: Boxplot of the time taken to issue a valid token

(4) Token issuance:

the token that allows a member app to reach the other member's enclave is registered in the blockchain. The implementation extends the token ERC-721 [38] by creating a non fungible token to provide data access with an exclusive uniform resource identifier. To this end, the tokenURI field of ERC-721 is populated with the enclave address concatenated with a generated string that serves as an access path that allows the user data to be consumed by the third party member. As this token issuance is a transaction in the blockchain, it needs a mining time. This transaction emits an event alert when the mining process is completed. The full single token issuance time was measured 1,000 time from the time the transaction was sent to the blockchain and the event was received back, drawn in Figure 6.8.

6.5 Use case

To elicit on how the approach would be used in a real life application, it is presented a use case of an insurance company that needs credit card as input for a model that calculate insurance quotes. In this scenario, the insurance company would be one member of the super app that retrieves credit card statements from the customer's bank, another member of the app. This use case is perfect to elicit the interactions between the stakeholders in the proposed architecture.

The sequence diagram in Figure 6.9 illustrates the message exchange in a successful case between the stakeholders: customer, super app instance, super app smart contract

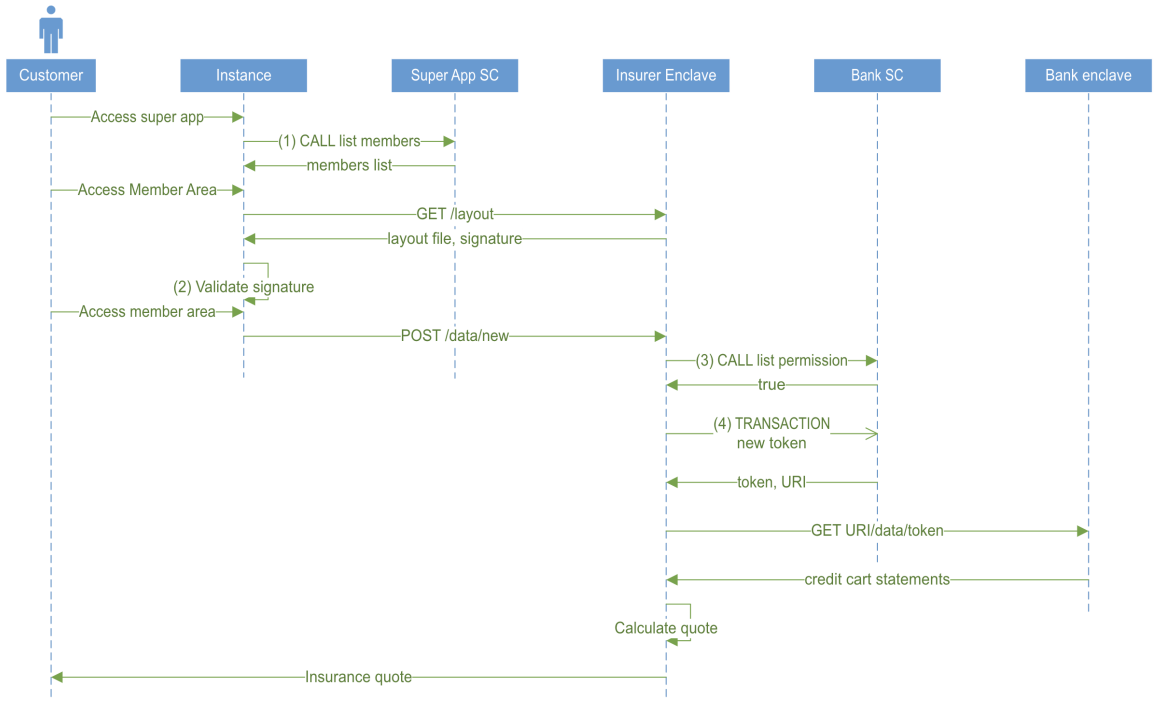


Figure 6.9: Use Case Sequence Diagram

(SC), insurance company (insurer) enclave, bank smart contract and enclave. All the bottlenecks evaluated on the previous step are marked in the figure with the correlation number in parenthesis.

The sequence starts when the customer access the super app to look for a insurance company service. The instance will then retrieve the members list from the super app smart contract. Then, the customer will select the insurer, triggering the first call to the layout files chain.

