Politecnico di Torino

FACULTY OF ENGINEERING
Master's Degree in Computer Engineering

Master's Thesis

Continuous authentication with behaviometrics on smartphones

Candidate:
Gabriele Vassallo

Supervisor:
Prof. dr. Silvia Chiusano

Research supervisor:
Dr. ir. D. Preuveneers

October 2017

Summary

The functionalities offered by smartphones are increasing, along with the demand of stronger security mechanisms. Password based authentication methods for access control are the only defenses that separate illegitimate users to access sensitive information stored on these devices or online services. This thesis research investigates the effectiveness of using keystroke dynamics to authenticate smartphones users continuously and at the same time transparently. By using these techniques, the identity of a user is verified based on his/her way of typing on the device keyboard. We propose a keystroke dynamics authentication framework whose services can be integrated into a contemporary Identity and Access Management (IAM) system to provide continuous authentication capabilities. We implemented privacy-preserving techniques for our solution − i.e. *permutation*, *substitution* and *suppression* − that can be used in order to prevent service providers from reconstructing the text originally typed by users.
We evaluate our solution measuring the authentication accuracy by using real user data. We also measured and compared the impact on this accuracy when the privacy-preserving techniques are applied. The Equal Error Rate (EER) obtained by applying the *permutation* technique remained around 16% for the 'authentication mode' and 18% for the 'anomaly detection mode', the same as the one registered without using this technique. Whereas, by applying the *substitution* technique, we registered an increase of 15% for the first task, and of 11% for the second one. For the third *suppression* technique, the EER increased as much as the number of keystrokes suppressed.
In summary, the key contributions made by this thesis are (1) a server and client side implementation for keystroke dynamics authentication on mobile, (2) the evaluation of state-of-the-art keystroke dynamics techniques using real datasets, (3) the implementation of privacy extensions on top of existing algorithms.

Acknowledgements

I would like to thank all the people who supported me writing my master thesis during these last ten months. First of all, I would like to thank Professor Wouter Joosen and all the people of the Distrinet research group that have steered me in the right direction with their helpful comments and observations during our meetings.
Most of my gratitude goes to my supervisor, Davy Preuveneers, whose wise guidance, week after week, has made possible the realization of this work. His valuable knowledge, as well as his constancy in leading me, pushed me to do my best. It was a pleasure to work with such a professional, interesting and inspiring person.
Furthermore, I would like to thank my Erasmus friends, who brightened my days during these months in Leuven. Special thanks go out to my roommate, Stefano, who was always by my side while facing good and bad times, and to Federica, who kept me going with her endless encouragement.
Finally, I would not be able to have done all of this work without my family in Italy, especially my parents and my sister, who have been able to send me their support notwithstanding the distance.
To finish this preface, I would like to thank my assessors for reading this text.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter provides an introduction of the thesis work. Section 1.1 presents the context and problem statement in order to introduce the reader to the goal of the thesis, that will be described in more detail in Section 1.2. Then, Section 1.3 presents the approach adopted to achieve the stated goal, followed with the achieved results in Section 1.4. The chapter concludes with Section 1.5 giving an overview of the thesis text.

1.1 Context and problem statement

The usage of smartphones and mobile devices during our day-life has become widely common. The capabilities of these devices have increased as well, allowing users to perform disparate kinds of tasks. Rather than simply placing calls, they are used to store personal information, access online sensitive data, e.g. bank accounts, making mobile payments, etc. For this reason, securing wisely these personal devices is vital. The access control mechanism widely adopted relies on simply unlocking them with a PIN code, a passcode, or a locking pattern. For some applications that access remote services on the web, other authentication mechanisms are used. The password based authentication method is the most universally adopted. By using this method a user's identity is verified by asserting the correctness of a secret string. The string has to be known only by the service provider and the specific user.
The two mentioned access control mechanism, the former performed locally on the device, the latter with the cooperation of a remote server, expose the same vulnerability. A malicious user, in order to access an illegitimate device, needs only to know a secret information. This information, that can be a passcode, a PIN, a password or a locking pattern, can be stolen and replicated easily by skilled attackers. This poses a great risk to smartphone users. In addition, these methods of authentication can no longer be used on devices with limited interaction capabilities. For instance, smartwatches does not let users type passwords or PIN codes. These are some of the

reasons that are making growing the interest in alternative authentication strategies. This is the case for what is called *active*, or *continuous*, authentication. With this new paradigm of authentication the identity of a user is continuously verified, possibly after the first login phase where a first validation of the identity is performed by means of traditional security techniques. The assessment of the authenticity of a *post-login* user session is measured using *behaviometric* techniques. These techniques extract patterns from the interaction of the users with the system, and, based on that, they are able to recognize or verify the users identities, without additional user interactions. These characteristics strengthen the authentication mechanism and, at the same time, make it user-friendly thanks to its transparency. Some behaviometrics have already been investigated, such as keystroke dynamics, mouse movement, stylometry, Web browsing, etc. Not all of them could be used with smartphones, due to the different hardware components of these devices. However, some of them still work in this different context. For instance, keystroke dynamics work with smartphones, even if the keyboards are virtual instead of a physical ones. These techniques, in addition with new (developed using the information extracted from the different sensors of these devices) can be exploited in order to build stronger authentication mechanisms.
The drawback of using these techniques is that some privacy concerns could arise. Malicious users, or *honest but curious* service providers, could have access to these behavioral data and exploit this data to possibly harm targeted users. Hence, privacy-preserving techniques should be applied in order to protect the users privacy. In addition, the continuous processing of data, used for authentication on remote servers, may possibly create performance bottlenecks when the number of user is sufficiently high. Therefore, a continuous authentication solution should be able to scale when the number of users increases, to guarantee the reliability of the system.

1.2 Goal

The goal of the thesis is to implement a continuous authentication framework that exploits behaviometrics, capable of working with smartphone devices. The framework should be able to scale easily. For instance, it should allow the instantiation of new resources when the number of users increases. It should also be composable with other authentication mechanisms and other frameworks that make use of behaviometrics to authenticate users.
Precisely, the solution proposed uses keystroke dynamics as behaviometric. The privacy problems of using this behaviometric should be considered in the design of the framework and mitigated. For instance, a common risk of using keystroke dynamics is that the original text typed by users could be reconstructed. For this reason, privacy-preserving solutions have to be investigated and implemented to avoid this attack.

To achieve the presented goal, a server and client side keystroke dynamics authentication framework has to be implemented. After designing the framework, together with the integration with the state-of-practice authentication system, a prototype will be developed that will serve as a proof-of-concept. Lastly, after the implementation phase, some experiments will be carried out, in order to evaluate our solution.

1.3 Approach

The approach followed starts with the presentation of a motivating example, in order to highlight the requirements, both functional and non-functional, of the system that has to be implemented. The example is used to identify the privacy threats that could arise while using such kind of techniques. After that, the analysis of a state-of-practice authentication architecture is presented. Precisely, the system described is the OpenAM one. OpenAM is an open source Identity Access Management (IAM) system. It provides different out-of-the-box authentication methods, along with the possibility of creating new. By taking advantage of this extensibility feature, a continuous authentication framework is designed, that can be integrated with the OpenAM platform. To design and implement such a framework, pre-existing approaches have to be researched in order to follow common design patterns. Once the design of a working solution is presented, along with the choice of which algorithms will be used for the behaviometrics analysis, the integration of the designed framework with the OpenAM system is illustrated. The complete architecture is then subjected to a privacy threat analysis, by using the LINDUNN methodology. The analysis is performed to identify the privacy threats in order to understand which techniques should be implemented to mitigate these risks. These privacy-preserving techniques are then presented. After the design phase, the implementation of a prototype is presented. In particular the prototype is composed of two applications: a server-side application, that provides the continuous authentication service, and a client-side application, that uses this service.
A simple use case scenario of the proposed solution is described as follows. A user can register, or log in, to the system and, after that, he/she can start writing in a apposite message box. For each interaction of the user with the virtual keyboard, the keystrokes are collected and sent to the server that continuously performs classification operations to ensure the authenticity of the user session. The client-application receives the authentication responses from the server. In the case in which it receives negative results it forces the user to log in again by forcing him/her to provide again his/her credentials.
The evaluation of the prototype is carried out by testing the system with data collected from real users. In particular, three different datasets are used for the evaluation, the first two were publicly available, whereas the third is made of samples collected through a self-developed web application. The purpose of the evaluation

is to measure the authentication accuracy. The metrics used for the evaluation are the False Acceptance Rate (FAR) and the False Rejection Rate (FRR). The two metrics measure whether the system is able to correctly accept legal user and reject illegitimate ones. The same experiments are performed also by applying the privacy-preserving techniques investigated. This will highlight how much the authentication accuracy is affected when these techniques are used. To conclude, the validity threats of these experiments are described in order to explain the encountered limitation problems and possible future work.

1.4 Key contributions

The key contributions of the thesis work can be summarized in three main points. They are listed below.

1. A server and client side implementation of keystroke dynamics authentication for smartphones.

2. The evaluation of state-of-the-art keystroke dynamics techniques for computer keyboards using real datasets of smartphone users with virtual keyboards.

3. The implementation and evaluation of privacy extensions on top of existing algorithms.

Regarding the first contribution, the server side application provides the support for the integration with an IAM authentication system. The focus of the work is on strengthening state-of-the-art authentication systems and on mitigating privacy threats that could arise when using behaviometrics for authentication.

1.5 Sections overview

The thesis text is structured as follows. Chapter 2 presents a discussion about some background topics, such as: authentication, multi-factor and continuous authentication, behaviometrics, keystroke dynamics and in the end concepts of privacy and privacy threats. Chapter 3 is about the design and extension of the authentication platform. It illustrates the OpenAM architecture and how it is possible to extend this architecture. It continues by presenting how the continuous authentication framework should be implemented and the algorithms used for the keystroke dynamics analysis, from an high level perspective. It concludes by describing how the complete system should work and the privacy threats that could arise with the proposed system, after having conducted the threat analysis with the LINDUNN methodology. Chapter 4 describes how the prototype has been implemented, whereas Chapter 5 presents the evaluation of the results obtained with the experiments performed.

Chapter 6 concludes with a look into the key contributions of the thesis and it discusses about possible future research work in this area.

Chapter 2

Background and related work

This chapter begins with Section 2.1, explaining the concept of authentication. It takes a look at the different strategies that have been developed in the past years and that are currently in use. In Section 2.2 the concept of behaviometrics is presented, i.e. the usage of behavioral characteristics to authenticate users, and the possible employment of such techniques to provide continuous authentication services. Further, in Section 2.3, a particular behaviometric technique is described, keystroke dynamics, along with the state-of-the-art about the application of these technique for authentication solutions. Then, in Section 2.4, the definitions of privacy are given as well as the privacy threats that could arise using these behaviometric techniques for authentication. Proposed strategies that have been implemented to mitigate the privacy problems are illustrated. The chapter ends with the gap analysis about behaviometrics technologies on mobile devices.

2.1 Authentication

A user has to identify himself/herself and confirm his/her identity on a local operating system or within an authentication server in order to access resources. This two tasks are called respectively **identification** and **authentication**.
During the first phase, the user announces to the system his identity. Then, the authentication phase is performed, in order to verify that the user is truly who he claims to be. Several strategies have been proposed to authenticate a user. The most well-known is the password based method, in which the identification phase is typically performed providing a unique name, the *username*, that has to be specified during the registration time. Along with the username, a secret string of characters has to be sent, the *password*. This method relies on the fact that the password, decided by the user during the registration phase, has only to be known by him/her and the service provider. According to this assumption, the system can confirm the identity of the user by verifying the correctness of these two pieces of information.

This technique is largely adopted since it does not require too much effort, both in implementation by developers and for the users who use it. However, it exposes different vulnerabilities. The first one that arises derives from the nature of the password, which is only a string, a single piece of information that can be copied, reused, leaked or simply guessed. This has led to several types of attacks [36] that pushed forward the research for stronger solutions.
To strengthen the user authentication paradigm, the **Multi-factor authentication** (MFA) access control method has been proposed. It increases the security of a system, constructing the authentication phase of different authentication steps, called *factors*. These factors belong mainly to three different categories: **knowledge** (something users know), **possession** (something users have), **inherence** (something users are). An example from the everyday life is the withdrawing of money. The transaction, in order to be accomplished by the bank, requires first that the user provides something he has (the bank card) and then something he knows (the PIN). Regarding the inherence factors, these factors are associated to personal user's characteristics. The ones that describe the physiological peculiarities of individuals are called biometrics. Examples of these metrics are fingerprints, hand geometry, face recognition, iris recognition, etc. [12]. When a biological characteristic qualifies to be a form of biometrics, it should generally bear the following four properties [24]

- *Universality*: Every person has the characteristic.
- *Distinctiveness*: Any two persons are distinguishable in terms of the characteristic.
- *Permanence*: The characteristic is stable over a period of time.
- *Collectability*: The characteristic can be measured in numbers.

Instead of biological characteristics, behavioral characteristics can be used as authentication factors. These characteristics are named **behaviometrics**.

2.2 Behaviometrics and continuous authentication

"Behaviometrics is measuring human behavior in order to recognize or verify the identity of a person" [23].
Each person has his unique pattern of using an electronic device. For instance, everyone could interact differently with the graphical interface of an application, with different mouse gestures, typing rhythm, etc. More unusual techniques have been studied to measure these particular *behaviors*, e.g. the battery consumption of mobile devices used to model how the users normally interact with their personal devices during the day [26]. Behaviometrics are interesting for the reason that the

combination of these techniques with the ones already used for authentication can improve significantly the level of security and reliability of a system. Plus, they are transparent and non-intrusive for users. They do not require different actions than the usual interaction with the system. The security of these techniques is based on the assumption that it is trickier to impersonate an individual behavior rather than simply provide a passphrase or a stolen credit card. This could not always be the case, very often new techniques let new attacks come out and new threats arise. Despite that, if we consider that these techniques are used along with others, in the context of multi-factor authentication, they still will make life difficult for adversaries.
Besides, the employment of common authentication techniques is no longer feasible on all kinds of smart devices. For instance, smartwatches do not include a keyboard at all. For this reason, authentication methods that require the user to type his/her credentials cannot be adopted anymore. Data collected from sensors of these devices can be used to model the normal user's behavior, in order to identify anomalies that could indicate that something suspicious happened.

Behaviometrics could also be used to modify the authentication paradigm itself. Most of the authentication methods are static, since once the authentication phase is accomplished successfully, then, no more checks are performed during the session. This could create flaws from both the usability and security perspectives. These "weak" authentication strategies could pose great security risks especially to mobile devices, that are frequently stolen. These devices are able to support a vast variety of services. Hence, they could contain personal information, such as private photos, phone numbers, bank account credentials, data that can be valuable for malicious people in case they want to harm someone for any reason. The majority of the applications in the market provide as method of authentication what is named *one-shot authentication*. By using this method, the credentials to access a resource, e.g. an application, are provided by the user only once, then they are stored on the device and reused automatically, without the user interaction. It is obvious that more effective solutions are needed.

In the past decades, the interest in *active* methods of authentication is growing. The objective of these authentication solutions is to continuously authenticate the users during the session. This field is also known as **continuous**, **implicit** or **context-aware** authentication. In order to accomplish that, the authentication system has to monitor constantly the user behavior to identity anomalies. The likelihood of an authentic user session is continuously assessed, thus it greatly increases the complexity of potential intrusions. The advantage of using behaviometrics for continuous authentication is that the system would act transparently, without interrupting the user unless it begins to doubt the person's identity.

Different behaviometrics have already been investigated. A user identity can be recognized and verified by means of disparate techniques, such as: keystroke dynamics, mouse movements (together with display resolution) [11], CPU and RAM used [13], stylometry [7], Web browsing behavior [1], etc. However, not all of them can actually work on smartphone devices, e.g. mouse movements.

Research on keystroke dynamics demonstrated that they can be used with smartphones, even if the keyboards are typically virtual instead of physical ones. In addition, this behaviometric is the one that achieved the best results in terms of authentication accuracy.

2.3 Keystroke dynamics for user authentication

Keystroke dynamics are the information that can be extracted when a person types on a keyboard, both on physical and virtual ones, that is used to model a person's typing behavior. Since each person tends to write in a different way from the others, researchers started to using the keystroke dynamic information to verify, or even determine, the identities of users. The first research dates back to 1980 by Gaines et al. [17]. They carried out an experiment in order to recognize six professional secretaries by analyzing the way they typed three passages of text. Then, in 1996, Obaidat and Soudun [35] for the first time attempted to use this information as user personal identifier.

The typing behavior is defined using the timing information of when each key is pressed and when it is released. The raw measurements are the **Dwell Time** (DT) and the **Flight Time** (FT). The former is the time duration that a key is pressed, the latter expresses the time duration in between releasing a key and pressing the next one. The time spent to type a sequence of keys is also specific for each person, therefore the time duration of combination of more than two keys can be exploited for the analysis.
A lot of work focused on keystroke dynamics for authentication on computer keyboards [10] [28] [41], more recent research focused on mobile devices with physical keypads [8] [9], whereas in the last years the focus moved to touchscreen devices [3] [34] [27].

Several techniques have been implemented to classify users as authentic or impostors. The idea behind these techniques is, to some extent, always the same: during the first stage the system needs authentic data from the user, in order to build his/her typing model. Then, it computes a score, using some algorithms, which expresses the likelihood that he/she is the legitimate user. Lastly, if the score is above a specified threshold, the user is rejected, otherwise, he/she can continue using the application.

Different algorithms have been proposed to perform this analysis, coming from both the machine learning and the statistical analysis domains.

2.3.1 Machine learning algorithms for keystroke dynamics

"Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed". (Arthur Samuel, 1959).
These algorithms can be firstly trained to build a user model, and then they can be used to determine whether the user acts suspiciously, by comparing the new received data with the reference model. Different algorithms have been proposed to perform these operations, that can work either in **classification** or **clustering** mode. In the authentication context the **classification** task has been used for anomaly detection, i.e. classify whether a user is legitimate or not, (**binary classification**) or even for authentication, i.e. to identify from which user the data received came from in order to verify whether the inferred user is the correct one or not (**multi-class classification**). The algorithms that have been tested are: probabilistic modeling (Bayesian Network) [16], decision tree [5], support vector machine [18] and neural network [31].
Clustering techniques instead group data coming from the same user in a cluster, that is a group of elements with same characteristics, and for each data coming from that user they compute the distance from his/her cluster. If the distance is too high, then the user is probably behaving in a unusual way. This could indicate that an attack is taking place. The research has been done applying K-Means [43], K-Star [37] and K-Nearest-Neighbors (k-NN) algorithms [6].

2.3.2 Statistical techniques for keystroke dynamics

There are several statistical techniques that have been used for biometric authentication, including the mean, the standard deviation [39] and the deviation tolerance [14]. However, an interesting technique is the *disorder-based* one proposed by Gunetti and Picardi [20] and perfectioned by Messerman et al. [32]. The work done by the researchers from the University of Turin [20] focused on the analysis of the typing behavior of people while they write freely on a keyboard. They proposed a new measure, called *degree of disorder* (or simply *disorder*), that indicates how differently two people write each combination of keys on a computer keyboard. The disorder technique has been extended even more by analyzing the typing speed of combination of more than two consecutive keys.
This measure is suitable for the analysis of free text since it takes into consideration the relative typing speed of an individual. They gave, in their report, the following explanation: *"The rationale behind R measures (disorder) is that the typing speed of an individual may change along with the psychological and physiological conditions of the subject, but we may expect the changes to affect all the typing characteristics in*

a similar way. A headache can cause the individual to type more slowly than usual, but the relative typing speed of the entered n-graphs will probably remain stable. If the individual normally types the four-graph "wait" more slowly than the four-graph "stop", this is likely to remain unchanged, even with an headache"[20].
This technique will be explained much more in detail in the next chapters.

2.4 User privacy

The term *privacy* has been mentioned frequently in these past years, especially after the revelations of Edward Snowden, former Central Intelligence Agency (CIA) and National Security Agency (NSA) employee. He disclosed the truth about mass surveillance programs that his ex-employers have been doing after the 9/11 terrorist attacks, in order to protect the safety of citizens. The problem was that they were, at the same time, spying and recording everyone's conversations and Internet traffic. Discussions about the correctness of using such extreme techniques raised after these revelations. Even if they are supposed to be applied for Intelligence purposes, for some people they also weaken everyone's privacy. But, what is the correct definition for privacy? There is not a unique interpretation of the term privacy, since it is an abstract and subjective concept.

2.4.1 Privacy definitions

Different definitions have been proposed, each one from a different perspective, they are listed below.
From a legal perspective:

- *"The right to be let alone"* [44]. This definition was given as a response to technological developments (photography, and its use by the press).
- *"The right of the individual to decide what information about himself should be communicated to others and under what circumstances"* [45].

From a social psychology perspective:

- *"The freedom from unreasonable constraints on the construction of one's own identity"* [2].

The common idea behind these definitions is that everyone should be able to decide and control which personal information will eventually be shared and for which purposes. This is because there are obvious risks related to the uncontrolled sharing of sensitive information. Problems such as data breaches of service providers can let bad people have access to everyone's information that is supposed to remain confidential and that only the service provider and the specific individual should

be able to retrieve. Other problems concerning the inappropriate usage of personal information are practices like:

- **profiling**: categorizing people in different classes, based on socio-economical state, health level, race, religion and sexual orientation;
- **discrimination**: the unjust or prejudicial treatment of different categories of people, especially on the grounds of race, age, or sex.
- **manipulation** e.g. **the filter bubble**: "*the intellectual isolation that can occur when websites make use of algorithms to selectively assume the information a user would want to see, and then give information to the user according to this assumption. Websites make these assumptions based on the information related to the user, such as former click behavior, browsing history, search history and location*"[40].

2.4.2 Privacy threats using behaviometrics

There are many methods for adversaries to track and observe user behavior remotely on the web. Techniques such as *website fingerprinting* [21], *persistent cookies* [38], *audioContext fingerprinting* and *canvas fingerprinting* [15] have already been studied and are ready to be used to track users. With keystroke biometrics algorithms, practices like identification of users can be performed even more easily. They could help to track users even if they use tool to avoid the possibility of being tracked. For instance, using Tor[1] to mask the IP address to navigate anonymously on the web, could not be sufficient to achieve the goal of being anonymous. Users would be identifiable by their behavior, that cannot be masked easily.
For these reasons, implementation of continuous authentication systems that use behaviometrics should take into consideration the privacy problem. Solutions are being studied also to preserve anonymity against adversaries who could exploit behavioral information to identify users on the web.

2.4.3 Privacy-preserving solutions for behaviometrics

Privacy aware solutions should avoid the service provider, or whoever has access, legitimate or not, to the data of a particular user to be able to reconstruct his/her behavior. At the same time, the authentication process should guarantee an high level of security. One example of this approach is MACA, *a privacy-preserving multi-factor authentication system* [30]. The researchers proposed, as a solution

[1] https://www.torproject.org

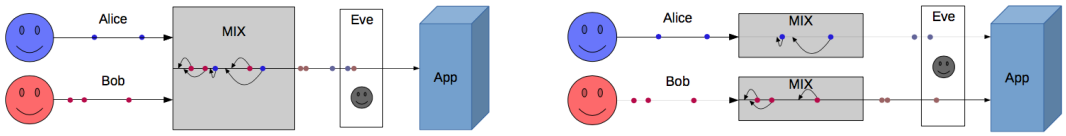
to preserve privacy, the application of *fully homomorphic encryption* (FHE). Using this cryptographic procedure, they could collect the data from different users, encrypt them and then, using the encrypted values, build users profiles in order to verify the authenticity of their sessions. By using FHE the cloud operator does not need to know the exact values of the data received to accomplish successfully the authentication operation. For this reason, the leakage of user data will not be a problem since only the users will have the keys to decrypt it. The results of their proposed scheme were definitely acceptable, in addition both system overhead and resource utilization were within the acceptable range. This could not be obvious, since operations executed with FHE normally were 100 trillion times slower than normal operations. A lot of research has been done in the field of FHE, to speed up the computation. The fact that the computation speed of electronic devices is also increasing will pave the way for the adoption of FHE techniques to develop different kinds of privacy-aware continuous authentication systems.

Another privacy-preserving solution could be to delegate the keystroke analysis to the client component. Behavioral data are collected on the client and then the results of the computation will be sent to the remote server, where the authentication decisions will take place. This could increase the overall performance of the system and, in the meanwhile, it will mitigate the risks of user profiling for the reason that the service provide will not have access to the personal data used for the analysis. This method could only work if the system can trust the results of the computation done in the client component. This could not always be the case. Potential attacks on the client component could subvert this computation easily, either by tampering the data provided to the algorithms or by sending arbitrary authentication messages to the server. A possible solution to ensure the authenticity of the measurements taken on the client is the one proposed by Nauman et al. in [34]. They used remote attestation, a method by which the client authenticate its hardware and software configuration to the server component. The goal of remote attestation is to enable a remote system to determine the level of trust in the integrity of platform of another system [25]. With this solution the server can trust the messages received from a client.

2.4.4 Obfuscation techniques to prevent identification by keystroke dynamics

The work done about exploiting keystroke dynamics for authentication purposes demonstrated that these techniques can be used to identify users by observing their typing behavior. In addition adversaries could exploit this information to impersonate a victim to gain access to a system that implements keystroke biometric access control. Therefore, Monaco et al. proposed two obfuscation strategies to prevent

adversaries from identifying and impersonating users [33]. Both strategies use the Chaum mix. This device is normally used to provide anonymity to users behind a router. In their first solution the Chaum mix has been used by exploiting its capability of reordering packets by introducing random delay to each packet, Figure 2.1a. Doing that, an eavesdropper that observes the sequence of packets transmitted by the mix can not distinguish by which particular user the packets have been originated. This strategy is effective but it requires the cooperation of different users to increase the anonymity level. The second proposed, instead, uses one mix for each user, Figure 2.1b. The packets that record the event generated by the user are sent to the mix. The mix reorders the packets of the single user such that the adversary, even if he/she knows who generated them, cannot reconstruct precisely their order.



(a) Chaum mix (b) User mix

Figure 2.1: Obfuscation techniques for keystroke dynamics (adapted from [33]).

2.5 Gap analysis

The demand of more secure authentication mechanisms has led to the research of valid alternative solutions for authentication. Continuous authentication is one of these solutions. It makes use of behaviometrics to verify continuously the identity of users, strengthening the overall security of a system. Despite this, at the moment there are very few implementations of continuous authentication solutions that are available and integrated in commercial authentication systems. In addition, the solutions proposed did not really face the privacy implications of using such techniques.

This research wants to investigate the effectiveness of using keystroke dynamics as behaviometric to provide continuous authentication to smartphone users. In order to accomplish that, a system has to be designed and developed. This system will be used to test state of the art techniques for keystroke dynamics analysis, in order to verify the authentication accuracy that these techniques achieved for smartphone users. Additionally, privacy-preserving techniques also have to be designed, in order to protect the privacy of the users of this system.

Chapter 3

Designing and extending a state of practice authentication platform

This chapter describes the design of an architecture for a continuous authentication framework, which authenticates users based on their typing behavior. The solution has been designed taking into consideration the privacy problems that could arise using these techniques for authentication.
Section 3.1 presents a motivating example, to give the reader an idea about the possible application of such authentication paradigm. In particular, a chat application use case scenario is presented, along with its requirements. The next section, Section 3.2, analyzes a state-of-practice authentication system, and a state-of-practice keystroke dynamics authentication framework. In Section 3.3, we describe how the analysis of keystroke dynamics is performed, illustrating specifically the techniques presented by Gunetti et al. in [20]. These techniques are explained in detail − even if they are not a research contribution of this thesis − because their understanding is required to comprehend the privacy enhancements researched, which are presented in this section as well and are a key contribution of the thesis. The examples provided to describe these techniques were also taken from the original research. Further, Section 3.4 describes the integration of the keystroke dynamics authentication framework with a state-of-practice authentication system. Section 3.6 concludes with a discussion about the arguments presented in this chapter.

3.1 Motivating example

This section illustrates a use case scenario, to give the reader an idea about the employment of continuous authentication in a real application. In particular, a chat application has been chosen, since the interactions of the users with the application are mostly based on writing, using the devices' keyboards. In addition, once the application is compromised, an attacker could use it to impersonate legitimate users,

or to access their private information, e.g. conversations. It has to be said that the same reasoning applies to other kind of applications, such as mail clients.

3.1.1 Chat application

Sending messages and making phone calls are the reasons for which mobile phones were conceived. The spread of Internet on large scale, along with the increasing capabilities that mobile phones have gained, have produced the diffusion of a large number of chat applications. SMS are no longer used, messages are sent over the Internet. During the last years, companies developed chat applications focusing also on security and privacy of users. Techniques, such as *end-to-end encryption*, at the moment, are implemented in most of the more used applications, in order to avoid attackers to gain access to private conversations. The other main reason is that privacy concerns are rising as well. Service provides could sell user information to advertising companies, or they could let government agencies retrieve everyone's sensitive data. The security precautions already implemented can prevent remote attackers to steal information, but they do not mitigate the threat of an illegitimate access to this information by people who can have access physically to the device. To mitigate this last mentioned risk, smartphones adopt a traditional access control mechanism: before using the phone, a user needs to unlock its screen with a password, with a lock pattern, or with biometric techniques, like fingerprints or face recognition. With only these techniques a malicious users could manage to gain access to the device via peeping or smudge attacks, i.e. observing oily smudges on the screen to reconstruct the password or the lock pattern.

At the moment, this access control mechanism is the only security precaution that prevent attackers to access a chat application. This is because no other authentication techniques are implemented, not even the simple password based one. Adding a static authentication, by forcing the user to provide his/her username and password, every time he/she has to unlock the phone to read a new message, could probably increase the security of the application, but it would not be very friendly for users. Moreover, a password based authentication method would suffer of the same problems that afflict the access control methods used to unlock the device.

The solution for this problem could be to authenticate continuously users during runtime, but current smartphones, normally, cannot do that. Monitoring continuously the user typing pattern, in order to assess whether he/she is the valid user, could be a possible countermeasure against device intrusions. The task could be performed transparently, without extra hardware requirements. The authentication process could be carried out by the interaction of an IAM (Identity and Access Management) system, with the operating system, or with the application itself. With

this solution the user typing pattern is analyzed and after the first stage, during which the user template is built. If a certain number of anomalies is registered, the supposed impostor will be locked out of the application. This will ensure that only the legitimate user can use his/her own phone. Furthermore, it will provide a mutual authentication for users, since they will be ensured that the person to whom they are talking to is continuously authenticated as well. Obviously this solution will not solve all the security problems. The adversary could observe the user typing behavior and try to mimic it. However, adding this new layer of authentication, will increase the overall security of the application.

By using these techniques some privacy concerns could arise. The authentication service provider could possibly have access to the typing information of the users. In the scenario of the chat application the typing information is basically the text typed by the users. Hence, the service provider could have the possibility of reading users conversations. Besides, if an attacker can break into the server, he/she will possibly access everyone's private conversations as well. Therefore, the continuous authentication solution should be implemented taking into consideration these problems.

3.1.2 Requirements

The requirements for a continuous authentication framework, that can be integrated into a chat application system are divided in **functional** and **non-functional**. Regarding the **functional requirements**, the system should:

- let users register and log into the system with a password based mechanism;
- analyze user interactions while they type and extract their typing behavior;
- continuously authenticate users verifying their typing behaviors.

The identified security and privacy threats can be translated into **non-functional requirements**, that are:

- the system has to block illegitimate users in a reasonable time;
- the system has to be scalable to prevent performance bottlenecks;
- the system has to avoid service providers from being able to reconstruct the text typed by users.

3.2 State-of-practice architectures

The building blocks that implement a continuous authentication solution have already been designed and implemented. In this section a description of the two main components is presented. The next section proposes their possible integration. In particular here the focus is on how an Identity Access Management (IAM) is designed, and how it works. Then, we will illustrate the design of a keystroke dynamics authentication framework, which acquires, models and classifies users typing behavior. This second component should be able to scale when it is required and it has to be integrated with an IAM system or with other authentication systems.

3.2.1 Identity Access Management systems

An Identity Access Management system is a set of technologies that are used for business processes for identity management. An IAM system can be used to capture, record and manage user identities. It is also used to manage their access permissions to resources. All these operations are performed in an automated fashion. Different open source IAM systems are available. Three of them have been considered for this research: Gluu[1], Keycloak[2] and OpenAM[3]. After an analysis of these systems, it turned out that OpenAM is the most mature IAM system. It is widely used in industry and it provides extensibility capabilities that let developers personalize the system easily, by letting them script their own authentication procedures. The other two have been discarded since Gluu is less flexible to extend, whereas Keycloak is a recent initiative from RedHat, hence it is a fast moving target and still immature. For these reasons OpenAM has been chosen as reference IAM system. The OpenAM system architecture is presented below.

3.2.2 OpenAM system architecture

OpenAM is an open source, centralized access management solution, that provides authentication, authorization and web security, in a single, integrated solution.
Figure 3.1 presents the OpenAM system architecture. The two main components are:

- **OpenAM core server:** The OpenAM core is composed by the OpenAM framework, OpenAM services and the OpenAM SPIs (Service Provider Interface). The OpenAM SPIs let developers to extend the OpenAM services, with the implementation of plugins that can be integrated in the OpenAM system.

[1] https://www.gluu.org

[2] http://www.keycloak.org

[3] https://www.forgerock.com/platform/access-management/

- **OpenAM Client SDK/API:** OpenAM provides client APIs that can be used by developers in order to integrate the OpenAM services, for authentication and authorization, within their applications.

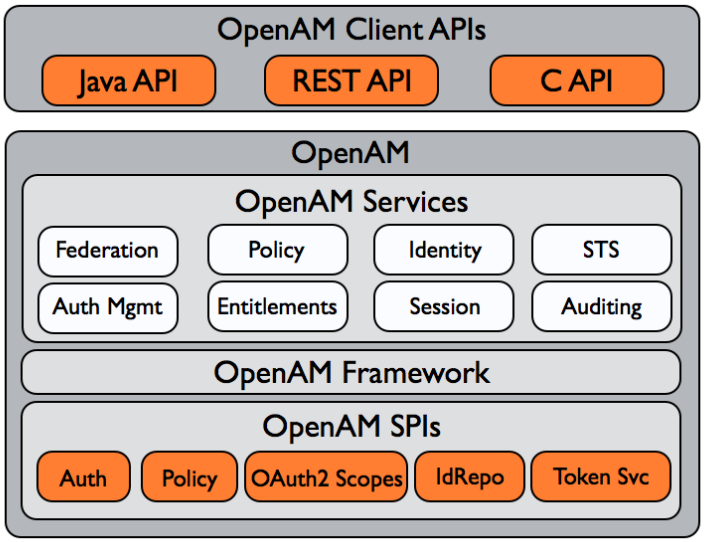


Figure 3.1: OpenAM high-level architecture[4].

By using the OpenAM SPIs it is possible to create an authentication chain. This chain is a sequence of authentication operations, that are performed in order to successfully authorize a user. The chain is composed of different modules, each module can be configurable. A logical authentication chain is shown in Figure 3.2. The modules in the chain are: password-based authentication mechanism module, a device fingerprinting authentication module, a one-time password (HOTP) authentication module, and so on. The chain can be extended as much as it is required. In this example the modules are configured as *Sufficient* or *Requisite*. In particular, the first module is specified *Requisite*, this means that if the first authentication operation is not accomplished successfully, the user will not be successfully authenticated to the system. On the contrary, if the user successfully logs in, then the authentication can continue by executing the second module. The second module is defined as *Sufficient*, so if the authentication is successful, the HOTP will not be triggered. Otherwise, the third module will be executed and the authentication chain will continue until it reaches its end.

[4]https://backstage.forgerock.com/docs/openam/13.5/dev-guide

A module can define and implement the authentication mechanism itself, or it can call an external service that already implements this capability and provides APIs to expose the authentication mechanism.

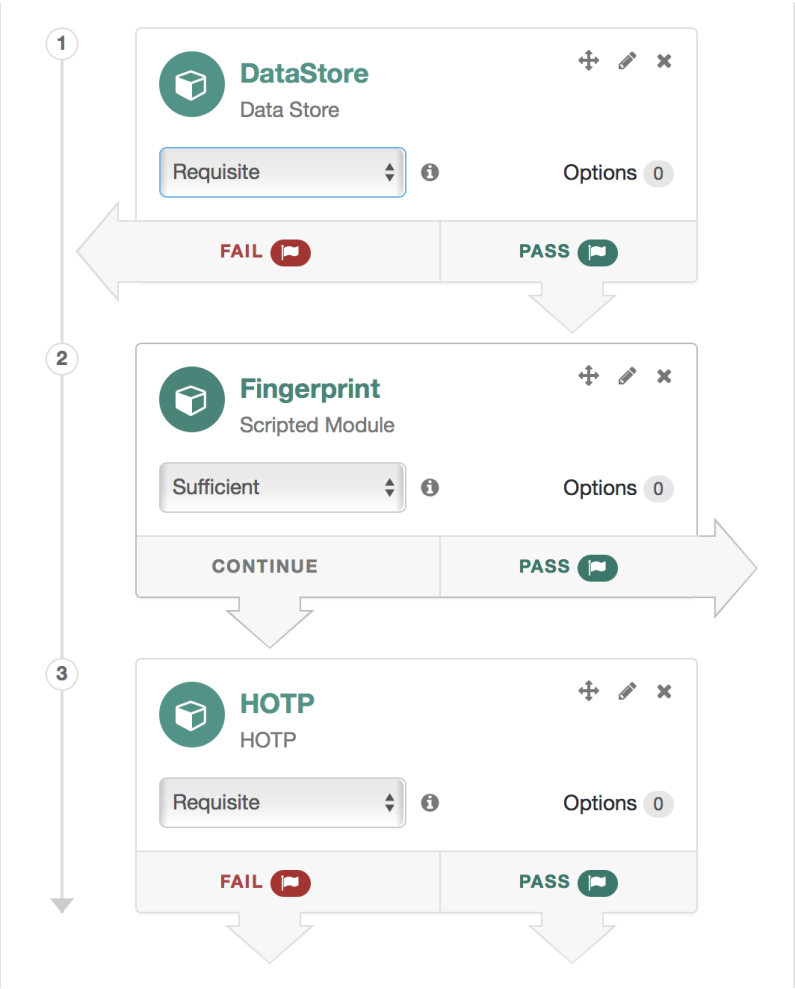


Figure 3.2: An example of an authentication chain in the OpenAM system.

3.2.3 A keystroke dynamics authentication framework

The task of authenticating users by analyzing their typing behavior is performed in different steps, that are showed in Figure 3.3. First of all, two different phases, within the execution of this framework, have to be distinguished, that are the **training** phase and the **classification** phase. For both stages the first operations are the same: data acquisition, data preprocessing and feature extraction.

Data acquisition The first process is performed in order to acquire raw data from the user. The task is normally performed continuously during the user session, or possibly it can be performed in a specified period of time. During the data acquisition the events generated by the interaction with the device are collected and stored.

Data preprocessing The data preprocessing process is carried out in order to transform raw data in an understandable format. Techniques such as outliers detection and removal are performed within this process. The goal is to produce data that is valuable for the analysis.

Feature extraction During this operation the system identifies and extracts from the data the distinctive features common to a user. This data are intended to be informative and non-redundant. They will be used later for the training of the model, i.e. to build the template of the user.

Then, if the system is acting in training mode, the **training of the model** will be performed. Otherwise, once the model is built, the **classification process** will take place.

Model training The model training, or template generation, is the operation carried out to transform the data into a compact form, that represents the user keystroke dynamics characteristic. The learning algorithm finds patterns in training data and it outputs a model that captures these patterns.

Classification process The classification process is the crucial operation. It is responsible of comparing the new incoming data, produced by the user during the sessions, against the generated templates. The operation has to produce as a result a matching score that will be used for the decision making operation.

Decision making This operation is carried out once the matching score is generated by the classification algorithms. The decision is made comparing the matching result, that expresses the similarity or the dissimilarity between the input data and reference models, against a predefined threshold.

The **retraining** of the model eventually can be carried out. The latest keystrokes could be used to retrain or update the reference template. This operation is performed since the user typing behavior will probably change over time. The system after successful logins should register the gradual changes of the user typing behavior.

The keystroke dynamics authentication system can be implemented in different modes of operation. The next session describes three of them: authentication,

identification and classification.

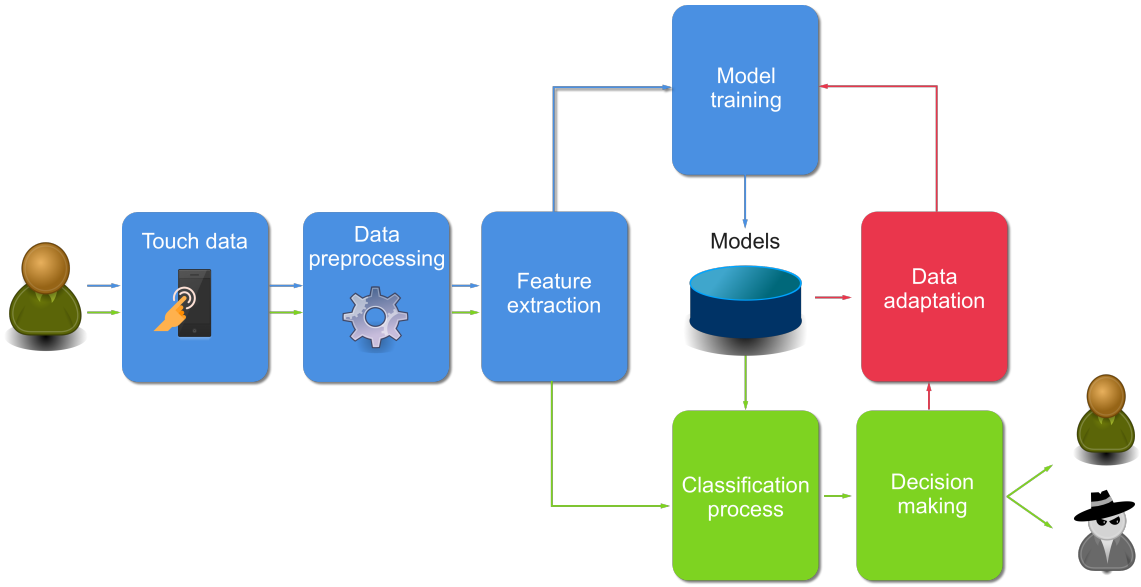


Figure 3.3: A keystroke dynamic authentication framework design.