After inserting the required information, the instance will POST it along with the quote request to the insurer enclave. This enclave will check the customer's permissions by sending a CALL to the members. This means that the customer has information residing at the specific bank. If there are many banks in the super app, the insurer enclave would be querying all of their smart contracts to discover where the referred customer has data. By knowing the list of banks, the insurer enclave makes a TRANSACTION to their smart contract to

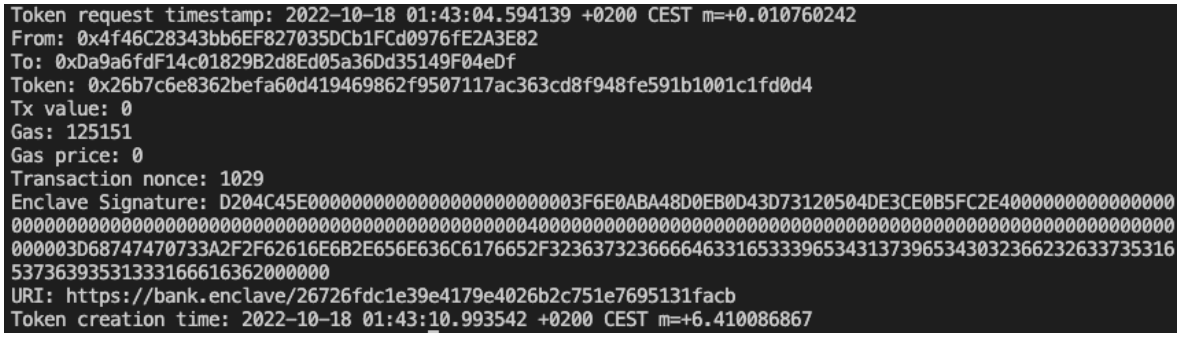


Figure 6.10: ERC-721 Token

request/purchase of a new token to have access to customer's data. After fulfilling the requirements of the bank smart contract, a token is issued, thus, registered in the Blockchain, such as the illustrated in Figure 6.10.

Finally, the insurer enclave request the data to the bank's enclave and process all the data to prescribe the training program. When the authorized enclave access the API, a mutual TLS channel is established, ensuring trust between both parties.

By analysing the timings presented in the previous session, it is possible to understand that the security mechanisms proposed in this work maintain a seamless experience to the user.

The first bottleneck (1) that is retrieving the list of members to super app smart contract is unlikely to exceed 400 milliseconds, unless the number of member scales up in a uncontrolled way. To improve this process, categories of members could filter the members list by affinity, reducing the size of the list queried.

The second issue (2) would be validating the signature of layout file which also proved to be negligible as the processing time requires less than 30 milliseconds for files that are below 10 megabytes, which would be very unrealistic as most of them are usually below 100 kilobytes.

The third concern (3) of querying the users permissions to discover the data could impose a big overhead if there are many members to be queried. The last problem (4) also adds up to 5 seconds to every data that has to be collected from a different member. This would be a real issue if the member app is need to provide a response in near real time,

which can be considered a limitation in the work. This means that the dependency on data exchange between members should be designed to allow a later response for the customers. For the use case, both (3) and (4) would be not an issue because the quote request can be processed asynchronously without any impacts to the customer.

Chapter 7

Conclusion

The thesis focused in answering the three questions from Chapter 1:

1. *How can the backend server guarantee that is receiving data from a legit instance of the mobile application?*
2. *Can the data processing be protected from those with access to the processing platforms in the cloud?*
3. *Can an user share data ensuring proper data governance?*

Ensuring that the connecting entity is either a user with a legit instance of the mobile application or a third party that is identified, and their communications secured and monitored is crucial to raise the bar of control to on mobile mobile application, responding to the question 1.

Section 2.3.2, Chapter 5 and Chapter 4 approaches can be combined to feed a real time monitoring system. Evolutionary, the respective user, environment and action attributes can be evaluated to form a attribute based access control system.

The dashed line represents the protection mechanisms focus of this work to protect against the attacks drawn in squares of Figure 7.1.

Protecting against social engineering attacks is still an open challenge. Behavioural in Chapter 4 extended with user fingerprinting and liveness detection in Chapter 4 covered in

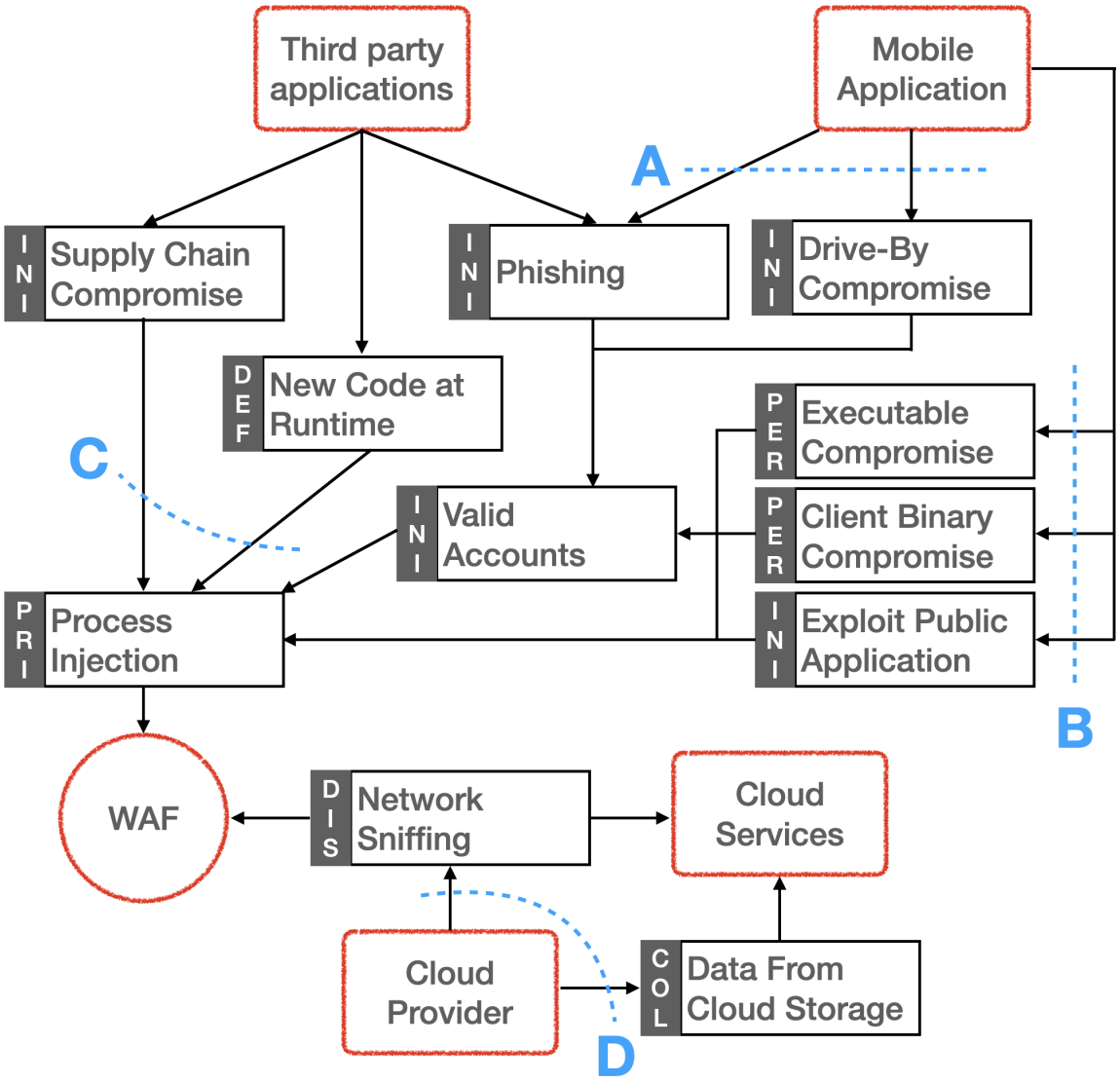


Figure 7.1: Mitigation mechanisms developed in this thesis (dashed blue line)

this work can help mitigating those attacks by reducing the scalability of the attacks, forming the mechanism A of Figure 7.1. Social engineering attacks targeting users of the mobile application are reduced to spear targets.

The prevention from reverse engineering the request aiming to guarantee that the request is coming from a legit instance of the application, which is depicted as B in Figure 7.1, count on 2 additional counter measures. One consists in encrypting the channel end-to-enclave with a 2nd authentication channel presented in Chapter 3. The other is anomaly detection that aims to spot counterfeited requests trying to abuse the APIs, described in Chapter 5.

Notice that trusted third party applications with the consent can freely send requests to the backend, thus, collecting data from the user and creating its own database. The mechanisms proposed by decentralised super app brings transparency to this data collection. This also comes with the protection and transparency on the code which is performing those requests, constituting the mechanism C from Figure 7.1.

Finally, the selected root of trust in the cloud environment enabled the creation of resilient services against an honest but curious cloud provider, addressing the question 2. The implementation of 3 delivers a message and at runtime.

The main product outcome of this thesis is creating a cloud based ecosystem enabling multiple parties to exchange data and services with upgraded CIA triad as a response to the question 3. Confidentiality is enforced to cloud owner resilient level, which means that all the sensitive data is treated in abstractions of memory protected even against someone with physical access to the machines. Integrity is achieved by combining immutable persistence with end-to-end encryption and server-side transaction risk checking. Availability is increased as the whole setup can operate safely in the current cloud service providers.

This work proposed protocol to transfer and process sensitive data in a trusted zone, such as SGX enclave. Our main target is to provide a secure channel to transfer data from a mobile device to a server in which the data can be safely analysed without exposing Personal Identifiable Information not even to the service provider. We proceeded experiments over three implementations of our protocol using three different encryption modes (i.e., CBC,

ECB, GCM encryption). The target is to measure the overhead of using the SGX to create the encryption key for the secure channel. In general the performance impact of key generation is considerably low. Moreover, in our ongoing future work, we are focusing on the security side of our approach. We aim to evaluate the system under different attacks, taking into consideration the possibility of each of the components present in the entire process being compromised.

The work presented in chapter 4 here succeed in automatising the process of detecting interest in a HUE. Our approach can fully map task-oriented mobile apps in a data-driven fashion. The novel HUE mining approach can be extended to other mobile apps domains. Our clustering in combination with scoring techniques are capable of highlighting behavior of users and sessions.