3.2.4 User authentication, identification and classification

Within biometrics, and then behaviometrics, the distinction between user authentication, identification and classification is often made. With user **authentication**, called also verification, a typing sample is provided to the system, along with a declaration of an identity. Then, the system has to assert whether the sample comes from the user, whose identity has been declared, or not. For the **identification** (or recognition) task the situation is slightly different. The system only sees the sample and it has to decide to which user it belongs. Eventually, it could decide that the sample does not belong to any user known to the system. Finally, **classification** is similar to identification, with the only difference that the sample provided comes only from a user known to the system. The operation has to decide from which user it comes from. Unfortunately, this last approach could not always be useful. Often attackers may come from the outside. Thus, their typing patterns may be completely unknown to the system.
The three modes of operations, ordered by increasing difficulty, could be listed as follows. When a new typing sample X is submitted to the system:

- **Classification:** X comes from a known user. The system has do decide from which user it comes from.

- **Authentication:** X is provided along with an identity. The system has to confirm if it belongs truly to that user. If it does not, it may possibly define whether it comes from a known user or from an external attacker.

- **Identification:** X is presented to the system, that has to identify from which user it comes from. It can produce two possible answers: X belongs to user U; or X belongs to someone unknown.

Using these modes, the outcome of the classification task is defined by observing which of the stored models matches better the sample X received from a user U. These techniques need a pool of models which hold the typing habits of different users, in order to carry on with the comparison tasks. If the comparison would be performed only between the sample from the user U with his/her only model, it would, in case of attack, incorrectly predict a wrong outcome. In the situation in which, for any reason, no other users data is provided for the analysis, the an anomaly detection mode would be more suitable.

3.2.5 Anomaly detection

An anomaly detection operation (also known as **outlier detection**) is responsible of the identification of items, in this case typing samples, that do not conform to an expected pattern, the user model. In the context of keystroke dynamics, the idea is to find a distance metric that can express how far the new received samples are from the user model. Once the metric is established, it can be used to compare the distance of each typing sample, against a specified threshold. If the distance is above the threshold, then it implies that an anomaly has been verified. This could simply be identified as a normal deviation from the normal user typing behavior, or, in case it continues to occur, it would result as a clear sign of intrusion.

3.3 Feature extraction and processing techniques

This section describes, from an high-level perspective, the different distance measures, that have been proposed by Gunetti et al. in [20] for the analysis of keystroke dynamics for free text input. These techniques, even if they are not the contribution of these research, are explained, since the privacy extensions proposed have been built on top of them. In the subsection 3.3.1 is explained which features are relevant for the analysis. In Subsection 3.3.2 the data types that represent these features are presented. Subsection 3.3.4 describes the so called “R” measure (or disorder), whereas Subsection 3.3.5 describes the “A” measure (or similarity).

3.3.1 Timing feature

As has been described in 3.2.3, in a keystroke dynamics authentication framework, the system has to extract from the raw data collected relevant and unique features. These features may be of different types, such as timing, spatial and motion features. For this analysis the focus is on the timing features. The raw data is usually in the form: key, direction and timestamp. This combination expresses when a particular key has been pressed, or released, on a (virtual) keyboard, and in which precise moment the event took place. The two most widely used timing features, that can be extracted from these data, for keystroke dynamic biometrics, are the **Dwell Time (DW)** and the **Flight Time (FT)**.

Dwell time The dwell time expresses the time duration of a touch event with the same key. It is also known as duration, or hold time.

Flight time The flight time is also known as latency. It describes the time duration between two consecutive keys. There are four possible variations of FT. Each of that is a different combination of the two possible events *press* and *release* of the two consecutive keys. The four possible calculations of the FT are described in Figure 3.4.

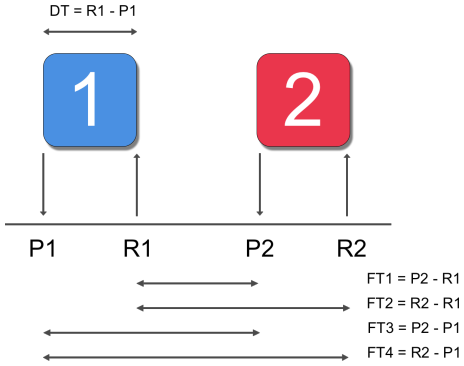


Figure 3.4: Combinations for the flight time timing feature.

3.3.2 The n-graph data type

A timing feature can be extracted with different timing feature lengths. The shorter timing feature that could be extracted is the *unigraph*. It is the timing feature that expresses the dwell time, the time duration when a key is pressed. More interesting timing feature lengths could be analyzed. The *digraphs* represents the flight time time between two consecutive keys. This can be generalized defining a *n-graph* as

the elapsed time between the pressure of the first and the nth (the last) key of a sequence of typed keys. Normally *n-graphs* of short size have been used. *Digraphs* and *trigraphs* are the ones that produce higher accuracy performances. This has been proven by Giuffrida et al. in [19]. In some cases, only the *digraph* representation is considered for the analysis.
The *n-graph* data type has been adopted due to its high expressiveness. It can be used to express different timing feature lengths easily. For instance, starting from a n-graph representation of a typing sample is straightforward to extend this representation in terms of (n+1)-graphs. In addition, shuffling the order of a sequence of *n-graphs* in a typing sample does not alter substantially the results of the comparison with another typing sample. This last characteristic can be exploited to implement privacy-preserving techniques.

3.3.3 Choosing a statistical approach

Given a typing sample, a representation in terms of *n-graphs* can be extracted. Then, the sequence can be analyzed by some algorithms in order to verify whether the typing sample belongs to the legitimate user. A choice was made to decide which algorithms could have been used to perform this verification. The statistical approach proposed by Gunetti et al. [20] seemed to be interesting and extensible. They achieved good results for the keystroke analysis of free text typed on computer keyboards. From a scientific perspective, it was interesting to test whether their techniques would work specifically with smartphone devices. The next two subsections describe how a distance measure between two typing samples is calculated, combining what the researchers called the "R" (standing for "Relative") and "A" (standing for "Absolute") measures. The examples proposed for the explanation of the two measures are taken from [20].

3.3.4 The "R" measure

The idea behind the R measure, is that people tend to write combinations of keys in a relative different way. For instance, user U1 may type the sequence *ab* faster than the one composed of the keys *cd*. While another user U2, may type the second sequence faster than the first one. These differences can be analyzed in order to compute a score that expresses how two typing samples have been typed differently. To give the reader an example, we can consider two typing samples **E1** and **E2**, produced by entering the text: *authentication* and *theoretical* respectively. The two samples may be expressed in the following way, where each letter is preceded by the time, in milliseconds, in which it has been pressed:

E1: 0 **a** 180 **u** 440 **t** 670 **h** 890 **e** 1140 **n** 1260 **t** 1480 **i** 1630 **c** 1910 **a** 2010 **t** 2320 **i** 2600 **o** 2850 **n**
E2: 0 **t** 150 **h** 340 **e** 550 **o** 670 **r** 990 **e** 1230 **t** 1550 **i** 1770 **c** 1970 **a** 2100 **l**

To calculate the *degree of disorder*, first of all, the *digraphs* can be extracted from the two samples. The *digraphs* in common between E1 and E2 are then stored in two different arrays, and ordered by their time duration. The disorder can be computed as the sum of the distances between the position of each element in the two arrays. As an example, the disorder between two arrays A=[2,5,1,4,3] and A'=[3,1,2,5,4] will be computed as:(2+2+1+1+4)=10. Two array with the elements in the same positions have 0 as disorder, whereas if one of the two has the elements in reverse order we will have the maximum disorder, that is given by the formula:

$$|A^2|/2 \text{ (if |A| is even);} \qquad (|A^2| - 1)/2 \text{ (if |A| is odd);}$$

where |A| is the length of the array. We can normalize the disorder of A by dividing it by the maximum disorder. The normalized disorder of the array A is: (2+2+1+1+4)/[(5^2-1)/2]=10/12=0.8333.
If now we come back to the two typing samples **E1** and **E2**, we can compute the disorder as shown in Figure 3.5. Hence, the degree of disorder of the two samples, considering only *digraphs*, that is expressed as R_2(**E1**,**E2**), is calculated as (2+0+2+3+1)/[(5^2-1)/2]=8/12=0.6666.

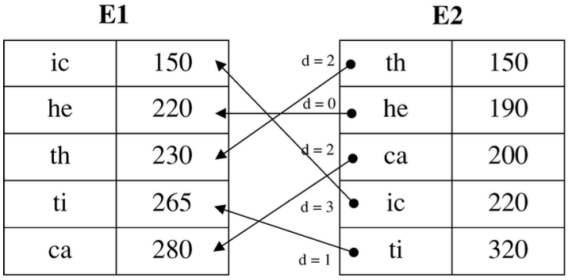


Figure 3.5: Computation of the disorder of the two typing samples E1 and E2.

This technique can also be applied using *trigraphs*, instead of *digraphs*, or with even longer *n-graphs*. The only constraint is that the two typing samples have to share enough *n-graphs* to compute R_n. Moreover, the values of the different R_i can be combined in order to produce a single measure. For instance, if two typing samples share N *n-digraphs* and M *m-digraphs*, with M > N, we can compute:

$$R_{m,n}(\mathbf{E1},\mathbf{E2}) = R_m(\mathbf{E1},\mathbf{E2}) + R_n(\mathbf{E1},\mathbf{E2}) * \mathrm{N}/\mathrm{M}$$

The resulting distance is the sum of the two distances, R_m and R_n, weighted by the different number of *n-graphs* and m-graphs they share.

This R measure does not take into consideration the absolute typing speed of an individual *n-graph*. Two typing samples may have a disorder equal to 0, even if all the *digraphs* of one sample are twice as fast than the ones of the other sample. The idea behind this measure is that, in different conditions, even if users may type differently in terms of speed, they would type still faster a certain combination of keys with respect to another that usually is typed more slowly.

3.3.5 The "A" measure

The A measure, also called *similarity*, takes into consideration the absolute typing speed of the different combinations of keys. Every user writes each n-graph with a certain speed. The measurement is performed comparing the typing speeds of each pair of identical *n-graphs* of two typing samples, and observing how many *n-graphs* are typed differently. To compare two *n-graphs*, if we indicate with d1 and d2 the time durations of the same n-graph, the two *n-graphs* are recognized as similar if $1 < max(d1, d2)/min(d1, d2) \leq t$, for a constant t greater than 1. The A distance between the two typing samples **S1** and **S2** w.r.t the n-graph they share, for a certain value t, is defined as:

A_n^t(**S1**,**S2**) = 1 - (number of similar *n-graphs* between **S1** and **S2**)/(total number of *n-graphs* shared by **S1** and **S2**)

As a consequence, the A measure can assume only values between 0 and 1. The value 0 means the all the pairs are similar, while 1 means that they are completely different. For instance, in the case of the two samples **E1** and **E2**, with t=1.25 we have:

E1		**E2**	
280	**ca**	200	(280/200 = 1.400)
220	**he**	190	(220/190 = 1.157) (similar pair)
150	**ic**	220	(220/150 = 1.466)
230	**th**	150	(230/150 = 1.533)
265	**ti**	320	(320/265 = 1.207) (similar pair)

The A measure is then computed as: $A_2^{1.25}$=1-2/5=0.6. Different A measures for *n-graphs* with different n length can be combined, like we saw with the R measure. The formula is strictly the same:

$$A_{m,n}^t(\mathbf{E1},\mathbf{E2}) = A_m^t(\mathbf{E1},\mathbf{E2}) + A_n^t(\mathbf{E1},\mathbf{E2}) * \mathrm{N}/\mathrm{M}$$

The combination of the $A_{m,n}^t$ and the $R_{m,n}$ measure can be then preformed in order to express more precisely the distance between two typing samples.

3.4 Integration of the privacy-preserving authentication framework with the OpenAM system

The keystroke dynamics authentication framework can be designed as a framework that works independently. It can also be designed as a system that can possibly be called by external authentication systems. By having this design in mind, a solution that can fulfill these requirements can be designed and implemented. It has to provide some APIs that can be contacted by external components in order to perform the authentication operations.

OpenAM, as it has been described in the Subsection 3.2.1, can be extended by means of programmable authentication modules. A module that can call the continuous authentication service can be developed, and executed when it is required by the OpenAM system, to perform continuously the verification of the authenticity of a user session.

The design for this integration is shown in Figure 3.6. OpenAM will be responsible for the management of user identities. It would execute different authentication operations, specified in its authentication chain. After the first login stage, it would call the continuous authentication service. The system that furnishes the CA service has to perform different tasks. It has to collect the keystrokes from a users thanks to the interaction with the mobile device, through an application component that runs on the device. This component will be responsible for: collecting the keystrokes, performing privacy-enhancing techniques on them, and finally sending the processed data to the CA system (that runs on a remote machine). When the CA service will receive the data, it will store the user keystrokes and it will perform the authentication operation. Then, it will send a response to the IAM. The IAM is responsible for deciding whether the user can continue to use the application or not, based on the authentication messages received from the CA component.

The design of the authentication solution should also address the scalability requirement. With the solution proposed there could be different possible techniques to scale out the system to prevent that computation overheads slow down the authentication operations. These techniques are described in the next chapter, after the presentation of the how the system has been implemented. The main idea behind them is that a component of the system should regulate the resource utilization, and in case a certain threshold is exceeded, new resources will be allocated to decrease the overall computation overhead. A load balancer could be used to distribute the traffic over the different instances of the system, as it is shown in Figure 3.7.

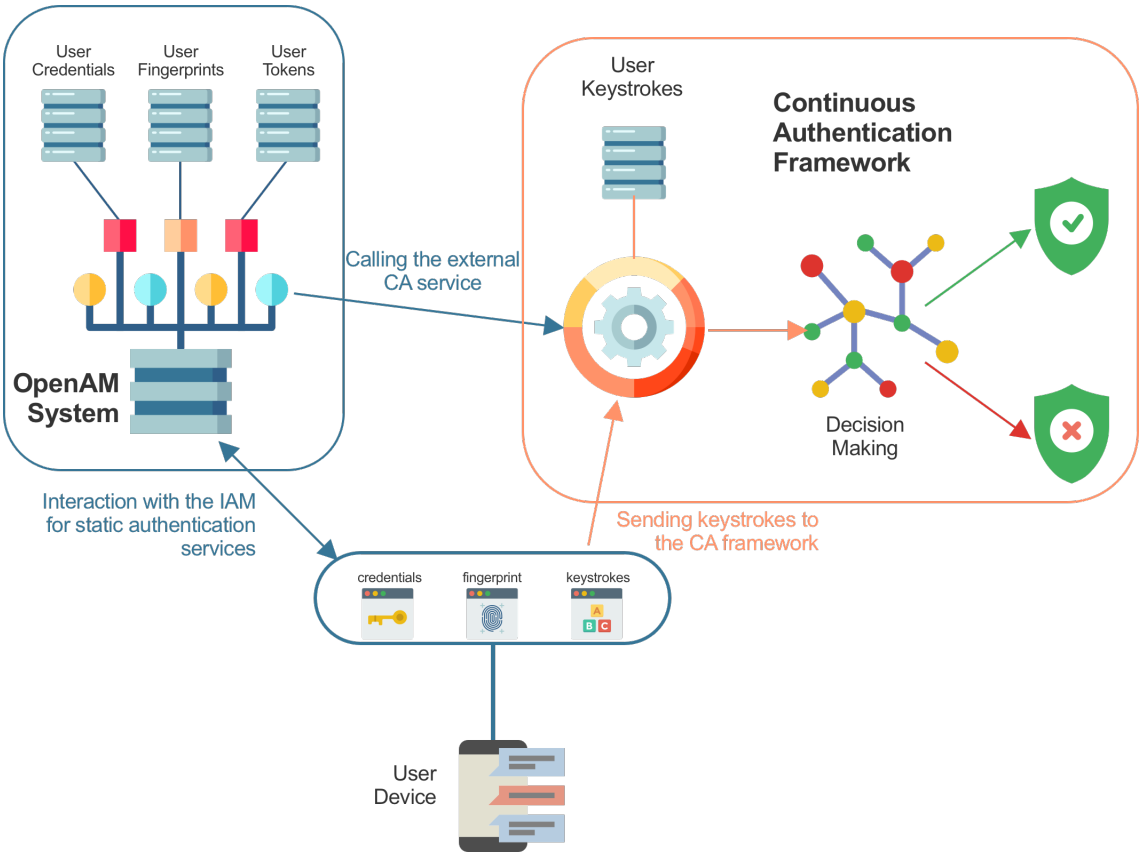


Figure 3.6: Complete system architecture design.

3.5 Extending the architecture with privacy techniques

In order to extend the proposed architecture with privacy-preserving techniques, it is required to identify the privacy threats of the system. To identify these threats in the designed solution, the LINDDUN[5] framework is used.

The threat modeling technique of the framework is composed of three steps [6]. These steps are: (1) defining a DTD (Data Flow Diagram), (2) mapping the privacy threats to the DFD elements and (3) identifying the threat scenarios. The DFD presented reflects the high-level system architecture proposed, that has been illustrated in

[5]https://distrinet.cs.kuleuven.be/software/linddun/index.php

[6]https://distrinet.cs.kuleuven.be/software/linddun/linddun.php

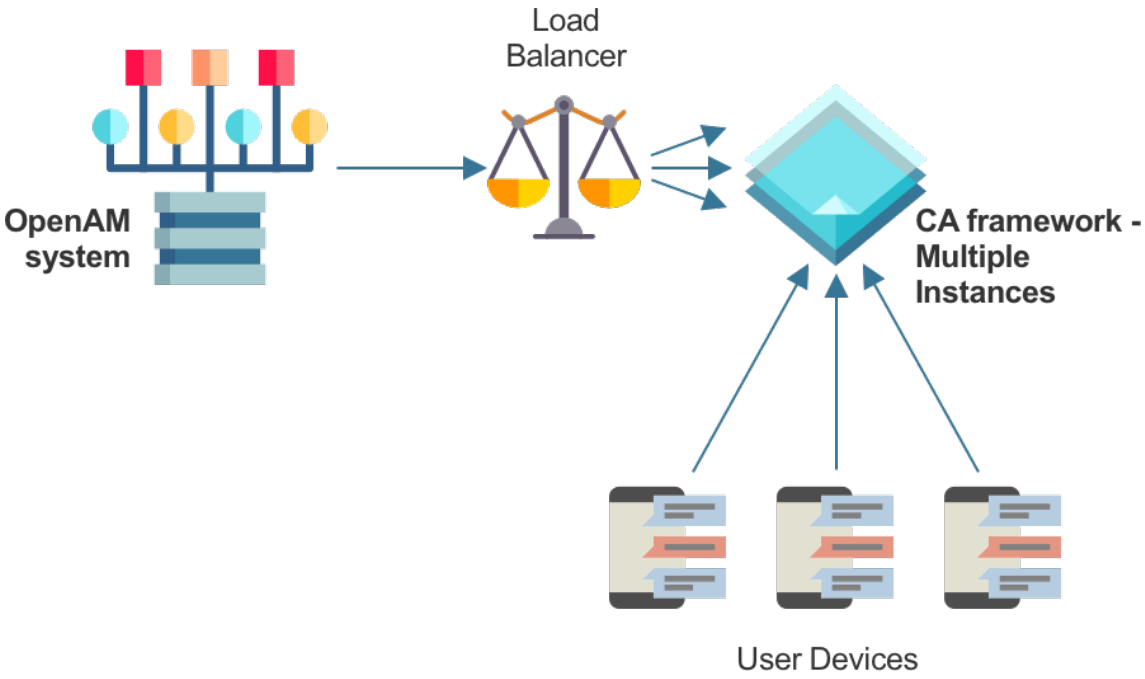


Figure 3.7: Scalable solution for the keystroke authentication framework.

Figure 3.6.

DFD modeling of the system

Figure 3.8 depicts the DFD of the system architecture.

Mapping threats to DFD

The second step requires to determine the threats that correspond to the DFD proposed. The different privacy threats are defined below [46].