In chapter 5, by combining the Levenshtein Distance and Markov Chain proved to achieve high performances to the task of identifying anomalous sessions in mobile application. The LSTM is computationally costly and in fact degrades the performance of the model in our approach.

The models can be successfully applied to monitoring systems to identify anomalous sessions in real-time, in a similar fashion as presented in [9]. By adding extra features related to the context of the user, such as the device used to connect, the model can work as an attribute based risk analysis.

Further work in the topic should explore the anomalies to try to automatically identify types. Also, additional work to try to reduce the false positives and negatives could directly decrease the need of manual evaluation of the sessions. In the bank sector, this means that it could also avoid transactions to be blocked by automation tools or to efficiently deny fraudulent ones to be executed.

As the future work suggestion in this direction, the combination of the models could lead to real-time setup that can score client sessions and handle the issues discussed in section 4.1.1 in real-time.

Super apps are becoming the new trend in mobile apps development. This new concept brings to the user the possibility of reaching varied services and products aggregate in only

one mobile application. However, the centralised power in hands of the owners of such app brings up real concerns in terms of data privacy. Thus, in this paper we propose a complete auditable decentralised super app architecture to: 1) prevent potential harms from the massive data control from one single owner; 2) provide a full log shared data; 3) control data access between the stakeholders. The experiments proved the feasibility of implementing such approach in real life.

Despite the fact that users tend to give up their privacy in exchange to some benefits, in some cases the privacy loss could be prohibitive, such as medical data processing by an insurance company. Therefore, we understand that the success of this architecture could make new business models possible.

Additionally, the creation of this infrastructure enables the following not exhaustive list of business outcomes:

- SAMS can charge for membership or per session established.
- Member apps can enrich data and sell it to other member apps.
- Users can be refunded for their sold data.
- Token curated registries of users can be sold.
- SAMS can create tokenized data markets of users.

As future work in this topic, a library could be developed to easily create sequence of layout files with attested server driven rendering as well as working in solutions that can enable near real time data analysis.

In conclusion, this thesis presents a comprehensive framework for securing mobile application ecosystems, addressing critical challenges in data integrity, privacy, and governance. Through innovative protocols, encryption techniques, and enhancing real-time monitoring systems, we have established a robust defense mechanism applicable across various societal sectors.

References

[1] Jaime D Acevedo-Viloria et al. “Feature-Level Fusion of Super-App and Telecommunication Alternative Data Sources for Credit Card Fraud Detection”. In: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE. 2021, pp. 1–6.

[2] Jaime D Acevedo-Viloria et al. “Relational graph neural networks for fraud detection in a super-app environment”. In: *arXiv preprint arXiv:2107.13673* (2021).

[3] Zahid Akhtar, Christian Micheloni, and Gian Luca Foresti. “Biometric liveness detection: Challenges and research opportunities”. In: *IEEE Security & Privacy* 13.5 (2015), pp. 63–72.

[4] Moutaz Alazab. “Automated malware detection in mobile app stores based on robust feature generation”. In: *Electronics* 9.3 (2020), p. 435.

[5] Mojtaba Alizadeh et al. “Authentication in mobile cloud computing: A survey”. In: *Journal of Network and Computer Applications* 61 (2016), pp. 59–80.

[6] Ehab M Alkhateeb and Mark Stamp. “A dynamic heuristic method for detecting packed malware using naive bayes”. In: *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. IEEE. 2019, pp. 1–6.

[7] Raja Waseem Anwar, Tariq Abdullah, and Flavio Pastore. “Firewall best practices for securing smart healthcare environment: A review”. In: *Applied Sciences* 11.19 (2021), p. 9183.

[8] Solution Focus Area. “FIDO* Support on Intel® Platforms”. In: (2018).

[9] Riyaz Ahamed Ariyaluran Habeeb et al. “Clustering-based real-time anomaly detection—A breakthrough in big data technologies”. In: *Transactions on Emerging Telecommunications Technologies* (2019), e3647.

[10] Aditya Asgaonkar and Bhaskar Krishnamachari. “Token curated registries-a game theoretic approach”. In: *arXiv preprint arXiv:1809.01756* (2018).

[11] Proff Abbas M Al-Bakri and Hussein L Hussein. “Static analysis based behavioral api for malware detection using markov chain”. In: *The International Institute for Science, Technology and Education (IISTE)* 5.2014 (2014).

[12] Michael Barbehenn. “A note on the complexity of Dijkstra’s algorithm for graphs with weighted vertices”. In: *IEEE transactions on computers* 47.2 (1998), p. 263.

[13] Fabrıcio Benevenuto et al. “Characterizing user behavior in online social networks”. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 2009, pp. 49–62.

[14] Eleanor Birrell et al. “SGX enforcement of use-based privacy”. In: *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*. 2018, pp. 155–167.