- **Linkability (L):** occurs when it is possible to distinguish whether two items of interest (IOI) are related

- **Identifiability (I):** occurs when it is possible to identify a particular subject (e.g. the user)

- **Non-repudiation (Nr):** occurs when it is possible to gather evidence so that a party cannot deny having performed an action

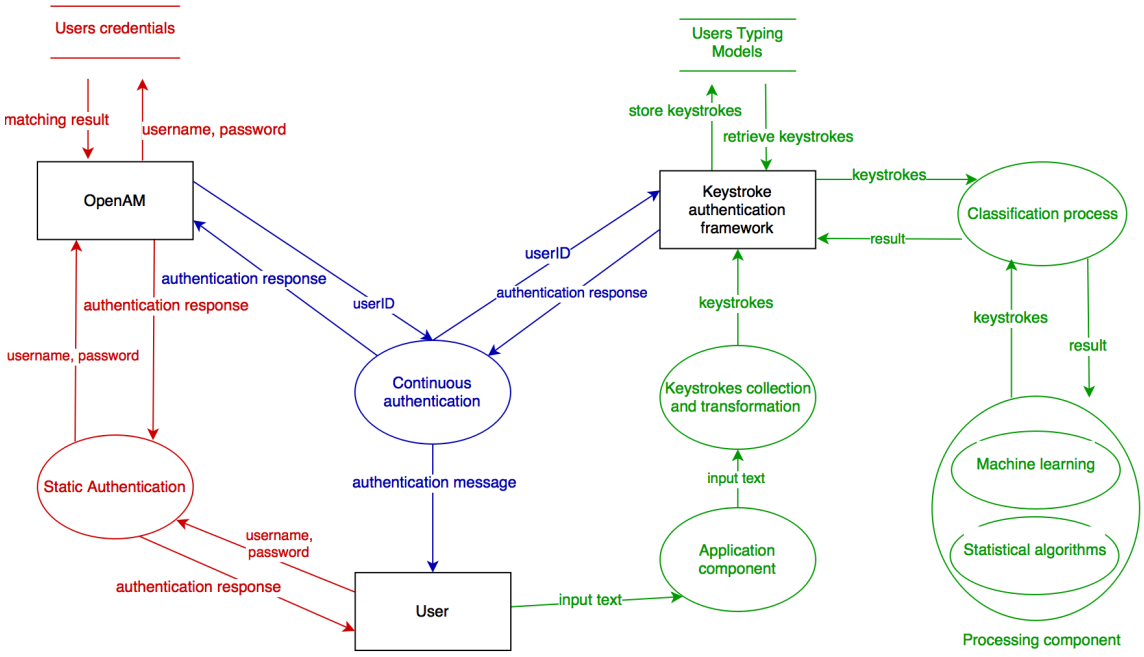


Figure 3.8: DFD representing the data flows in the designed solution.

- **Detectability (D):** occurs when one can sufficiently distinguish whether an IOI exists in a system

- **Disclosure of information (D):** is the exposure of information to individuals who are not supposed to have access to it

- **Unawareness (U):** occurs when the user is unaware of the information he is supplying to the system and the consequences of his/her act of sharing

- **Non-compliance (N):** occurs when the system is not compliant with the (data protection) legislation, its advertised policies and the existing user consents

Table 3.1 indicates the threat categories that are applicable to the DFD elements (**E**ntity, **D**ata **F**low, **D**ata **S**tore, **P**rocess). Each "X" indicates if there could be a potential threat risk to a certain DFD element, for each of these four elements.
The assumptions made are that the data the processes performed by the IAM entity are trusted. Whereas, processes within the keystroke dynamics authentication

Threat Categories	E	DF	DS	P
Linkability	X	X	X	X
Identifiability	X	X	X	X
Non-repudiation		X	X	X
Detectability		X	X	X
information Disclosure		X	X	X
content Unawareness	X			
policy/consent Non-compliance		X	X	X

Table 3.1: Mapping of the LINDDUN threat categories to the DFD element types.

system are not trusted. It is assumed that even the service provider could try to to derive the underlying sensitive information from the users data collected. Table 3.2 illustrates the mapping of the privacy threads to the DFD elements of Figure 3.8.

Identify threat scenarios

This phase tries to determine which are the potential threat scenarios in the designed system. As stated before, the processes carried out by the IAM system are trusted. For this reason, the static authentication process is not considered in the analysis. The IAM system is supposed to be trusted and securely protected. The user data store, that contains the profiles of the user and in particular their credentials, is managed by the IAM, and supposedly secured as well. For the other elements of the DFD this assumption does not hold.

The main privacy threat scenario is an attacker that gains access to the system and tries to reconstruct the original text typed by the user. The same attack could be executed by the continuous authentication service provider. If we consider the situation in which the service provider is the same as the one carrying out the continuous authentication method, then the threat of reconstructing the text would not be a problem. This is because the service provider already has access to the text independently of this authentication method. However, if we consider the chat application scenario, the conversations could be probably encrypted, therefore the usage of this authentication system would make the encryption futile.

Privacy-preserving techniques

To prevent malicious users to misuse the system, privacy techniques must be designed and implemented. A first solution to mitigate these privacy problems could be to perform most of the authentication processes locally on the device. In this

	Threat target	L	I	N	D	D	U	N
Data store	profile data	X	X	X	X	X		X
	typing model	X	X	X	X	X		X
Data flow	OpenAM -> profile database			X	X			X
	OpenAM -> static authentication			X	X			X
	OpenAM -> continuous authentication			X	X			X
	profile database -> OpenAM			X	X			X
	static authentication -> OpenAM			X	X			X
	static authentication -> user			X	X			X
	continuous authentication -> OpenAM			X	X			X
	continuous authentication -> user			X	X			X
	continuous authentication -> KA framework			X	X			X
	user -> static authentication			X	X			X
	user -> application component			X	X			X
	KA framework -> typing models	X	X	X	X	X		X
	KA framework -> continuous authentication			X	X			X
	KA framework -> classification process	X	X	X	X	X		X
	application component -> keystrokes preprocessing			X	X			X
	keystrokes preprocessing -> KA framework			X	X			X
	typing models -> KA framework	X	X	X	X	X		X
	classification process -> KA framework	X	X	X	X	X		X
	classification process -> processing component	X	X	X	X	X		X
	processing component -> classification process	X	X	X	X	X		X
Process	static authentication			X	X			X
	continuous authentication			X	X			X
	application component			X	X			X
	keystroke collection and transformation			X	X			X
	classification process			X	X			X
	processing component			X	X			X
Entity	OpenAM	X	X				X	
	User	X	X				X	
	Keystroke authentication framework	X	X				X	

Table 3.2: The mapping of the LINDDUN threat categories to the elements of the system DFD.

case the service provider would not have access to the user keystrokes, but it would process only the authentication messages exchanged by the client component. For instance, these messages could contain a score that expresses how much the user typing behavior is similar to his/her typing model. Based on that, the authentication framework would decide to let the user continue using the application or not. This could seem a sound solution, but the problem is that the application could be somehow compromised. In case this happens, the application may send tampered data in order to trick the continuous authentication mechanism. This is the reason that determined the choice of not performing the classification process on the user device. Another solution could be to apply privacy-preserving techniques to

the keystrokes, before sending them to the remote component. One can argue that also in this case the attacker could still tamper the data before it is sent to the authentication framework. However, in this case the attacker will not know how the authentication is actually performed on the remote system, thus it will not be able to tamper the data in order to trick the system. Some compromises have to be made, between privacy and security, in order to protect users from both types of threats. The implementation of the privacy-preserving strategies investigated is discussed in the next chapter.

3.6 Discussion

The chapter started with presenting a motivating example with the purpose of highlighting the requirements for the proposed authentication system. After presenting these requirements, the design of the system has been illustrated. In particular, the architecture of OpenAM, an open source identity access management system, has been explained along with the possibility of extending the platform. With the purpose of extending this system, a CA component has been designed that can be plugged in the authentication chain of OpenAM. This building block could also add privacy capabilities to the system, along with support for independent scalability. The next chapter describes how the system designed has been implemented.

Chapter 4

Implementation

This chapter presents the implementation details of the continuous authentication system proposed in Chapter 3. Section 4.1 gives an overview about the different components of this system. Then, following sections describe how the different building blocks have been implemented. Precisely, in Section 4.2 we describe how the server component has been implemented, whereas in Section 4.3 the client application implementation is illustrated. In Section 4.4 the algorithms used for the keystroke dynamics analysis are explained in detail while the privacy-preserving extensions are presented in Section 4.5. The chapter concludes with Section 4.6, where solutions to scale the our system are presented.

4.1 System overview

To implement the authentication system, the **client-server** architecture pattern has been adopted. The client component is responsible for collecting the keystroke dynamics data generated when a user types on the keyboard. In the meanwhile, it transmits the data collected to the server. The second component, the server, has to retrieve the data from the client and it has to carry out the authentication operation. The server is responsible for informing periodically the client whether the authentication has been accomplished successfully or not. In the case in which anomalies are detected, it has to prevent the client to continue using the application.

Given this architecture, two main applications have been developed: a *client-side* application and a *server-side* one. The former runs on a remote server, while the latter runs on the user mobile device. The server application has been implemented in **Node.js**, an event driven *JavaScript* runtime that is used to build scalable network applications. The client application has been developed for the **Android** Operating System.

4.2 Server component

The server application is the component responsible for managing the authentication processes of the different users of the system. It lets a new user register to the system, log in with a password based mechanism and it has to continuously assess the legitimacy of the user sessions. The server exposes **JSON RESTful APIs**, that the client can contact to perform these operations. Exposing the APIs in an established standard makes the integration with the client straightforward. Developers that want to integrate the authentication services in their applications will have to implement only a module capable of communicating with the server application through these APIs. Moreover, authentication systems that can call external services to perform additional authentication tasks, could integrate our service easily. This is the case of the OpenAM system presented in 3.2.1. OpenAM could manage the static authentication of the users, delegating only the continuous assessment of the authenticity of their sessions to the server application.
Subsection 4.2.1 describes the endpoints of this server application.

4.2.1 Server endpoints and methods

The communication between the client and the server is made possible by the exchange of HTTP (Hypertext Transfer Protocol) messages between the two components. Since the REST APIs are accessible through HTTP, it is necessary to identify the URIs (Uniform Resource Identifier), the respective methods, and where the resources they expose can be accessed. The APIs have been implemented using **express**, a Node.js framework. **Express** lets the programmer to define different **routes**. These routes determine how the application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, etc.). To give the reader an idea of how an **express** route is defined, we propose an example taken from the official **express** website[1].

The route definition has the following structure:

```
app.METHOD(PATH, HANDLER)
```

Where: **app** is an instance of **express**, **METHOD** is an HTTP request method, **PATH** is a path on the server and **HANDLER** is the function executed when the route is matched. Let us suppose that we want the server to respond with a *Hello world!* messages on a GET request on the homepage, the route will be defined as:

```
app.get('/', function (req, res) {
  res.send('Hello World!')
```

[1] https://expressjs.com/en/starter/basic-routing.html

})

Following this approach, the different endpoints of the application, the HTTP methods and their handlers have been defined as follows:

/api/register, method:POST This endpoint is contacted by the client application in order to let a new user register to the system. The registration data is sent along with the POST request, and it is expressed as a **JSON**. In detail:

- **fullname**: a string that represents the user complete name.
- **username**: a string used to identify uniquely the user within the system. Hence, there cannot be two users with the same username.
- **password**: a secret string that will be provided by the user, along with the username, in the next login operations in order to accomplish the authentication.

Passport[2], a Node.js authentication middleware, has been used to implement the authentication operation. A simple *local authentication strategy*, as it is called in the **passport** notation, has been implemented for both the registration and the login phases. It works as follows. Once the server receives a registration request, it executes the *local-signup* strategy: the hash of the password is generated using the **bcrypt** password hashing function, then the hash produced is provided, along with username, firstname and lastname to the function responsible for creating a new user into the "USER" database table.
If the operations are successfully performed, the server will send back to the client a message with the *200* (OK) status code. Otherwise, the *500* (Server Error) status code will be sent in the response.
An example of a successful registration operation is shown in Figure 4.1.

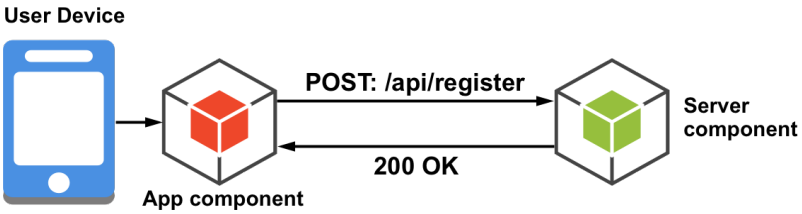


Figure 4.1: Example of a successful registration operation.

/api/auth, method:POST This endpoint is contacted in order to perform the login operation. The POST data, sent along with the request, in the **JSON** format, is:

[2]http://passportjs.org

- **username**: a string that represents the username chosen by the user during the registration phase.
- **password**: a secret string associated with the username provided.

Once the endpoint is contacted, and the POST data received by the server, the login operation is performed, by invoking the *local-login* authentication strategy. Like for the *local-signup* strategy, the hash of the password is produced using the **bcrypt** hashing function. Then, the system searches for a user in the "USER" table with the username equal to the one provided in the POST request. If there is not such a user, a *404* (Not Found) will be sent in the response. Otherwise, the server will check whether there is a match between the hash of the password sent and the hash of the password saved during the registration phase. If this is not the case, this means that the password sent was wrong, therefore the *500* (Server Error) status code will be sent back. Otherwise, if the two hashes match, the *200* (OK) status code will be provided in the response. An example of a successful login operation is shown in Figure 4.2.

After both the registration and login operations a **JSON Web Token**[3] (JWT) is sent in the server response, along with the 200 status code. The JWT is generated, and digitally signed, by the server. Each subsequent request, made by the user, must include the JWT, allowing the user to access the other routes, services and resources[4]. A middleware function has been implemented in order to verify the JWT. In the case that the JWT is missing, or not valid, the access to the other routes will be denied, and a *403* (Forbidden) status code will be sent in the response. The JWT contains the user identifier of each user, once it is verified and the content decoded, it is used by the server to identify from which user the requests are coming.

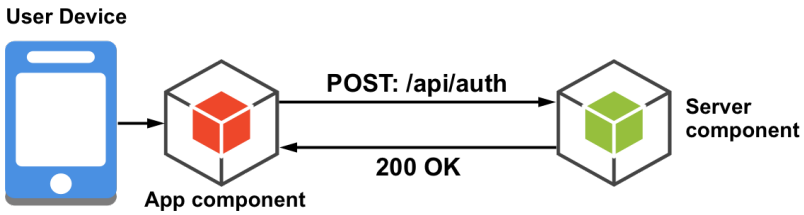


Figure 4.2: Example of a successful login operation.

/api/monitor, method:POST This endpoint is contacted in order to perform the continuous authentication operations. The POST data, in the **JSON** format, is:

[3]https://jwt.io

[4]https://jwt.io/introduction

- **from:** the first key pressed.
- **to:** the second key pressed.
- **time:** the time elapsed between the two keys.

The server uses the keystroke information in order to analyze the user typing behavior. If the authentication is not performed successfully, a *500* (Server Error) message will be sent to the client. Otherwise, the *200* (OK) status code will be provided in the response. Section 4.4 describes in detail how the keystroke dynamics analysis is performed.

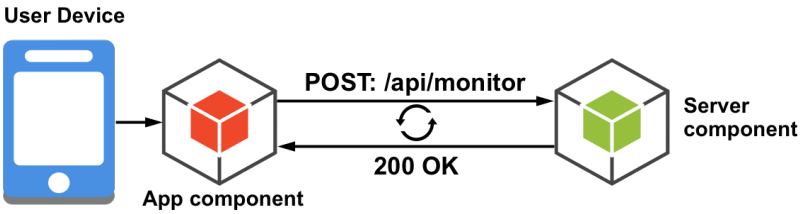


Figure 4.3: Example of a successful continuous authentication operation.

4.3 Client component

The client component is implemented as an Android application. It presents a simple interface. It is composed of three **Activities**. An **Activity** in Android is a single, focused thing that a user can do. The activity is responsible for creating a **View** that lets the user to interact with the application to perform the operations specified in the **Activity**. The three activities that compose the application are:

- **RegisterActivity:** it lets a user register with a new account.
- **LoginActivity:** it lets a user log in into the application, once he/she is already registered.
- **MainActivity:** it lets a user to input some text to verify his/her typing behavior, in order to perform the continuous authentication operation.

The next three subsections describe the operations performed by these three activities.

4.3.1 RegisterActivity

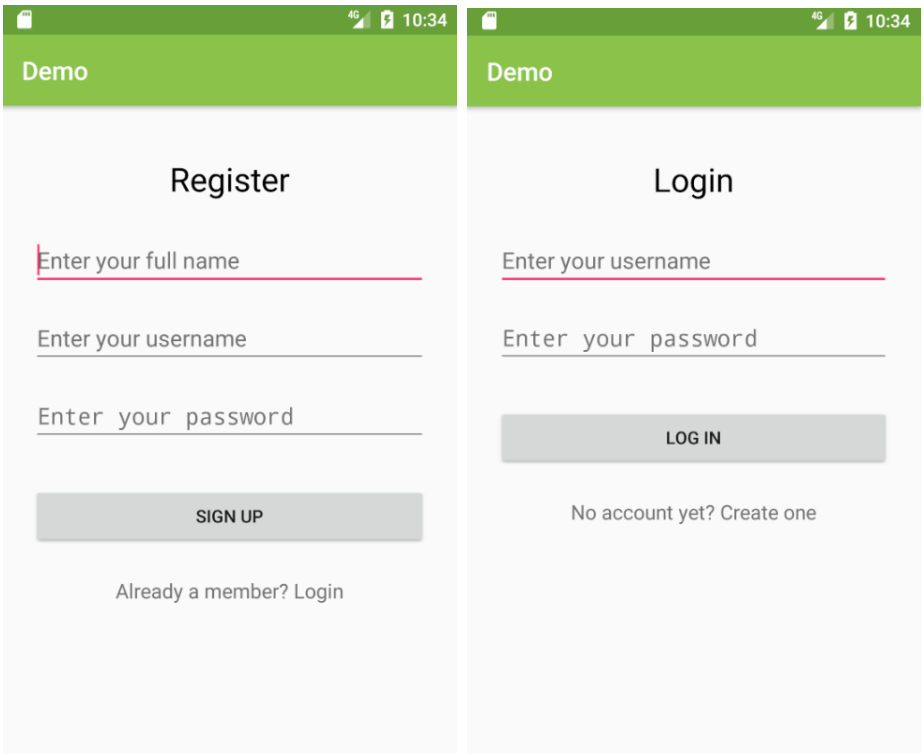
The **RegisterActivity** presents a view, that is shown in Figure 4.4a, composed of a registration form. The form is made of three input boxes where the user can insert his/her credentials in order to accomplish the registration, which are: his/her full name (first and last name together), the username and the password. By clicking on the submit button, an HTTP POST request is sent to the server in order to trigger the registration operation on the server. A **JSON** is created with the string inputted in the three boxes, which is sent as the POST data of the HTTP request. After the request, the client waits for the server response. In the case in which the registration is accomplished successfully, the activity saves on the device the JWT that is received in the response. Then, the **MainActivity** is executed.
Whereas, if the authentication fails, an alert will appear to show the user an error message.

4.3.2 LoginActivity

In the case the user is already registered, the **LoginActivity** will be executed in order to let him/her log into the system. The activity presents a view similar to the one of the **RegisterActivity**, Figure 4.4b. The form in this case is composed by only two input boxes that let the user insert his/her credentials (username and password). Once the user submits the form, the server endpoint responsible for the login operation is contacted through a POST HTTP request. Even in this case, a **JSON** that contains the strings username and password is provided along with the request. Like for the **RegisterActivity**, if the login operation is completed successfully, the server will send back the JWT that will be stored on the device. Whereas, if an errors occurs, the login operation will be interrupted and an error message will be shown to the user. Once the user is successfully logged in, the **MainActivity** is executed.

4.3.3 MainActivity

The **MainActivity** shows a view with a simplistic message conversation interface, Figure 4.5. It lets the user type in an input box and to send the written messages. In fact, no message is actually sent. The view is for demonstration purpose only. The application was not intended to be a real chat application. The relevant operations performed by the activity are carried out when the user starts typing in the message box. An *event listener* is registered to this box, that triggers the execution of a *Java method* every time a user presses a key. In particular, the *method* has to sent to the server the **JSON** composed of: first key pressed, second key pressed and time elapsed between the two keys. The response from the server can be either a successful response or an error response. In the first case the user can continue using



(a) RegisterActivity View. (b) LoginActivity View.

the application. Otherwise, an alert will pop up showing an error message and, after that, the **LoginActivity** will be executed in order to force the user to log in again to the application.

4.3.4 Discussion about the client

The client application was developed as a proof of concept, to demonstrate how in practice it is possible to integrate the continuous authentication service, that is provided by the server application, in a real world application. The application was not meant to be a real chat application, for this reason most of the functionalities required by standard chat applications are missing. Each of the three activities implemented for the application contacts a particular endpoint of the server application, to demonstrate how these endpoints can be used.
Another possible solution, instead of implementing a native application, could have been implementing a custom virtual keyboard. In this case, the keyboard would have been responsible for contacting the server endpoints. The static authentication with username and password would have been implemented in the keyboard settings. Whereas, the keystrokes would have been sent to the server component for every interaction of the user with the keyboard. In this way, the authentication would

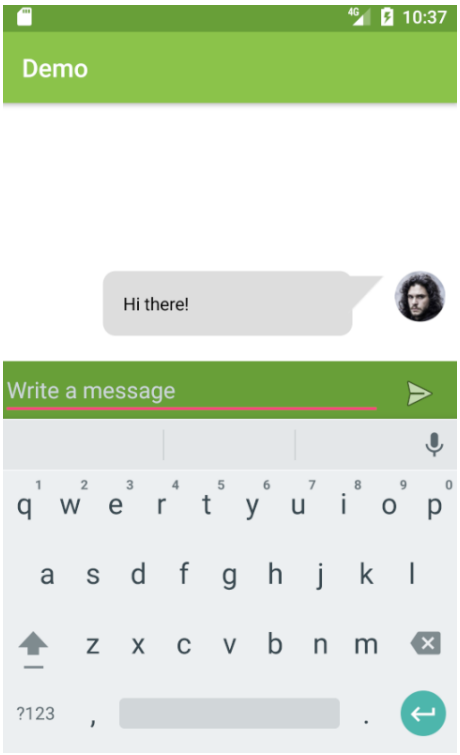


Figure 4.5: MainActivity View.

have been not only in a particular application, but it would have been extended to the whole Operating System.
Otherwise, another possible solution to do this keystroke dynamics OS-wide authentication, instead of implementing a virtual keyboard, could have been to implement an Android *service*. A *service* is an application component, without user interface, that performs long-running operations. In this case, the *service* would have monitored the interaction of the user with the virtual keyboard, and would have sent the data to the server in order to continuously authenticate the user.
The reasons why we opted for the Android application are that (1) it reflects the use case we presented in Chapter 3, (2) it is the least invasive solution for mobile phone, since the collection of keystrokes is restricted to the application only, and (3) it was easier to test with the emulator.