[15] Paul Black, Iqbal Gondal, and Robert Layton. “A survey of similarities in banking malware behaviours”. In: *Computers & Security* 77 (2018), pp. 756–772.

[16] Alexey Borisov et al. “A click sequence model for web search”. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 45–54.

[17] Ethan Buchman, Jae Kwon, and Zarko Milosevic. “The latest gossip on BFT consensus”. In: *arXiv preprint arXiv:1807.04938* (2018).

[18] Randolph E Bucklin and Catarina Sismeiro. “Click here for Internet insight: Advances in clickstream data analysis in marketing”. In: *Journal of Interactive marketing* 23.1 (2009), pp. 35–48.

[19] Alan Bundy and Lincoln Wallen. “Breadth-first search”. In: *Catalogue of artificial intelligence tools*. Springer, 1984, pp. 13–13.

[20] *Burp Suite Web Application Security Testing Platform*. URL: https://portswigger.net/burp.

[21] Fernando Kaway Carvalho Ota et al. “A Decentralized Super App”. In: *2023 24th IEEE International Conference on Mobile Data Management (MDM)*. 2023, pp. 81–88. DOI: 10.1109/MDM58254.2023.00024.

[22] Sanjay Chakraborty, Naresh Kumar Nagwani, and Lopamudra Dey. “Performance comparison of incremental k-means and incremental dbscan algorithms”. In: *arXiv preprint arXiv:1406.4751* (2014).

[23] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[24] Ting Chen, Martin Renqiang Min, and Yizhou Sun. “Learning k-way d-dimensional discrete codes for compact embedding representations”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 854–863.

[25] Kathleen Cheng et al. “Emergence of a Post-App Era-An Exploratory Case Study of the WeChat Mini-Program Ecosystem.” In: *Wirtschaftsinformatik (Zentrale Tracks)*. 2020, pp. 1444–1458.

[26] CNIL. *Blockchain and the GDPR: Solutions for a responsible use of the blockchain in the context of personal data*. URL: https://www.cnil.fr/en/blockchain-and-gdpr-solutions-responsible-use-blockchain-context-personal-data.

[27] Katriel Cohn-Gordon et al. “A formal security analysis of the signal messaging protocol”. In: *Journal of Cryptology* 33.4 (2020), pp. 1914–1983.

[28] European Comission. *Staff working document on the free flow of data and emerging issues of the European Data Economy*. URL: https://digital-strategy.ec.europa.eu/en/library/staff-working-document-free-flow-data-and-emerging-issues-european-data-economy.

[29] Stefan Contiu et al. “IBBE-SGX: Cryptographic group access control using trusted execution environments”. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2018, pp. 207–218.

[30] Victor Costan and Srinivas Devadas. “Intel SGX explained”. In: *Cryptology ePrint Archive* (2016).

[31] Thomas M Cover and Peter E Hart. “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27.

[32] D. R. Cox. “The regression analysis of binary sequences (with discussion)”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 215–242.

[33] T Dierks and E Rescorla. “The transport layer security (TLS) protocol version 1.2”. In: *RFC 5246* (2008).

[34] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[35] Saharnaz Dilmaghani et al. “Privacy and security of big data in AI systems: a research and standards perspective”. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 5737–5743.

[36] Saharnaz Dilmaghani et al. “Privacy and security of big data in AI systems: a research and standards perspective”. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 5737–5743.

[37] Arun Donti et al. “A Behavioral Model System for Implicit Mobile Authentication”. In: *Worcester Polytechnic Institute* (2017).

[38] William Entriken et al. *EIP-721: Non-Fungible token standard*. Jan. 2018. URL: https://eips.ethereum.org/EIPS/eip-721.

[39] Christian Esposito, Francesco Palmieri, and Kim-Kwang Raymond Choo. “Cloud message queueing and notification: challenges and opportunities”. In: *IEEE Cloud Computing* 5.2 (2018), pp. 11–16.

[40] Benedict Faber et al. “BPDIMS: A blockchain-based personal data and identity management system”. In: (2019).

[41] Caixiang Fan et al. “Performance Analysis of Hyperledger Besu in Private Blockchain”. In: *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE. 2022, pp. 64–73.

[42] Daniel Fasnacht. “Banking 4.0: Digital Ecosystems and Super-Apps”. In: *Theories of Change*. Springer, 2021, pp. 235–256.

[43] Li Feng et al. “High utility event analysis in large transaction databases”. In: *Data Mining and Knowledge Discovery* 15.3 (2007), pp. 273–298.

[44] Alberto Fernández et al. “SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary”. In: *Journal of artificial intelligence research* 61 (2018), pp. 863–905.

[45] Massimo Ficco. “Comparing API call sequence algorithms for malware detection”. In: *Workshops of the International Conference on Advanced Information Networking and Applications*. Springer. 2020, pp. 847–856.

[46] Massimo Ficco. “Detecting IoT malware by Markov chain behavioral models”. In: *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE. 2019, pp. 229–234.

[47] Michèle Finck. *Blockchain regulation and governance in Europe*. Cambridge University Press, 2018.

[48] Wensheng Gan et al. “A survey of utility-oriented pattern mining”. In: *IEEE Transactions on Knowledge and Data Engineering* 33.4 (2019), pp. 1306–1327.

[49] Elena Gatti and Christina Richter. “WeChat–Die chinesische Super-App”. In: *Digitales China*. Springer, 2019, pp. 23–30.

[50] Anna Georgiadou, Spiros Mouzakitis, and Dimitris Askounis. “Assessing mitre att&ck risk using a cyber-security culture framework”. In: *Sensors* 21.9 (2021), p. 3267.

[51] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: *Neural computation* 12.10 (2000), pp. 2451–2471.

[52] Mike Goldin. “Token-curated registries 1.0”. In: *Medium (accessed 2 April 2019) https://medium: com/@ ilovebagels/token-curated-registries-1-0-61a232f8dac7* (2017).

[53] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[54] Xiao Han and Shuhan Yuan. “Unsupervised Cross-system Log Anomaly Detection via Domain Adaptation”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 3068–3072.

[55] Daniel Hintze et al. “CORMORANT: On implementing risk-aware multi-modal biometric cross-device authentication for Android”. In: *Proceedings of the 17th International Conference on Advances in Mobile Computing & Multimedia*. 2019, pp. 117–126.

[56] Daniel Hintze et al. “Location-based risk assessment for mobile authentication”. In: *Proceedings of the 2016 ACM international joint conference on pervasive and ubiquitous computing: Adjunct*. 2016, pp. 85–88.

[57] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[58] Scott Hollenbeck. *Transport layer security protocol compression methods*. Tech. rep. 2004.

[59] Ling Huang et al. “Adversarial machine learning”. In: *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 2011, pp. 43–58.

[60] Husna Sarirah Husin and Norsuhaili Seid. “Discovering users navigation of online newspaper using Markov model”. In: *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. 2017, pp. 1–4.

[61] Luis-Daniel Ibáñez, Kieron O’Hara, and Elena Simperl. “On blockchains and the general data protection regulation”. In: *EU Blockchain Forum and Observatory*. 2018, pp. 1–13.

[62] Honey Jindal, Neetu Sardana, and Raghav Mehta. “Analysis and visualization of user navigations on web”. In: *Data Visualization and Knowledge Engineering*. Springer, 2020, pp. 195–221.

[63] George H John and Pat Langley. “Irrelevant features and the subset selection problem”. In: *Machine learning* 11.3 (1994), pp. 127–171.

[64] Don Johnson, Alfred Menezes, and Scott Vanstone. “The elliptic curve digital signature algorithm (ECDSA)”. In: *International journal of information security* 1.1 (2001), pp. 36–63.

[65] Leonard Kaufman and Peter J Rousseeuw. “Finding the number of clusters in a data set: An improvement to the Hartigan and Wong algorithm”. In: *Journal of Classification* 7.1 (1990), pp. 145–155.

[66] Ankita Khobragade, Saurabh Suryawanshi, and Sonam Mikhail. “Moving towards a super-app for India: what it takes to be a one stop solution for an Indian netizen”. In: (2018).

[67] Youngjoon Ki, Eunjin Kim, and Huy Kang Kim. “A novel approach to detect malware based on API call sequence analysis”. In: *International Journal of Distributed Sensor Networks* 11.6 (2015), p. 659101.

[68] Jon M Kleinberg. “Authoritative sources in a hyperlinked environment”. In: *Journal of the ACM (JACM)* 46.5 (1999), pp. 604–632.

[69] Kubilay Ahmet Küçük et al. “Exploring the use of Intel SGX for secure many-party applications”. In: *Proceedings of the 1st Workshop on System Software for Trusted Execution*. 2016, pp. 1–6.

[70] Ashish Kumar, Jari Salo, and Hongxiu Li. “Stages of user engagement on social commerce platforms: analysis with the navigational clickstream data”. In: *International Journal of Electronic Commerce* 23.2 (2019), pp. 179–211.

[71] Leslie Lamport. “The Byzantine Generals Problem”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), pp. 382–401.

[72] Vladimir I Levenshtein et al. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.

[73] Fudong Li et al. “Behaviour profiling for transparent authentication for mobile devices”. In: (2011).

[74] Jieling Li, Hao Zhang, and Zhiqiang Wei. “The Weighted Word2vec Paragraph Vectors for Anomaly Detection Over HTTP Traffic”. In: *IEEE Access* 8 (2020), pp. 141787–141798. DOI: 10.1109/ACCESS.2020.3013849.

[75] Yan Li et al. “Seeing your face is not enough: An inertial sensor-based liveness detection for face authentication”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 1558–1569.

[76] Xueping Liang et al. “Man in the cloud (mitc) defender: Sgx-based user credential protection for synchronization applications in cloud computing platform”. In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE. 2017, pp. 302–309.