4.4 Implementation details of keystroke dynamics analysis

In this section we describe the algorithms used for the authentication mechanism. A JavaScript object, named **Classifier**, has been implemented in order to perform

the authentication tasks. The object is instantiated globally in the application. It represents the state of the application. At the first run, the application creates a new instance of this object, providing in the constructor some options that are specified in a configuration file as a JSON. The options define the algorithms settings. For instance, they specify how many keystrokes are required to train the user model, the threshold used to compare if two typing samples are similar, and so on. A *cron job* is executed in order to save, every minute, the state of the **Classifier** (its attributes) into a file. This is done because, every time the server restarts, the application can restore the previous state of the **Classifier**, retrieving it from the backup file.

To summarize, when the server starts it first checks whether is present a backup file. If it exists, then the **Classifier** will be created with the options and the previous state in the constructor. Otherwise, a new fresh **Classifier** will be created, with the options specified. The choice of instantiating this global object has been made for performance reasons. A possible alternative to this approach could have been to allocate this object for each new request made by the client. In this case the users models would have been retrained each time, retrieving the data required for the training from the database. This alternative approach would have created a performance bottleneck, especially in the case of a high number of users. However, if it would be required to store the keystroke data of the users, another component, or another *cron job*, could be implemented in order to parse the backup file and store its content in the database.

The next subsections describe how the models of the users are trained and how the authentication operations are performed.

4.4.1 Training of the user model

After a user registers to the system, the training operation is performed. The system collects the typing information from the user and stores it. The data received is the **JSON** that contains the *digraph* sent by the client through the **/api/monitor** endpoint. Each time the server receives a new *digraph*, it checks whether there are enough *digraphs* stored in order to perform the training of the model. If there are not, the new *digraph* received will be stored in an array, along with the previous ones generated by that user. Once there is enough data, the training operation can be executed.

In order to accomplish the training, a first **filtering** operation is performed. All the *digraphs* that have the time associated above a given certain threshold are discarded, along with all the ones that contain non printable characters. The threshold for the time duration can be specified in the configuration file. After the filter is applied,

the remaining *digraphs* are organized in M arrays of N *digraphs* samples. The two numbers, M and N, can be specified in the configuration file. At this point, the mean of the distances between each possible pair of these samples is calculated. For instance, suppose we have three samples for a user A that are A1, A2, A3, and that we have:

$$\text{d(A1,A2)} = 0.312378;\ \text{d(A1,A3)} = 0.304381;\ \text{d(A2,A3)} = 0.326024;$$
$$\text{m(A)} = (0.312378 + 0.304381 + 0.326024)/3 = 0.314261$$

"The *m(A)* can be seen as a sort of number representative of the way user A types on a keyboard. It gives us an idea of the distance we may expect between two typing samples provided by user A" [20]. This number will be used for the authentication process.

4.4.2 Authenticating the user

The authentication process is performed for every new *digraphs* received, once the training phase is terminated. Like for the training phase, a specified number of *digraphs* is required in order to perform the authentication. The new *digraphs* coming from a user are stored in an array, until the required number is reached. At this point, the user identity can be verified. Two modes of operation can be distinguished.

Authentication mode (classification) The *mean distance* (*md* for short) of a typing sample X from the user A is defined as:

$$\text{md(A,X)} = [\text{d(A1, X)}+\text{d(A2, X)}+...+\text{d(An, X)}]/\text{n}$$

where $\{A_1, A_2, ..., A_n\}$ are the typing samples collected during the training phase and d is a given distance measure. In order to authenticate user A correctly, two requirements have to be satisfied:

1. md(A,X) must be the smallest w.r.t any other md(B,X), where B is another legal user in the system;
2. md(A,X) is smaller than m(A), **or** md(A,X) is closer to m(A) than to any other md(B,X) computed by the system.

The first condition implies that the sample X, coming from the user A, must be the closer sample to A's samples. Moreover, the second requirement states that it has to be sufficiently close to A in order to accept it. The technique has been proposed in [20].

Anomaly detection mode (clustering) By using the anomaly detection mode, a user sample is compared only against his/her model. This technique is easier to implement, faster in execution speed, but less precise in terms of accuracy.

To assess that a sample X comes from the user A, the following condition has to be satisfied:

1. md(A, X) must be smaller than m(A).

In this case, no other users' samples are required. The sample must only be sufficiently close to A. This mode of operation should only be used in case there is no other user data to compare with, or when optimization, in terms of time and computation, is required, since only one comparison is actually performed.

4.4.3 Retraining of the user model

The user typing behavior may probably change over time. Once he/she gets familiar with the smartphone keyboard, his/her typing speed may eventually increase. This would produce wrong results during the authentication process. To avoid unpleasant situations, e.g. a legitimate user rejected by the system, the *retraining* of the model could be performed. The retraining option can be specified in the configuration file. If its value is set to true, once the authentication process is performed successfully (the sample is classified as belonging to the legitimate user) the new sample can be saved, replacing the oldest one. Then, the mean m can be recomputed. The system has to be sure that the data used to retrain the model are legitimate data from the user. This is because an attacker could use this property to subvert the user model. For this reason, the retraining is performed only after a certain number of successive typing samples is recognized as legitimate.

4.4.4 Calculating the similarity between typing samples

As we described in Section 3.3, the distance measure that is used to compute the distance, or (dis)similarity, between two typing samples, is given by the combination of the "R" measure and "A" measure, proposed by Gunetti et al. in [20].

R measure Different R measures can be produced, based on which representation in terms of *n-graphs* is chosen to calculate it. The one we chose is the $R_{2,3}$ measure, since it is the one that produced best results. To compute it, R_2 and R_3 have to be measured first. To calculate R_2 between two typing samples, a series of operations are performed.

1. Two sequences of *digraphs* are produced starting from the two samples. Since the two samples are made of *digraphs*, the operation is almost performed. The only operation required is the removal of same *digraphs* within the same user sample. Only one of *digraphs* that express the transition of the same two keys must remain. The mean of the time spent to write the same *digraph* is calculated and assigned to the remaining *digraph*. For instance, suppose we

have three *digraphs* that represent the transition from key a to key b, called DA, DB and DC, we have:

$$\text{DA} = (\text{a, b, 100}); \text{DB} = (\text{a, b, 200}); \text{DC} = (\text{a, b, 300});$$
$$\text{DR} = (\text{a, b, } (100 + 200 + 300)/3) = (\text{a, b, 200}).$$

2. Once the two sequences of *digraphs* are created, they are filtered in order to produce two sequences that share the same *digraphs*. For instance, if a *digraph* is present only in sample S1, it has to be removed.

3. Then, the two sequences are ordered, producing two new sequences with *digraphs* that have been typed faster in the first positions.

4. The *degree of disorder* is computed as the sum of the distances between the position of each element in the two arrays. The final result is the normalized value of the disorder, i.e. the disorder calculated divided by the maximum disorder.

Once the R_2 measure is calculated, R_3 is computed. The operations are the same. The only difference is that *trigraphs* representation is used. To produce the *trigraphs* two subsequent *digraphs* are combined together. If the second key of the first *digraph* is equal to the first key of the second *digraph*, then the resulting *trigraph* can be produced. The time duration is calculated summing the two time duration of the two *digraphs*.

$$\text{DA} = (\text{y, e, 200}); \text{DB} = (\text{e, s, 100});$$
$$\text{TA} = (\text{y, e, s, 300}).$$

In order to produce the $R_{2,3}$ measure, the following formula is used:

$$R_{2,3}(\mathbf{E1},\mathbf{E2}) = R_2(\mathbf{E1},\mathbf{E2}) + R_3(\mathbf{E1},\mathbf{E2}) * \text{N}/\text{M}$$

Where E1 and E2 are the two typing samples, N and M are the lengths of the two sequences of *digraphs* and *trigraphs*, respectively.

A measure The A measure, can be computed, like the R one, by combining different A_n with different values for n. Previous research [20][32] showed that using only A_2 gives the best accuracy. In order to calculate it, the two sequences of shared *digraphs* have to be computed. Once the two are ready, the A measure is calculated using the technique described in 3.3.5. The threshold t, that is used to decide whether two *digraphs* are similar, can be specified in the configuration file.

Once $R_{2,3}$ and A_2 are calculated, the resulting distance measure d is produced as the sum of the two measures:

$$d = R_{2,3} + A_2$$

4.5 Privacy-preserving extensions

Privacy-preserving extensions have been implemented, with the aim of reducing the possibility for illegitimate users (attackers or *honest but curious* service providers) of reconstructing users' behavior. Since the users' typing habits are exploited in order to model and analyze their behavior, the implemented system in order to preserve their privacy should avoid the possibility of reconstructing the original text typed by them.

The fact that the *n-graph* data type has been adopted to represent the timing features of a typing behavior, is already a privacy precaution. The *n-graph* does not include the timestamp of when a certain key was pressed. For this reason, the order of the keystrokes in a typing sample can be shuffled and the result of the authentication process should not get worse. To implement this mentioned solution, it will be futile to perform this keystrokes permutation on the server-side application. Thus, this operation should be carried out by the client application. Instead of sending each new keystroke collected, a batch of keystrokes is sent, where the order of the keystrokes is permuted.

A similar solution is to sent only a certain portion of the keystrokes and observe if the results are still good. A combination of the two techniques could also be applied in order to make harder the reconstruction of the original text.

The third technique we implemented is to modify each key sent, substituting it with a close key on the keyboard. Figure 4.6 shows with which keys the "s" key could be substituted. A random key among the colored ones of the figure is chosen to substitute the "s" key. For instance, instead of sending the *digraph* composed by the keys "s" and "o", the *digraph* sent will be composed of the keys "a" and "l". The key used for the substitution is chosen randomly every time a new *digraph* has to be sent. This means that same typed *digraphs* will result as different when they will be sent.

This solution is based on the assumption that certain sequences of keys, with keys that are close in the keyboard, are typed probably with the same speed. This could not always be the case. Certain combinations of keys are typed more frequently than others, hence they are typed faster. The impact on the authentication accuracy using this technique is presented in the next chapter. Also in this case, the technique could be combined with the first two proposed.

The three techniques presented have been named respectively **permutation**, **suppression** and **substitution**.

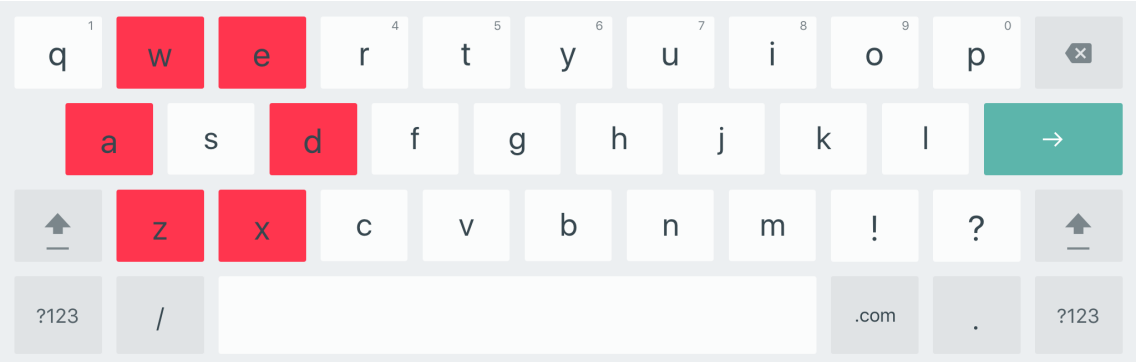


Figure 4.6: Example of substitution for the "s" key.

4.6 Scaling the system

In order to make the system scale, new resources have to be allocated when it is required. Different possibilities have been explored to accomplish that. It has to be noticed that the computation overhead increases proportionally to the number of users of the system. This is because of two reasons: (1) when the number of users increases the system has to manage more data at the same time, (2) as it is described in 4.4.2, when the system works in **authentication mode**, it has to compare the typing samples against the other users models. Whereas, using the **anomaly detection mode** this problem does not occur. A possible solution to prevent the delay of the authentication process could be to instantiate a new classifier each time a certain number of users registers to the system. Doing so, the system should have to keep track of which classifier is associated to a particular user in order to redirect the typing samples received to the correct classifier. These classifiers could be deployed on different machines, in order to scale out the system. This configuration is depicted in Figure 4.7.
Another solution to scale the system horizontally could be to deploy the same classifiers across different machines, and distribute the computation over them, in order to balance the machines load. This means that every machine should have the same copy of the different classifiers and these replicas should be synchronized. In order to do that, a centralized approach could be adopted. In this case, a central machine would store the classifiers and the other machines would refer to the central one to retrieve a certain classifier and to update it after an authentication operation is performed.

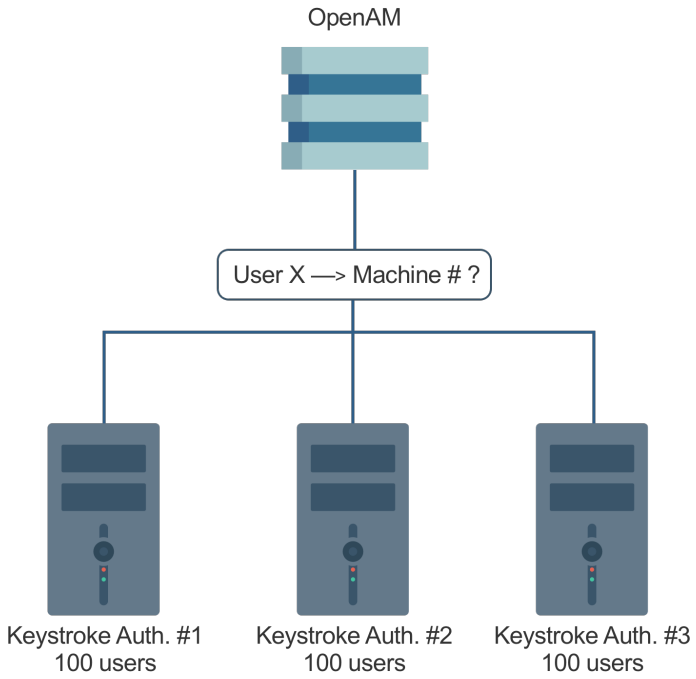


Figure 4.7: Scaling the system among different machines.

Chapter 5

Evaluation

In Chapter 3 we identified six functional and non-functional requirements for our continuous authentication system. In this chapter we focus on the evaluation of our solution. We mainly focus on the evaluation of the authentication accuracy of the system implemented, as well as the influence on this accuracy when our privacy-preserving techniques are applied. Thus, we first propose a quantitative evaluation of the functional requirements, that are (1) the system should let users authenticate to the system with username and password, (2) it has to monitor constantly the user behavior in order to (3) continuously authenticate them based on their typing information. In particular, the only functional requirement that has to be evaluated is (3). We present the experiments performed in order to evaluate the authentication accuracy by presenting: the experimental settings in Section 5.1, the evaluation criteria in Section 5.2, the results of the experiments we have performed without applying the privacy-preserving techniques in Section 5.3 and the ones obtained using these techniques, in Section 5.4. In Section 5.5 we describe the validity threats of our results. Finally, the chapter concludes with Section 5.6 with a qualitative evaluation of the non-functional requirements we identified for our system.

5.1 Experimental settings

In order to evaluate our authentication system, different datasets have been used. In particular, some datasets were publicly available, while new data has been collected through a self-developed web site. The availability of these datasets is vital for the evaluation of keystroke dynamics research. Even if some dataset were found online, the problem encountered was that most of them were made of samples taken from a user who typed fixed-length strings. This is because most of the work done until now focused on exploiting the user typing behavior in order to provide an additional step within the *static* multi-factor authentication process. The aim of

these previous works was to research a more secure password based authentication method. With this new method the correctness of the password is checked, along with the way password was typed. Datasets used for the analysis of free text were available, however they were constituted of samples collected through computer keyboards, not with smartphones. This may be due to the fact that smartphone devices have not been with us for a very long time, and the acquisition of such data is a time and resource consuming process. The next subsections explain the settings of the datasets that have been used for the evaluation. A **python** script has been implemented in order to convert these datasets in the required form.

5.1.1 Dataset 1: Fixed-length text on smartphones

The first dataset[1] is composed of data collected from 54 users who typed easy and strong passwords on mobile devices. We refer to [4] for the associated research and its results. The keystrokes were collected using an application implemented for Android devices. They collected different types of features was collected: time-based, touch-based and accelerometer-based. We used for our evaluation only the time-based, since our solution exploits only this type of feature in order to model and analyze user typing behavior. We considered this dataset for our evaluation in order to understand whether the algorithms we implemented, that were supposed to work for the analysis of free composed text, were also suitable for the analysis of keystrokes generated by users who typed fixed length strings on smartphones.

5.1.2 Dataset 2: Free and transcribed text on computers

The second dataset[2] used for the evaluation is made of samples collected from users who typed on computer keyboards free and transcribed text. This data was collected for the research conducted by Killourhy et al. in [29]. The aim of their research was to establish, in the context of keystroke dynamics analysis, whether samples produced by subjects that transcribed passages of text and samples generated by freely writing users would have produced equivalent results in terms of authentication accuracy. The transcription task is often easier for the subjects of the experiments, rather than free composition one. The results of their research showed that the difficulty of collecting freely composed text is unnecessary, and the transcription task can still be used.
This second dataset is made of keystrokes collected from 20 users, where each user completed two free and two transcription tasks. We used this dataset in order to verify whether the techniques we used would achieved high authentication accuracy

[1] `https://www.ms.sapientia.ro/~manyi/mobikey.html`

[2] `http://www.cs.cmu.edu/~keystroke/laser-2012`

in the environment they were conceived for, e.g. analysis of free text on computer keyboard [20].

5.1.3 Dataset 3: Transcribed text on smartphones

Since there was a lack of data produced by freely texting users on mobile devices, we developed a website used to collect this kind of data. The website presents a simple interface, composed by a paragraph of text and an input box to copy that text inside. When this form is completed, and the "submit" button is pressed, the generated keystrokes are sent to the collecting server. A client-side JavaScript script gathers the timing information of the typed text. The script, as long as the user types in the box, generates a sequence of *digraphs*, where each digraph is made of two consecutive keys pressed and the time elapsed between the keys. The subjects that were involved in the data acquisition process were 10, and they all used different devices. The number may not seem sufficiently high, but the study carried out by Messerman et al. [32] showed that an increasing number of users in the system does not influence significantly the results. For this reason the conducted their experiments by using a constant number of users, **nr**, choosing randomly **nr** users among the ones of their dataset. This was done especially for scalability reasons, in order to have constant performance in terms of time even when the number of users would have increased.

5.2 Evaluation criteria

In order to test the system, we needed illegitimate user attempts, as well as legitimate ones. Since in the datasets we used there were only legitimate users samples, we opted for the following strategy. For each user of the dataset:

- To simulate legitimate samples, we simply used his/her keystrokes.
- To simulate illegitimate samples, we used other users keystrokes.

The evaluation metrics for the experiments are described in Subsection 5.2.1, whereas Subsection 5.2.2 illustrates how the validation of the model has been performed.

5.2.1 Evaluation metrics

The metrics that are used for the evaluation of biometric authentication methods are **False rejection rate (FRR)**, **False acceptance rate (FAR)** and **Equal error rate (EER)**. Their definition is given below [42].

False rejection rate (FRR) The FRR is the number of falsely rejected users attempts, against the total number of legitimate attempts. A low FRR implies that a few number of legitimate users is rejected, this means that the system as an high level of usability. FRR is also knows as false alarm rate, false negative rate, false non-match rate.

False acceptance rate (FAR) The FAR is the number of falsely accepted users attempts against the total number of illegitimate attempts. A low FAR indicates an high level of security, since illegitimate users are normally rejected. FAR is also knows as miss alarm rate, false positive rate, false match rate.

Equal error rate (EER) The EER is the number that expresses the overall accuracy of a biometric authentication method. It has to be noticed that the FAR and the FRR are negatively correlated, it is not possible to decrease at the same time both the values of the FAR and the FRR. The EER can be found intersecting the graphs of the FRR and FAR. A low value of FRR and FAR produces a low value of EER. For this reason EER is used to express the overall accuracy since it represents the correlation between these two measures.