[77] R Lindemann, D Baghdasaryan, and B Hill. “FIDO Registry of Predefined Values”. In: *Proposed Standard. URL: https://fidoalliance. org/specs/common-specs/fido-registry-v2* ().

[78] He Liu et al. “Software abstractions for trusted sensors”. In: *Proceedings of the 10th international conference on Mobile systems, applications, and services*. 2012, pp. 365–378.

[79] Zhen Liu et al. “A statistical pattern based feature extraction method on system call traces for anomaly detection”. In: *Information and Software Technology* 126 (2020), p. 106348.

[80] Juan Lopez et al. “A survey on function and system call hooking approaches”. In: *Journal of Hardware and Systems Security* 1 (2017), pp. 114–136.

[81] Zhihui Lu et al. “Smartly Deploying WeChat Mobile Application on Cloud Foundry PaaS”. In: *International Conference on Smart Computing and Communication*. Springer. 2018, pp. 345–355.

[82] Mika Mäntylä, Martın Varela, and Shayan Hashemi. “Pinpointing Anomaly Events in Logs from Stability Testing–N-Grams vs. Deep-Learning”. In: *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE. 2022, pp. 285–292.

[83] Moxie Marlinspike and Trevor Perrin. “The x3dh key agreement protocol”. In: *Open Whisper Systems* 283 (2016).

[84] Emma McCLARKIN. *Report on Blockchain: A forward-looking trade policy: A8-0407/2018: European Parliament*. URL: https://www.europarl.europa.eu/doceo/document/A-8-2018-0407_EN.html.

[85] Giovanna Menardi and Nicola Torelli. “Training and assessing classification rules with imbalanced data”. In: *Data mining and knowledge discovery* 28.1 (2014), pp. 92–122.

[86] Collin Mulliner, Nico Golde, and Jean-Pierre Seifert. “{SMS} of Death: From Analyzing to Attacking Mobile Phones on a Large Scale”. In: *20th USENIX Security Symposium (USENIX Security 11)*. 2011.

[87] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: (2008).

[88] Maleknaz Nayebi and Guenther Ruhe. “Optimized functionality for super mobile apps”. In: *2017 IEEE 25th international requirements engineering conference (RE)*. IEEE. 2017, pp. 388–393.

[89] Lucky Onwuzurike et al. “Mamadroid: Detecting android malware by building markov chains of behavioral models (extended version)”. In: *ACM Transactions on Privacy and Security (TOPS)* 22.2 (2019), pp. 1–34.

[90] Fernando Kaway Carvalho Ota et al. “Event-Driven Interest Detection for Task-Oriented Mobile Apps”. In: *Mobile and Ubiquitous Systems: Computing, Networking and Ser-*

vices: 18th EAI International Conference, MobiQuitous 2021, Virtual Event, November 8-11, 2021, Proceedings. Springer. 2022, pp. 582–598.

[91] Fernando Kaway Carvalho Ota et al. “Mobile App to SGX Enclave Secure Channel”. In: *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE. 2019, pp. 258–263.

[92] Fernando Kaway Carvalho Ota et al. “Towards Privacy Preserving Data Centric Super App”. In: *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*. IEEE. 2020, pp. 1–4.

[93] Jayashree Padmanabhan and Melvin Jose Johnson Premkumar. “Machine learning in automatic speech recognition: A survey”. In: *IETE Technical Review* 32.4 (2015), pp. 240–251.

[94] Karl Pearson. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572.

[95] Abdurrahman Pektaş and Tankut Acarman. “Deep learning for effective Android malware detection using API call graph embeddings”. In: *Soft Computing* 24.2 (2020), pp. 1027–1043.

[96] Orit Raphaeli, Anat Goldstein, and Lior Fink. “Analyzing online consumer behavior in mobile and PC devices: A novel web usage mining approach”. In: *Electronic commerce research and applications* 26 (2017), pp. 1–12.

[97] Luisa Roa et al. “Super-app behavioral patterns in credit risk models: Financial, statistical and regulatory implications”. In: *Expert Systems with Applications* 169 (2021), p. 114486.

[98] Luisa Roa et al. “Supporting financial inclusion with graph machine learning and super-app alternative data”. In: *Proceedings of SAI Intelligent Systems Conference*. Springer. 2021, pp. 216–230.

[99] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The annals of mathematical statistics* (1951), pp. 400–407.

[100] Phillip Rogaway. “Evaluation of some blockcipher modes of operation”. In: *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan* (2011).

[101] David Russell and G Gangemi. “Trusted computing base”. In: *Computer security handbook* 1 (2003), pp. 51–79.

[102] Nashad Ahmed Safa, Reihaneh Safavi-Naini, and Siamak F Shahandashti. “Privacy-preserving implicit authentication”. In: *IFIP International Information Security Conference*. Springer. 2014, pp. 471–484.

[103] Fabian Schneider et al. “Understanding online social network usage from a network perspective”. In: *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*. 2009, pp. 35–48.

[104] Ingo Scholtes. “When is a network a network? Multi-order graphical model selection in pathways and temporal networks”. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 1037–1046.

[105] Kristoffer Severinsen, Christian Johansen, and Sergiu Bursuc. “Securing the Endpoints of the Signal Protocol using Intel SGX based Containers”. In: *Security Principles and Trust Hotspot 2017* (2017), p. 1.

[106] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. “Towards trusted execution of multi-modal continuous authentication schemes”. In: *Proceedings of the Symposium on Applied Computing*. 2017, pp. 1444–1451.

[107] Zdeňka Sitová et al. “HMOG: New behavioral biometric features for continuous authentication of smartphone users”. In: *IEEE Transactions on Information Forensics and Security* 11.5 (2015), pp. 877–892.

[108] Gustavo Botelho de Souza, João Paulo Papa, and Aparecido Nilceu Marana. “On the learning of deep local features for robust face spoofing detection”. In: *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE. 2018, pp. 258–265.

[109] Sampath Srinivas, John Kemp, and FIDO Alliance. *FIDO UAF architectural overview*. 2013.

[110] Sampath Srinivas et al. “Universal 2nd factor (U2F) overview”. In: *FIDO Alliance Proposed Standard* 15 (2015).

[111] François-Xavier Standaert. “Introduction to side-channel attacks”. In: *Secure integrated circuits and systems*. Springer, 2010, pp. 27–42.

[112] Jakapan Suaboot et al. “Sub-curve HMM: A malware detection approach based on partial analysis of API call sequences”. In: *Computers & Security* 92 (2020), p. 101773.

[113] Gabriel Suarez et al. “Enhancing User’s Income Estimation with Super-App Alternative Data”. In: *arXiv preprint arXiv:2104.05831* (2021).

[114] Lichao Sun et al. “Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pp. 228–240.

[115] Edgeless Systems. *Marblerun*. URL: https://www.edgeless.systems/products/marblerun/.

[116] Nick Szabo. “Smart contracts: building blocks for digital markets”. In: *EXTROPY: The Journal of Transhumanist Thought,(16)* 18.2 (1996), p. 28.

[117] Robert Tarjan. “Depth-first search and linear graph algorithms”. In: *SIAM journal on computing* 1.2 (1972), pp. 146–160.

[118] Tin Truong-Chi and Philippe Fournier-Viger. “A survey of high utility sequential pattern mining”. In: *High-Utility Pattern Mining*. Springer, 2019, pp. 97–129.

[119] Dalton Cézane Gomes Valadares et al. “Achieving data dissemination with security using FIWARE and Intel software guard extensions (SGX)”. In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2018, pp. 1–7.

[120] Scott Vanstone and Alfred J Menezes. “Handbook of Applied Cryptography”. In: (1996), pp. 233–251.

[121] Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. “An anomaly detection method to detect web attacks using stacked auto-encoder”. In: *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*. IEEE. 2018, pp. 131–134.

[122] Lyan Verwimp, Joris Pelemans, Patrick Wambacq, et al. “Character-word LSTM language models”. In: *arXiv preprint arXiv:1704.02813* (2017).

[123] Shermin Voshmgir. *Token economy: How blockchains and smart contracts revolutionize the economy*. Shermin Voshmgir-BlockchainHub, 2019.

[124] Gang Wang et al. “Unsupervised clickstream clustering for user behavior analysis”. In: *Proceedings of the 2016 CHI conference on human factors in computing systems*. 2016, pp. 225–236.

[125] Yao-Te Wang and Anthony JT Lee. “Mining Web navigation patterns with a path traversal graph”. In: *Expert Systems with Applications* 38.6 (2011), pp. 7112–7122.

[126] Xi Xiao et al. “Android malware detection based on system call sequences and LSTM”. In: *Multimedia Tools and Applications* 78.4 (2019), pp. 3979–3999.

[127] Yang Xin et al. “A survey of liveness detection methods for face biometric systems”. In: *Sensor Review* 37.3 (2017), pp. 346–356.

[128] Ming Yang et al. “Detection of malicious behavior in android apps through API calls and permission uses analysis”. In: *Concurrency and Computation: Practice and Experience* 29.19 (2017), e4172.

[129] Weizhao Yuan et al. “API recommendation for event-driven Android application development”. In: *Information and Software Technology* 107 (2019), pp. 30–47.

[130] Matei Zaharia et al. “Spark: cluster computing with working sets”. In: *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association. 2010, pp. 10–10.

[131] Linghan Zhang, Sheng Tan, and Jie Yang. “Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 57–71.

[132] Linghan Zhang et al. “Voicelive: A phoneme localization based liveness detection for voice authentication on smartphones”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1080–1091.

[133] Zhi-Hua Zhou. “Ensemble learning, Encyclopedia of Biometrics”. In: *doi* 10 (2009), pp. 978–.