5.2.2 Model validation

In order to perform the experiments, some data is required for both training the models and testing the system. Since there is not a predefined distinction between training and testing data within the datasets, the **k-fold cross validation** method has been used to validate the results of the experiments. Using this method the datasets are split into k subsets. At each iteration, one of the k subsets is used to test the system, whereas the other *k-1* subsets are used to train the model. We opted for a *3-fold-cross validation* because an high number of keystrokes was required in order to build enough typing samples for the testing phase.

5.3 Performance evaluation

In this section we present the results of the experiments carried out with the three datasets. Each subsection focuses on the results we obtained by using a specific dataset and an observation about each of these results is proposed. We performed these experiments modifying some parameters of the algorithm configuration. The two important ones are:

- **number of digraphs per sample:** the number of *digraphs* required to build a typing sample.

- **score adjuster:** This value has been introduced in order to regulate the FAR and FRR of the authentication. It defines how near to the user model a typing sample has to be in order to recognize it as belonging to that user. By modifying this parameter, different values of FAR and FRR can be obtained. If an high value is assigned to this variable, it is very likely that the system will have an high FAR and a low FRR. An high value can be used when the system requires high usability. On the contrary, when more security is required, a low value has to be assigned to this variable, in order to produce a low FAR and an high FRR.

The experiments have been performed changing these two parameters, while others have been left unchanged. Precisely:

- the number of samples required for the training of the models has been set to 50 samples.
- the threshold to calculate the similarity (A measure) between typing samples, as we described in 3.3.5, has been set to 1.25, as proposed by Gunetti et al. in [20].
- the threshold used to discard the digraphs based on their time duration are: 620 ms for both the first and third datasets, and 750 ms the second one. We selected these numbers after a first round of experiments where we tested different values. Hence, we selected the values that produced the best authentication accuracies.

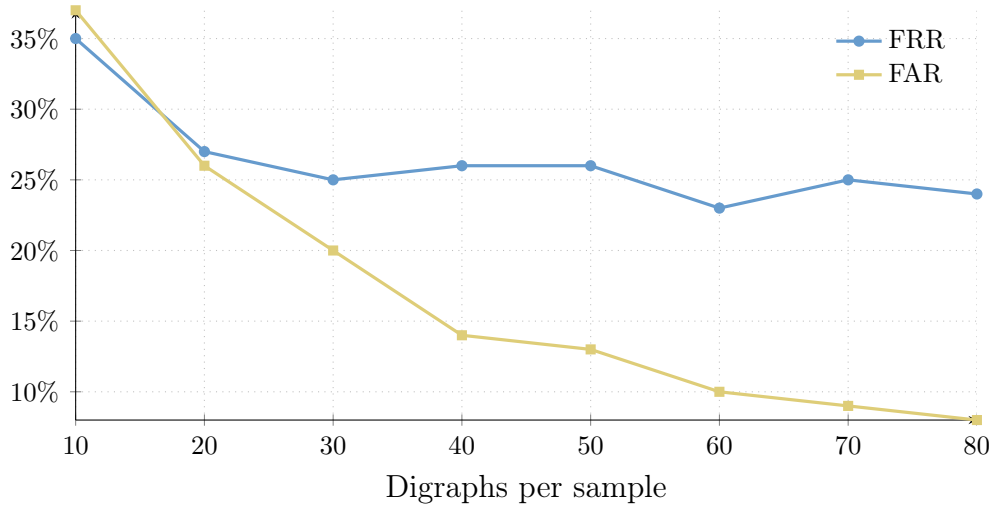
For each dataset the experiments have been carried out testing the two modes of operation implemented: the *authentication mode* and the *anomaly detection mode*.

5.3.1 Dataset 1: results and observations

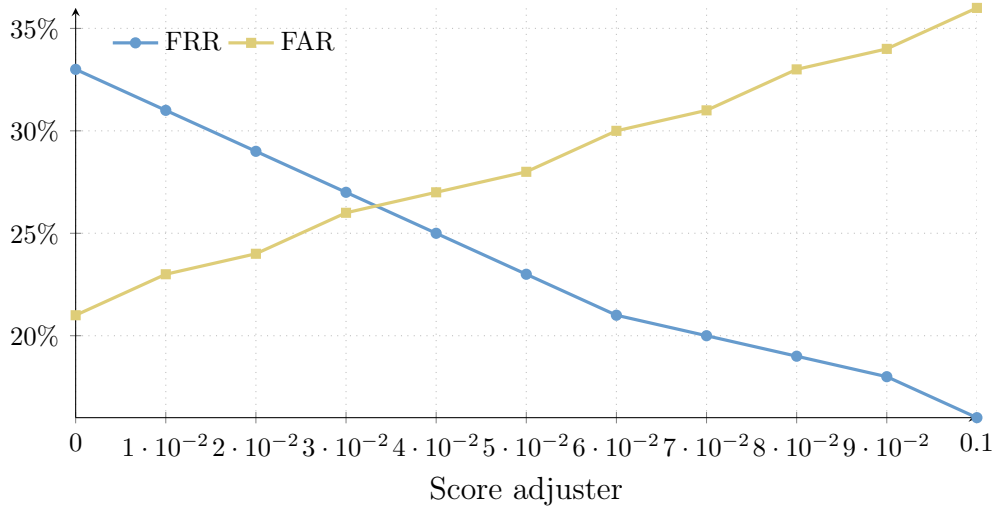
Dataset 1: authentication mode

Figure 5.1 shows the results of the experiments performed using the *authentication mode*. In particular, Figure 5.1a, shows values for FAR and FRR obtained by changing the number of *digraphs* required for each typing sample. In this first experiments the value of score adjuster has been set to 0.03, while the number of *digraphs* has been varied between 10 *digraphs* and 80 *digraphs*. Results show that by increasing this number the performance has increased as well. This is especially the case for the FAR, whereas the FRR seems to stabilize around 25%. The problem in this case is that a password normally is chosen of about 10 characters at maximum, 20 in very special cases. For this reason, the only significant results are the ones obtained when 10 and 20 *digraphs* are used to build typing samples.

By setting the number of *digraphs* per sample at 20 *digraphs*, the next experiments have been carried out changing the value of the score adjuster. Results are shown in Figure 5.1b. The intersection of the two lines represents the EER, that is equivalent to 26%.



(a) FAR and FRR obtained by changing the number of digraphs per sample.



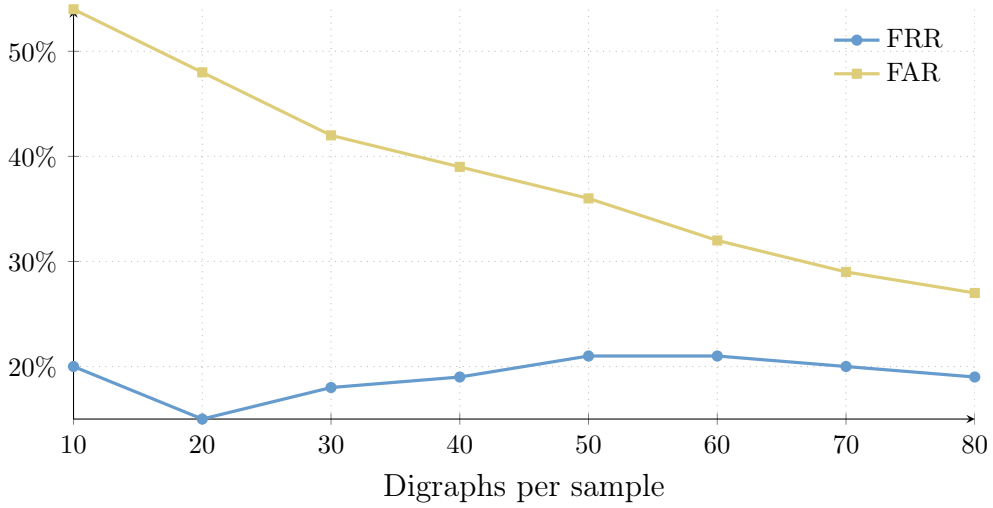
(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.1: Dataset 1: results for authentication mode.

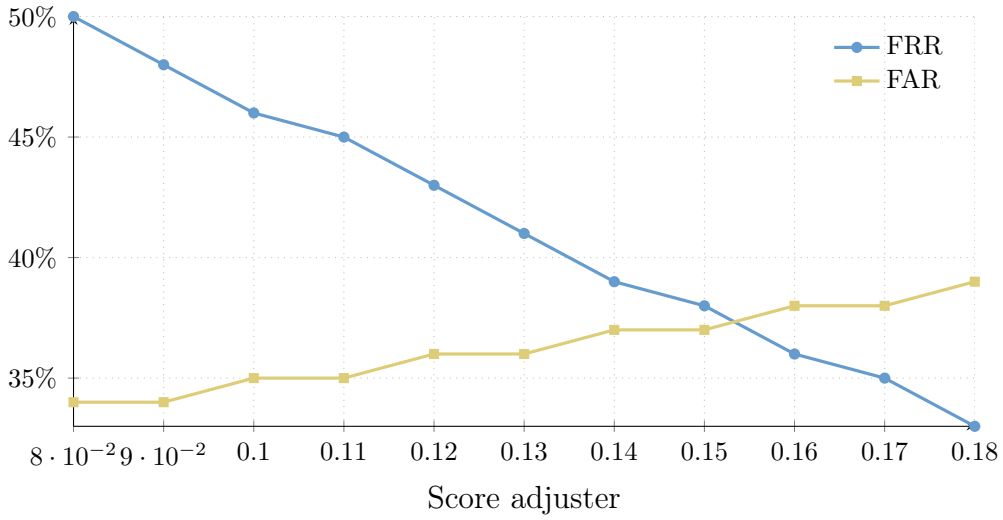
Dataset 1: anomaly detection mode

The same reasoning was made for the experiments related to the second mode of operation. In this case, the EER obtained, by setting the number of *digraphs* to 20,

is around 39%. The results are shown in Figure 5.2.



(a) FAR and FRR obtained by changing the number of digraphs per sample.



(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.2: Dataset 1: results for anomaly detection mode.

Dataset 1: observations

These presented results demonstrate that the authentication techniques we used for our solution do not achieve good results for both the two modes of operation. The *anomaly detection mode* cannot be very suitable to analyze the typing behavior of people who type fixed length strings. This is because the EER, that is around 39%, is too high to be acceptable. The same reasoning applies to the *authentication mode*,

since good performance is registered when there are more characters available with respect to the normal number of characters that composes a password.

5.3.2 Dataset 2: results and observations

For these experiments only the keystrokes collected through the free-composition task of the subjects have been used.

Dataset 2: authentication mode

Figure 5.3 shows the results of the experiments performed using the *authentication mode*. The best performance is registered when 50 *digraphs* are used to compose the typing samples. The EER is located in the intersection of the FAR and FRR in Figure 5.3b. Its value is around 10%.

Dataset 2: anomaly detection mode

The results for the *anomaly detection mode* are similar to the ones obtained using the *authentication mode*. The EER registered is between the 12% and 13%.

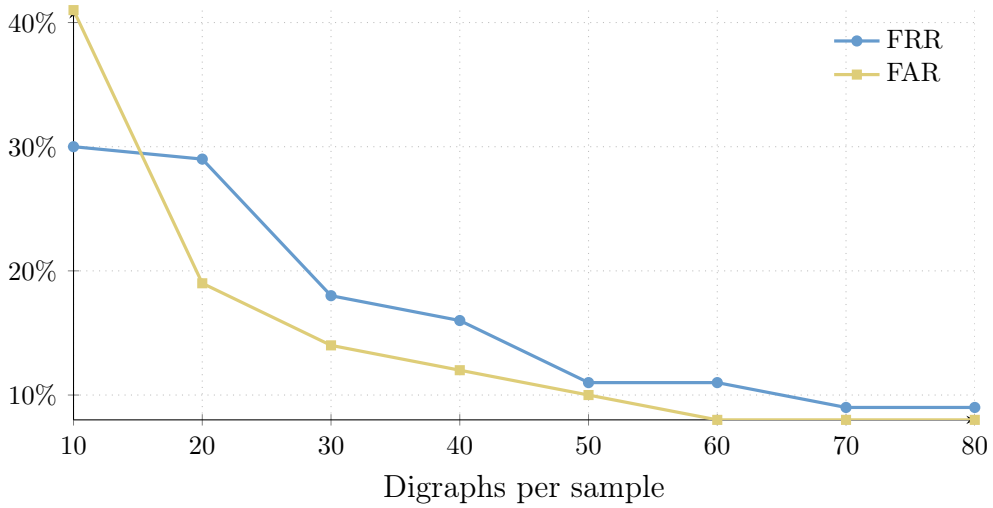
Dataset 2: observations

These results show that the techniques used are definitely suitable for the analysis of the typing behavior of users who type freely on computer keyboards. Both the authentication and anomaly detection mode work really well, since the EER are respectively 10% and 12.5%. However, these values are still not sufficiently low in order to use this method of authentication by itself. Other behavioral authentication techniques should be used in combination with the proposed one, in order to obtain values for the EER that are close to zero.

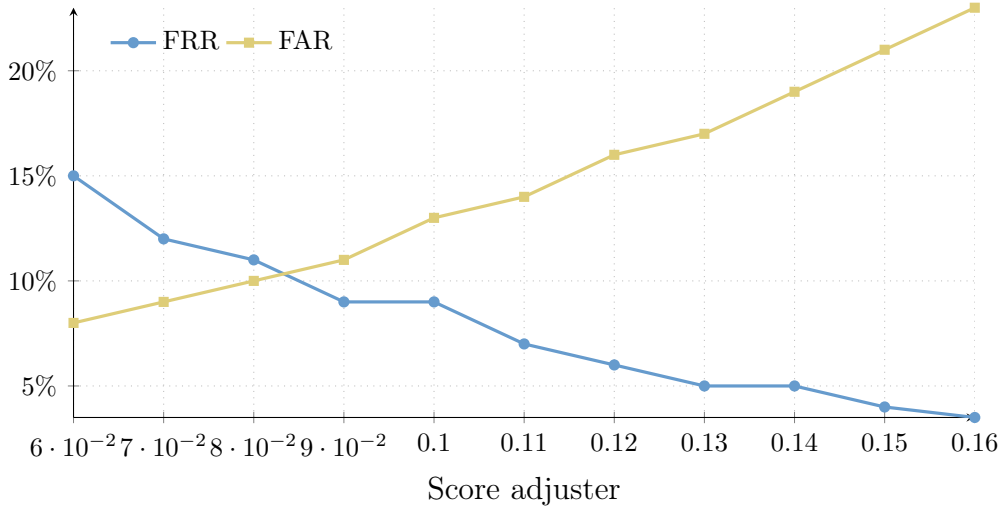
5.3.3 Dataset 3: results and observations

The third dataset is made of typing samples of 10 users, who transcribed the same text. The text was composed of about 950 characters. The subjects of the experiments had simply to copy the proposed text using their own smartphones, without any restrictions. For instance, a subject could have used the autocompletion, the suggestions proposed by his or her virtual keyboard, or he or she could have used a third-party keyboard. Precisely, half of the subjects used an Android device and the others an phone device; they all used the autocompletion and one of them used the SwiftKey[3] third-party keyboard for Android devices.

[3]https://swiftkey.com/en/keyboard/android



(a) FAR and FRR obtained by changing the number of digraphs per sample.



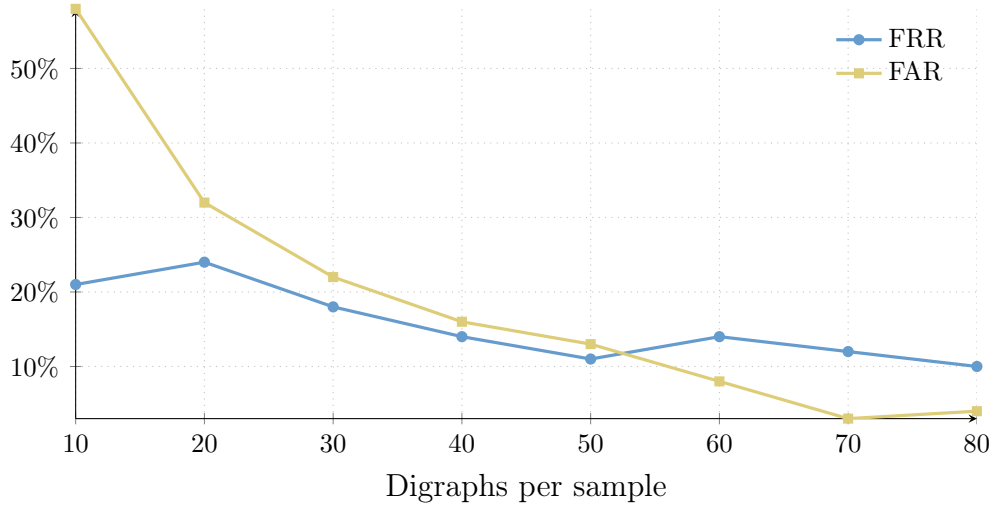
(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.3: Dataset 2: results for authentication mode.

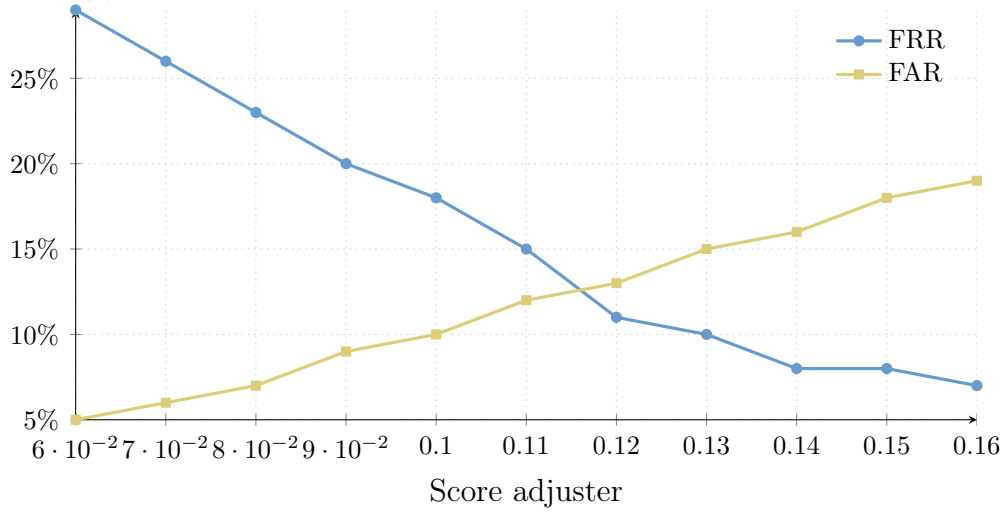
Dataset 3: authentication mode

The results of the experiments are presented in Figure 5.5. Figure 5.5a illustrates the results obtained by changing the number of *digraphs* per typing sample. The results show that by increasing this number the overall performance increases significantly. The best increase of performances is registered when the typing samples are made of 40-50 *digraphs*, with values of FAR and FRR are between 15% and 20%.

For the second experiment the number of *digraphs* per typing sample has been set to 50. The EER, that is represented by the intersection of the FAR and FRR in



(a) FAR and FRR obtained by changing the number of digraphs per sample.



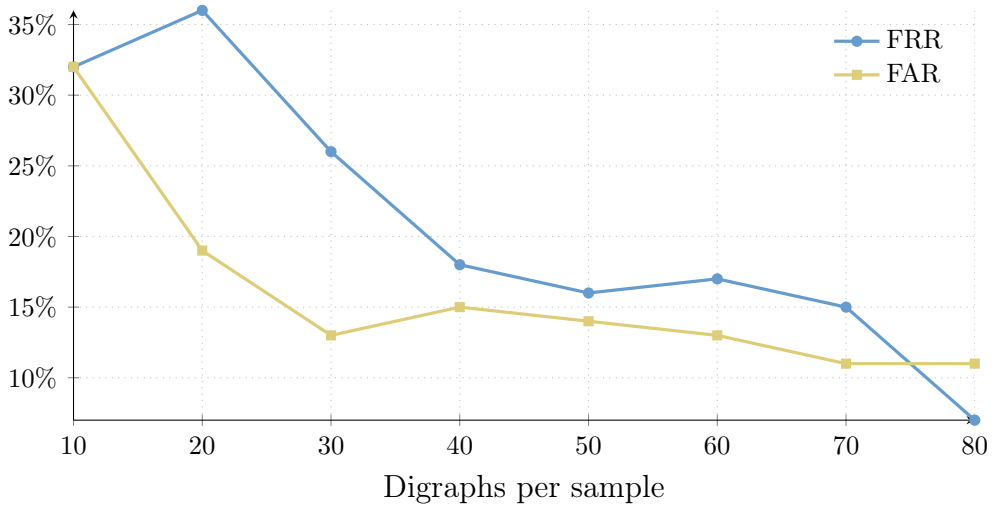
(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.4: Dataset 2: results for anomaly detection mode.

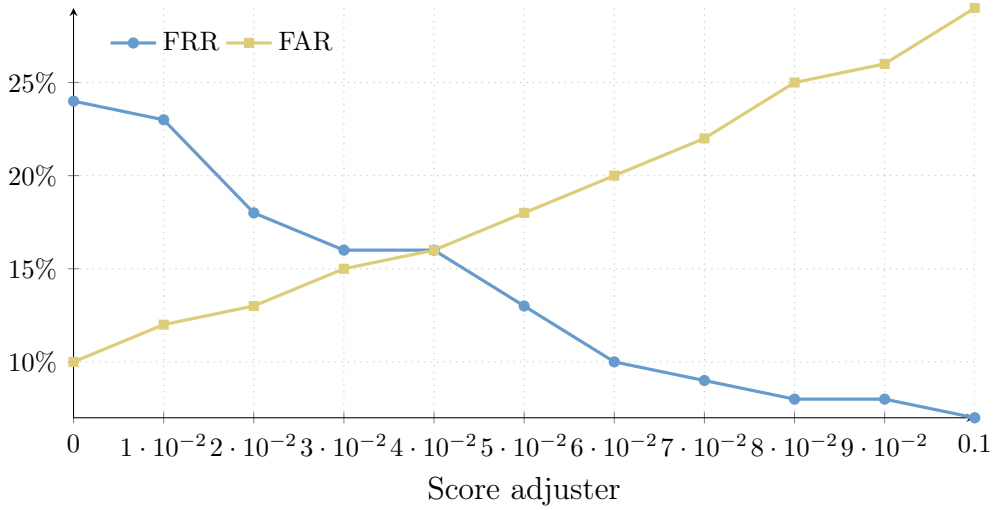
Figure 5.5b, is equivalent to 16%.

Dataset 3: anomaly detection mode

For the anomaly detection mode the results are slightly worse. The EER, calculated using 50 *digraphs* per sample, increases by 2%, reaching the 18%.



(a) FAR and FRR obtained by changing the number of digraphs per sample.

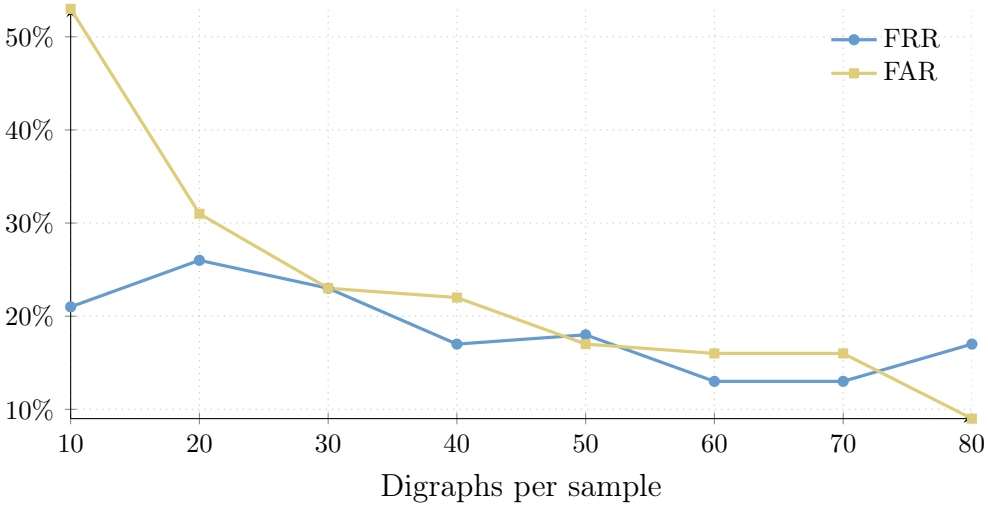


(b) FAR and FRR obtained by changing the value of the score adjuster.

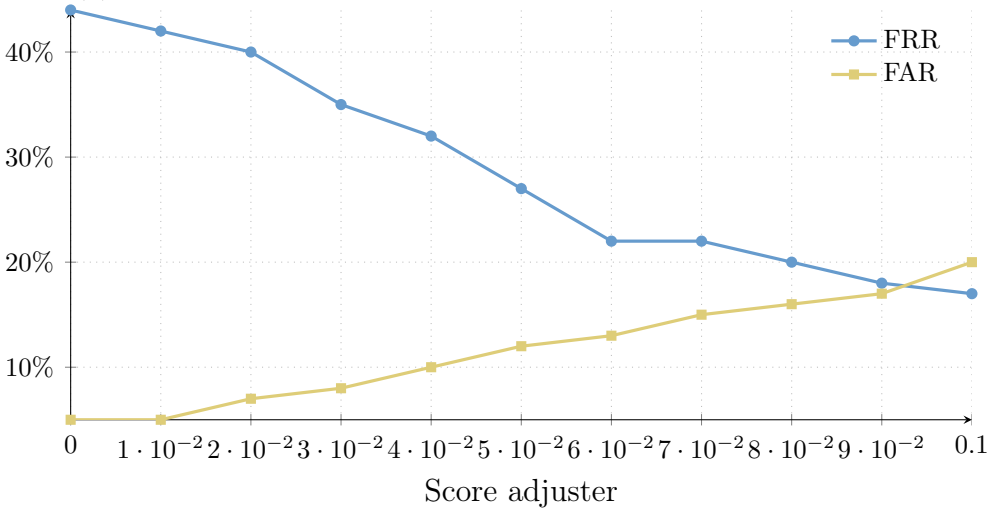
Figure 5.5: Dataset 3: results for authentication mode.

Dataset 3: observations

As we could be expected, the overall performance of the system using this third dataset decreased, with respect to the previous one. This could be due to the fact that the typing rhythm on a real keyboard is more personal than the one that people have on a smartphone devices. A computer keyboard is larger and each person writes on it with different fingers. For instance, one could only use his/her 4 fingers, whereas another person may use all the fingers of the two hands. On the contrary, the typing rhythm on a smartphone is less personal. To type on a



(a) FAR and FRR obtained by changing the number of digraphs per sample.



(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.6: Dataset 3: results for anomaly detection mode.

smartphone the thumbs are normally, since the others fingers are used to hold the device. In addition, on computer keyboards there are neither autocompletions nor suggestions. Nevertheless, the performance is still acceptable if we consider using our authentication framework along with other behavioral authentication techniques, or simply other methods of authentication.

5.4 Privacy evaluation

In order to perform the privacy evaluation of the system, the same kind of experiments have been conducted, applying the different privacy preserving techniques implemented. Since the keystroke dynamics authentication framework we implemented is conceived for the analysis of the typing behavior of users who type freely on smartphones, the experiments have been only performed using the third dataset. The next subsections illustrate the results obtained applying these techniques.

5.4.1 The "permutation" technique

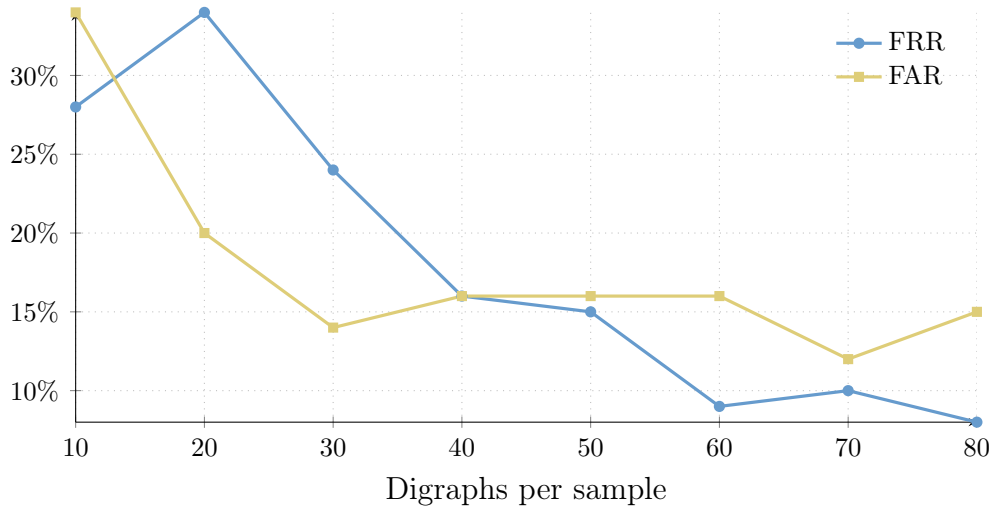
The *permutation* technique is our first privacy-preserving technique. The results for both the two modes of operation, shown in Figure 5.7 and Figure 5.8, are similar to the one obtained without applying this technique. By permutating the keystrokes before sending them to the server has the only drawback that the *trigraphs* cannot be used for the to calculate the disorder between the typing sample. This is because the *trigraphs* are created on the server-side application, by combining subsequent *digraphs*. The problem here is that the order of the *digraphs* does not reflect the order in which the *digraphs* were originally typed. For this reason, the "R_3" measure, hence the "$R_{2,3}$" one, cannot be computed. The resulting distance measure that can be used is only the one composed of "$R_2 + A_2$", since only *digraphs* are required. The results demonstrate that by applying this technique overall performance does not decrease. The EER obtained is 16% for the *authentication mode* and 18% for the *anomaly detection mode*. The same EERs that were obtained without applying the *permutation*.

5.4.2 The "substitution" technique

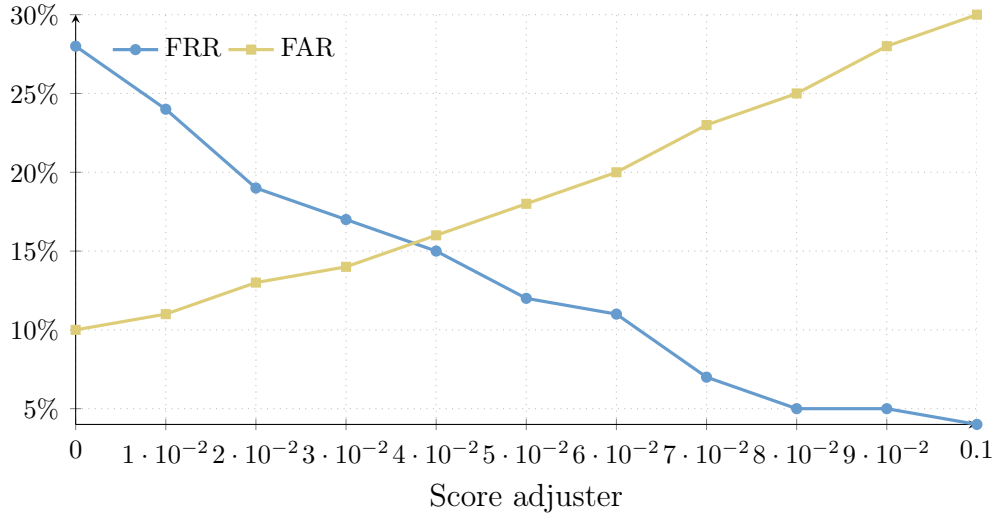
By applying the *substitution* we have the same drawback encountered with the previous technique. The *trigraphs* cannot be used for the classification task. Moreover, since the typing samples are altered, due to the *substitution* operation, we have obtained worse results. The EER is around 30% for both the two modes of operation. The results are presented in Figure 5.9 and Figure 5.10.

5.4.3 The "suppression" technique

For what concerns the *suppression* technique there can be two modes of applying this technique. The first one avoids sending more than a certain percentage of keystrokes collected by the client. In this case we would have the same results obtained using the default configuration. The consequence will be that the system will take more time to collect the required number of *digraphs* to perform the decision making operation. Another possible application would be to send a fewer number



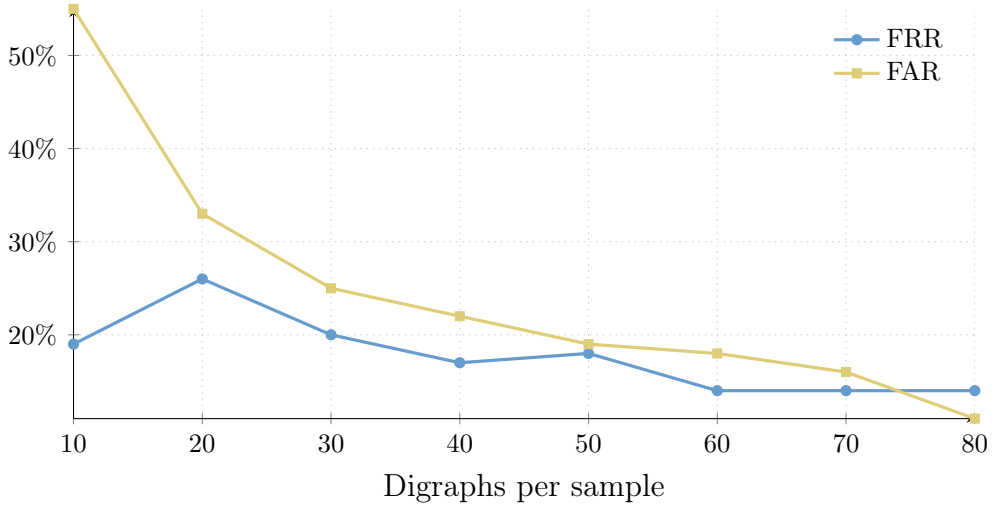
(a) FAR and FRR obtained by changing the number of digraphs per sample.



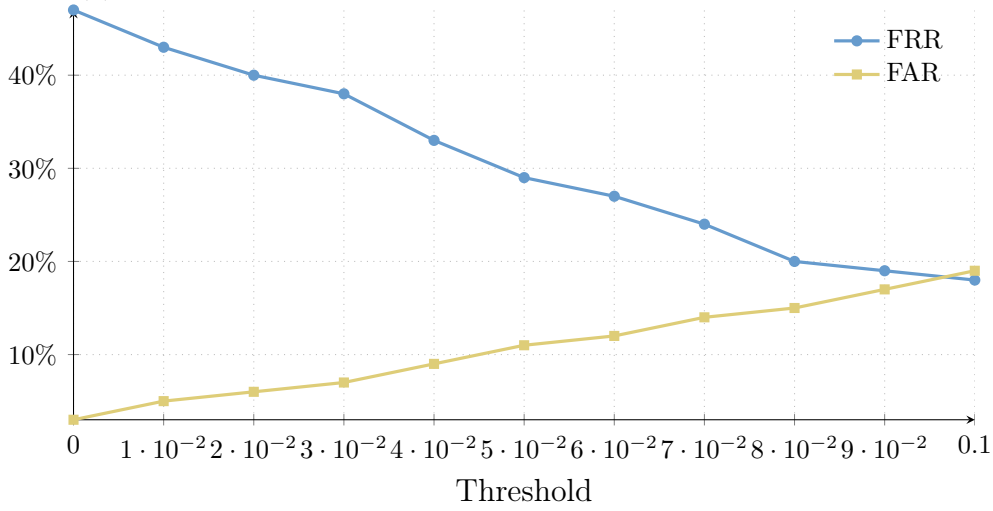
(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.7: Dataset 3: results for authentication mode with the *permutation* technique.

of keystrokes and build typing samples that are shorter. To observe the impact of this second method on the authentication accuracy the reader can refer to Figure 5.7 and 5.8.



(a) FAR and FRR obtained by changing the number of digraphs per sample.

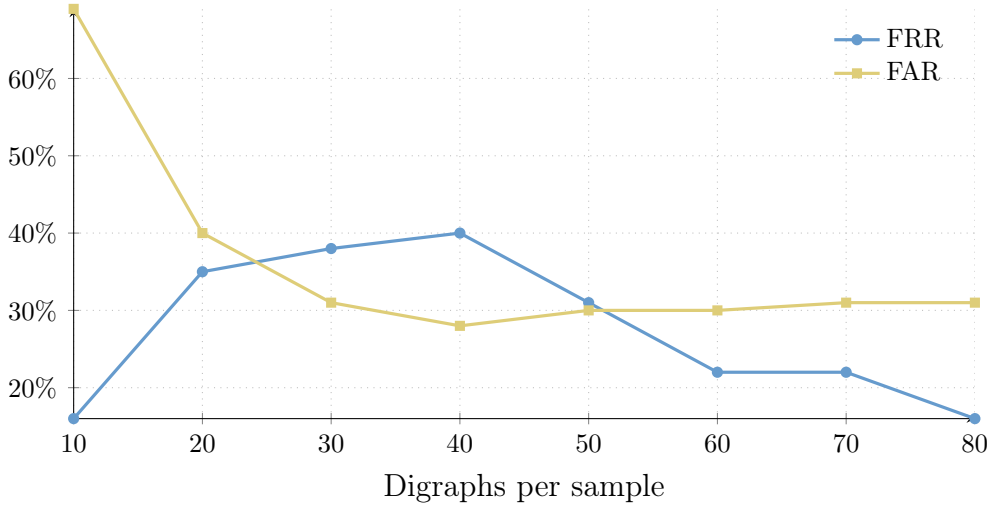


(b) FAR and FRR obtained by changing the value of the score adjuster.

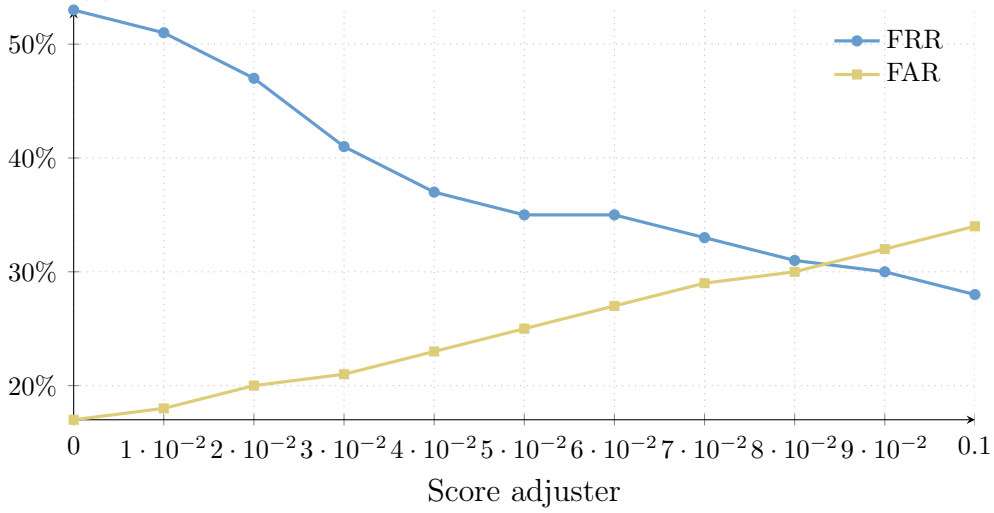
Figure 5.8: Dataset 3: results for anomaly detection mode with the *permutation* technique.

5.4.4 Combining the privacy-preserving techniques

The techniques presented are easily composable. The results obtained using the *substitution* are the same that would have been achieved by combining this technique with the *permutation*. Moreover, the two techniques could also be applied in combination with the third technique proposed, in order to strengthen the security of the system, with the consequence that more time will be required to receive authentication responses.



(a) FAR and FRR obtained by changing the number of digraphs per sample.

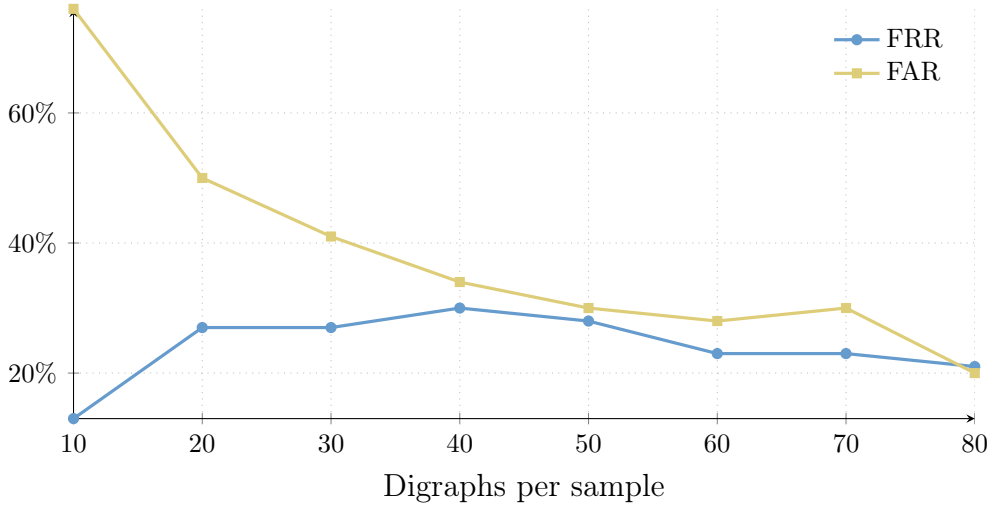


(b) FAR and FRR obtained by changing the value of the score adjuster.

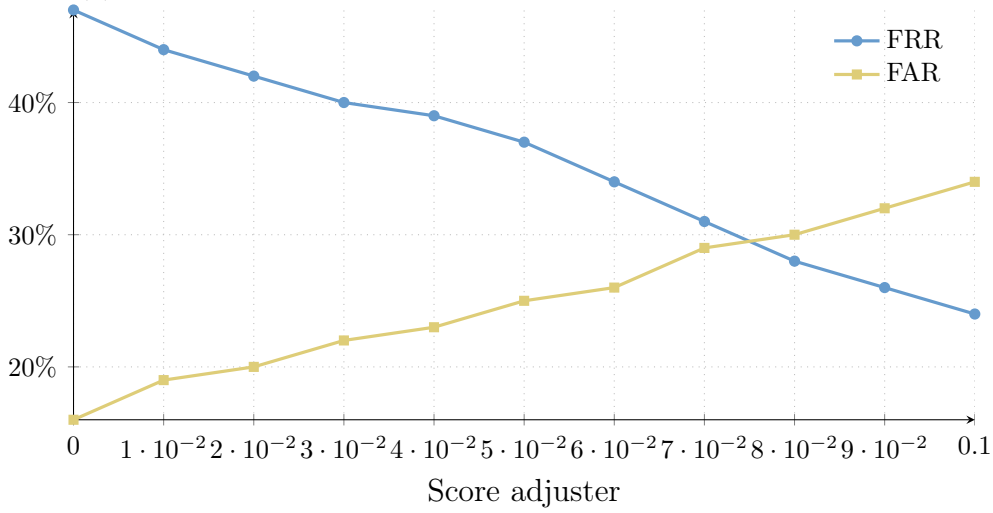
Figure 5.9: Dataset 3: results for authentication mode with the *substitution* technique.

5.4.5 Privacy evaluation: observations

The techniques proposed certainly increase the privacy of users, since they make harder the reconstruction of the text typed. However the privacy level depends significantly on the number of keystrokes sent per typing sample. For instance, if a typing sample is composed of few *digraphs*, after applying the *permutation* technique the reconstruction of the original text is still easy for an attacker. The same applies for the *substitution* technique. In this case, an attacker that knows in advance the



(a) FAR and FRR obtained by changing the number of digraphs per sample.



(b) FAR and FRR obtained by changing the value of the score adjuster.

Figure 5.10: Dataset 3: results for anomaly detection mode with the *substitution* technique.

privacy-preserving technique used, could try to infer the original text by doing some frequency analysis of the combinations of letters.

5.5 Validity threats

The main validity threat, when biometric authentication systems have to be tested, is that a lot of data produced by different users is required. The problem is that there is not a specified threshold for the number of this data. In a realistic scenario,

the system should be able to manage a large number of users. For this reason, the more data is available in the evaluation phase, the more the results will be accurate and realistic.
For each of the datasets used, the number of data samples per user was sufficiently high. Whereas, more realistic results would have been obtained with more users for each dataset. For the first two dataset, data from about 20 users has been tested. This number can be satisfactory, since other experiments have been performed with these same datasets. However, for the third dataset, the subject of the experiments were only 10. A system with only 10 users registered may not be realistic and, as a consequence, also the results obtained. However, the research of Messerman et al. [32] demonstrated that, after a certain number of users considered in the experiment, increasing this number did not influence their results. The techniques of authentication used in their work were similar to the ones used in the solution proposed. For this reason, we believe that the same reasoning could apply in the case of the third dataset. However, more experiments should be performed to prove that.
Another validity threat is that the data, used for the experiments, has been collected in a controlled environment. This is especially the case for the first and the third dataset. Smartphones could be used while the user is in motion. This will produce possibly different authentication results. To validate the results obtained, more experiments should be carried out in order to test the system in the different contexts in which it could be used.

5.6 Qualitative evaluation of the non-functional requirements

If we consider the non-functional requirements we listed in Chapter 3, we can observe that the first two focus mainly on performance, whereas the third one focuses on privacy. For the first two, we stated that: (1) the system has to perform the authentication operation in a reasonable time, in order to block illegitimate users in a timely manner, and (2) the system has to scale in order to prevent performance bottlenecks. It is logical that the two requirements are strictly correlated, since scaling the system would help improving the response time of the authentication. For this reason, these two requirements can be evaluated simultaneously.

Unfortunately, we could not perform this evaluation, due to the fact that there was not enough user data in order to simulate a real scenario of a system made up of hundreds or thousands of users. However, a possible solution to perform this evaluation could have been stressing the system by issuing multiple concurrent authentication requests. In our case, we would have sent a huge amount of keystrokes coming from

different users and calculated the rate at which those keystrokes would have been successfully handled [22]. We would have made use of the *Universal Scalability Law (UCL)* in order to give a quantitative evaluation of the scalability of our authentication system. Despite these limitations encountered, we can still provide some observations.

It has to be noticed that the computational performance depends on the configuration used for the system. For instance, the space required to store a single user model depends on the number of keystrokes used to train the model. The keystrokes have to be saved, since they are used to compare new typing samples against the user models of the system and to retrain the model (if required). If we consider the distributed approach we proposed in Section 4.6 to scale out the system, the amount of space for the user models will affect the update of these models among different machines.

The other configuration aspect that influences the performance is the mode of operation used. By using the *anomaly detection mode* the effort required to authenticate a user will not depend on the number of users of the system, since his/her typing samples will be compared only against his/her typing model. Therefore, the computational overhead will be linearly proportional to the number of users that sends keystrokes to the server at the same time. Whereas, by using the *authentication mode* the system has to compare a user typing sample against the different user models. In this case, the authentication will have a quadratic complexity w.r.t. the number of users of the system. If we indicate with N the total number of the users managed by the system, we can state that for the first case the complexity of the authentication operation is $O(N)$ in the worst case, whereas in the second case it is $O(N^2)$. Instead, if we compare a user sample against a fixed number of users, let us say X users, as we proposed also in Section 4.6, the complexity will be $O(N \cdot X)$. Moreover, if we split the users across different machines, where each machine manages M users, with M $<$ N, we will have a complexity of $O(M \cdot X)$.
To have an idea, using typing samples made of 50 *digraphs* and user models composed of 600 *digraphs*, we calculated that the time required to compute the distance between a typing sample and a single user model is about *10 ms*. Thus, if this comparison is performed against 10 user models, e.g. using the *authenticate mode*, the time required will be around *100ms*, and so on. The machine used for the experiments had an Intel Core i5 processor with a 2,4 GHz clock speed.

The third non-functional requirement we specified was that the authentication system should avoid service providers from being able to reconstruct the text typed by users. In order to meet this requirement, we implemented three privacy-preserving techniques. The problem, in this case, is that it is not trivial to evaluate how our

techniques make it more difficult to reconstruct the typed text. This highly depends on the ability and ingenuity of attackers and the effectiveness of our techniques cannot be easily evaluated in a quantitative manner.

Chapter 6

Conclusion

This final chapter summarizes the conclusions about the work carried in the frame of this master thesis. The aim of this research was to investigate the effectiveness of using keystroke dynamics to authenticate users continuously and transparently specifically on smartphones and with consideration of privacy. In order to do that, we implemented a client-server prototype that provides support for privacy and we evaluated our solution by using real user data. This final chapter is structured as follows. The first section presents the work that have done in order to achieve the goals presented in Chapter 1. In addition, we present the results of the evaluation of the proposed solution, along with the final insights. We conclude the chapter by discussing opportunities for future research work in this area.

6.1 Key contributions

The contributions of this thesis are (1) a server and client side implementation for keystroke dynamics authentication on mobile, (2) the evaluation of state-of-the-art keystroke dynamics techniques using real user data, (3) the implementation of privacy extensions on top of existing algorithms.

For contribution (1) a prototype of the continuous authentication framework has been developed. The authentication services are exposes by means of RESTful APIs that can be contacted by clients and IAM systems through simply HTTP requests. By adopting this design the clients are untied from the server application architecture, and it leaves room for the extension and integration with other authentication services.

Contribution (2) has been achieved by evaluating our solution which makes use of state-of-the-art algorithms proposed for keystroke dynamics analysis. The evaluation has been performed using three different datasets, in particular real data was

appositely acquired in order to test the system in its real usage environment (users who type freely on smartphones). The results obtained give us the evidence that our solution analyzes better the typing behavior of users when they type free text rather than fixed-length strings. They also show that more users have a more particular typing behavior when they type on computer keyboard rather than on smartphones. In the first case the EER of the system was 10% and 12% when using the *authentication mode* and the *anomaly detection* one respectively. Whereas, higher values of EER has been obtained when users type on smartphones, the EER was 16% using the *authentication mode* and 18% using the *anomaly detection* one.

For contribution (3) we researched, implemented and properly evaluated three privacy-preserving techniques, named *permutation*, *substitution* and *suppression*. These techniques can be used in order prevent *honest but curious* service providers to reconstruct what users type. The impact of using these techniques on the authentication accuracy demonstrated that there is a trade-off between the security of the system and the privacy of the users. Precisely, for the *permutation* technique we obtained the same values of EER, for both the two modes of operation, as the ones registered without applying this technique. Whereas, by applying the *substitution* technique, we registered an increase of 15% for the first task, and of 11% for the second one. For the third *suppression* technique, the EER increased as much as the number of keystrokes suppressed. The privacy-security trade-offs have to be regulated based on application requirements.

6.2 Future work

As a possible future work first of all a more exhaustive evaluation of our solution needs to be performed. This implies that more user data is required in order to validate the authentication accuracy we obtained in our experiments. This also applies for the experiments we performed with the privacy-preserving extensions. In addition, a qualitative evaluation of the two strategies we proposed to achieve scalability should be carried out. It could be also interesting to combine keystroke dynamics with other behaviometrics in order to characterize and verify better user behavior. A possible solution could be to implement another behaviometric authentication framework that can work in parallel with our proposed one. The final authentication decision would then be based on both the results obtained by the two systems, weighted proportionally w.r.t. to their respective accuracy.

Bibliography

[1] M. Abramson and D. W. Aha. User authentication from web browsing behavior. Technical report, DTIC Document, 2013.

[2] P. E. Agre and M. Rotenberg. *Technology and privacy: The new landscape*. Mit Press, 1998.

[3] A. Alzubaidi and J. Kalita. Authentication of smartphone users using behavioral biometrics. *IEEE Communications Surveys & Tutorials*, 18(3):1998–2026, 2016.

[4] M. Antal and L. Nemes. The mobikey keystroke dynamics password database: Benchmark results. In *Software Engineering Perspectives and Application in Intelligent Systems*, pages 35–46. Springer, 2016.

[5] M. Antal, L. Z. Szabó, and I. László. Keystroke dynamics on android platform. *Procedia Technology*, 19:820–826, 2015.

[6] D. Buschek, A. De Luca, and F. Alt. Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1393–1402. ACM, 2015.

[7] K. Calix, M. Connors, D. Levy, H. Manzar, G. MCabe, and S. Westcott. Stylometry for e-mail author identification and authentication. *Proceedings of CSIS Research Day, Pace University*, pages 1048–1054, 2008.

[8] P. Campisi, E. Maiorana, M. L. Bosco, and A. Neri. User authentication using keystroke dynamics for cellular phones. *IET Signal Processing*, 3(4):333–341, 2009.

[9] N. L. Clarke and S. M. Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1–14, 2007.

[10] H. Crawford. Keystroke dynamics: Characteristics and opportunities. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 205–212. IEEE, 2010.

[11] P. X. de Oliveira, V. Channarayappa, E. O'Donnel, B. Sinha, A. Vadakkencherry, T. Londhe, U. Gatkal, N. Bakelman, J. V. Monaco, and C. C. Tappert. Mouse movement biometric system. *Proc. CSIS Research Day*, 2013.

[12] K. Delac and M. Grgic. A survey of biometric recognition methods. In *Electronics in Marine, 2004. Proceedings Elmar 2004. 46th International Symposium*, pages 184–193. IEEE, 2004.

[13] I. Deutschmann, P. Nordström, and L. Nilsson. Continuous authentication using behavioral biometrics. *IT Professional*, 15(4):12–15, 2013.

[14] S. Dhage, P. Kundra, A. Kanchan, and P. Kap. Mobile authentication using keystroke dynamics. In *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*, pages 1–5. IEEE, 2015.

[15] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1388–1401. ACM, 2016.

[16] T. Feng, X. Zhao, B. Carbunar, and W. Shi. Continuous mobile authentication using virtual key typing biometrics. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 1547–1552. IEEE, 2013.

[17] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, DTIC Document, 1980.

[18] H. Gascon, S. Uellenbeck, C. Wolf, and K. Rieck. Continuous authentication on mobile devices by analysis of typing motion behavior. In *Sicherheit*, pages 1–12. Citeseer, 2014.

[19] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos. I sensed it was you: authenticating mobile users with sensor-enhanced keystroke dynamics. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 92–111. Springer, 2014.

[20] D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)*, 8(3):312–347, 2005.

[21] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42. ACM, 2009.

[22] T. Heyman, D. Preuveneers, and W. Joosen. Scalability analysis of the openam access control system with the universal scalability law. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 505–512. IEEE, 2014.

[23] B. Inc. Behaviometrics. `https://www.behaviosec.com/behaviometrics`. Accessed: 2017-05-25.

[24] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14(1):4–20, 2004.

[25] L. Jain and J. Vyas. Security analysis of remote attestation. Technical report,

CS259 Project Report, 2008.

[26] W. J. Jan Spooren, Davy Preuveneers. Leveraging battery usage from mobile devices for active authentication. *Mobile Information Systems*, 2017.

[27] G. Kambourakis, D. Damopoulos, D. Papamartzivanos, and E. Pavlidakis. Introducing touchstroke: keystroke-based authentication system for smartphones. *Security and Communication Networks*, 2014.

[28] M. Karnan, M. Akila, and N. Krishnaraj. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2):1565–1573, 2011.

[29] K. S. Killourhy and R. A. Maxion. Free vs. transcribed text for keystroke-dynamics evaluations. In *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, pages 1–8. ACM, 2012.

[30] W. Liu, A. S. Uluagac, and R. Beyah. Maca: A privacy-preserving multi-factor cloud authentication system utilizing big data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 518–523. IEEE, 2014.

[31] Y. Meng, D. S. Wong, R. Schlegel, et al. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In *International Conference on Information Security and Cryptology*, pages 331–350. Springer, 2012.

[32] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–8. IEEE, 2011.

[33] J. V. Monaco and C. C. Tappert. Obfuscating keystroke time intervals to avoid identification and impersonation. *arXiv preprint arXiv:1609.07612*, 2016.

[34] M. Nauman, T. Ali, and A. Rauf. Using trusted computing for privacy preserving keystroke-based authentication in smartphones. *Telecommunication Systems*, pages 1–13, 2013.

[35] M. Obaidat and B. Sadoun. Keystroke dynamics based authentication. In *Biometrics*, pages 213–229. Springer, 1996.

[36] M. Raza, M. Iqbal, M. Sharif, and W. Haider. A survey of password attacks and comparative analysis on methods for secure authentication. *World Applied Sciences Journal*, 19(4):439–444, 2012.

[37] S. Sen and K. Muralidharan. Putting 'pressure' on mobile authentication. In *Mobile Computing and Ubiquitous Networking (ICMU), 2014 Seventh International Conference on*, pages 56–61. IEEE, 2014.

[38] M. S. Siddiqui and D. Verma. Evercookies: Extremely persistent cookies. *International Journal of Computer Science and Information Security*, 9(5):165, 2011.

[39] C.-J. Tasia, T.-Y. Chang, P.-C. Cheng, and J.-H. Lin. Two novel biometric features in keystroke dynamics authentication systems for touch screen devices.

Security and Communication Networks, 7(4):750–758, 2014.

[40] Techopedia. What is a filter bubble? `https://www.techopedia.com/definition/28556/filter-bubble`. Accessed: 2017-05-25.

[41] P. S. Teh, A. B. J. Teoh, and S. Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013, 2013.

[42] P. S. Teh, N. Zhang, A. B. J. Teoh, and K. Chen. A survey on touch dynamics authentication in mobile devices. *Computers & Security*, 59:210–235, 2016.

[43] M. Trojahn, F. Arndt, and F. Ortmeier. Authentication with keystroke dynamics on touchscreen keypads-effect of different n-graph combinations. In *Third International Conference on Mobile Services, Resources and Users (MOBILITY)*, pages 114–19, 2013.

[44] S. D. Warren and L. D. Brandeis. The right to privacy. *Harvard law review*, pages 193–220, 1890.

[45] A. F. Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

[46] K. Wuyts. Privacy threats in software architectures. 2015.

Appendices

Appendix A

Popularizing article

Continuous authentication with behaviometrics on smartphones

Gabriele Vassallo, *KU Leuven*

Nowadays almost all web-applications and online services deal with our personal information, such us identity (social networks), friends (messaging apps) and our money (mobile payments). For this reason more sophisticated mechanisms of authentication are becoming widely used to assure stronger security for our personal data. Interest is growing to behavioral authentication techniques, e.g. keystroke dynamics, as a next step in a multi-factor authentication mechanism, since it is powerful in identifying intruders and does not require additional hardware given that it uses built-in components of the devices we use during our day life.

The goal of the research is to investigate which kinds of techniques fulfill better the task of analyzing this data, and how much these techniques can scale, since they have to continuously process a big amount of data and, at the same time, be responsive as much as possible, while preserving security and privacy of users.

Introduction

Ensuring security in modern applications is becoming more and more challenging. New security threats are proving that single-factor authentication is not robust enough to assure a reasonable level of security, especially for applications that deal with sensitive data.

Single-factor authentication (SFA) is a process for securing access to a given system, that identifies the party, which is requesting access, through only one category of credentials. These credentials, that should be something that only the user can know, prove the identity of that user. The password-based authentication method is the most common example of SFA.

It relies on the fact that: only the user knows his/her own password, the password is highly difficult to be guessed, and that the authentication system is robust enough to prevent intrusions and dictionary based attacks [1]. In order to carry out these last mentioned attacks, a malicious user has to enumerate, and try, all the possible passwords until he/she finds the right one. In order to decrease the number of passwords to try, a dictionary of common passwords can be used.

Password-based authentication can be considered as the weakest form of authentication, but it is still the most widely adopted. This is because it has low cost of implementation, it is intuitive and it is not intrusive, since it does not require users' personal information, such as biometric data (e.g. eye iris, fingerprints, face recognition, etc.).

The other reason is that companies implemented this method of authentication in their system several years ago, therefore a transition to new standards could be really expensive.

Multi-factor authentication (MFA), also called Multi-Modal authentication, adds new layers of security in a system, granting the access to a resources after the verification of at least two type of credentials, named *factors*, provided by the user. The authentication factors have to be of different natures, in particular they should be:

- something secret to the user, e.g. passwords, PINs, etc.
- something the user possesses, e.g. a USB with a token, a bank card, etc.

- some physical characteristics of the user (biometrics), e.g. fingerprints, eye iris, typing pattern behavior, etc.

An example of MFA authentication is the combination password and security token we use to access a bank account. The security token is a passcode with short lifetime that is generated randomly and that only a specified user can read it , e.g. through a USB stick, such that if an attacker guess it, it will not work for the next session. The good advantage of MFA is that it can be extended as much as the system requirements. This extension is simply provided by the composition of new factors, which will implement additional steps for the authentication operation.

Biometric can be seen as secure factor in authentication, considering that it validates the identity of a user, who wishes to sign into a system, by measuring some intrinsic characteristics of that specific user, i.e. his/her unique biological attributes.

Nevertheless, new researches are showing that also this kinds of verifications can be easily circumvented with the help of new technologies. An attacker can extract eye iris simply from a photo of a targeted user, whereas fingerprints can be reproduced just with a camera and glue, or in extreme cases by physical attacks.

For these reasons, and since these techniques can be considered intrusive, not hygienic, and risky, the interest is moving to new forms of authentication, which are based on users' behavioral characteristics.

Figure 1: *Biometrics ©findbiometrics.com*

Behavioral Authentication

Behavioral Authentication (BA) is the cutting-edge technology that analyzes user behavior to help verifying a user's identity by continuously collecting information about his/her usage of an application. This information could be keystroke dynamics, mouse events or even more unique characteristics, e.g. motion behavior.

BA is a good trade off, since it provides unique characteristics of the user, as biometric authentication does. On the other hand, it is not intrusive and user friendly (it runs in the background providing passive authentication).

These techniques can be exploited to develop **Continuous Authentication** systems, that continuously analyze the users' behavior in order to authenticate them. In this way, the risk of session hijacking or Man in the Middle (MITM) attacks are mitigated, since the users are re-authenticated during their sessions. Considering that a lot of applications ask credentials only at the first usage (*one-shot authentication*), this re-authentication mechanism adds a worthwhile layer of security.

However, new types of attack could be developed, such as the observation of a user' behavior in order to mimic it and circumvent the authentication system. There are already some behavioral based authentication solution providers, one of the most well-known is BeavioSec [4]. They assure that with their solution it can be possible securing end-user accounts, as well as preventing fraud and delivering continuous authentication.

This new trends are showing that sooner BA will be brought to the mainstream, thanks to the support of new data analysis techniques and algorithms. The techniques developed in the last years had satisfactory results, but the key challenge at the moment is to apply them on smartphones, and this could not be very straightforward. We use these devices are during our day life, sometimes while we are in motion, or in general in 'not controlled environments'. There are several data that can be combined to construct users' behavior patterns, the goal is to find which ones embed most meaningful characteristics.

Machine Learning for Authentication

Machine learning is the subfield of computer science that "gives computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959).

Different numbers of algorithms have been implemented during the last years, with the aim of building models of data that has to be analyzed, such that predictions for future data can be automated without manual intervention.

Machine Learning (ML) techniques have attracted the interest of the security community since

they can be used to correlate security-related data efficiently and automate the process of identifying anomalies.
Most of the techniques used for BA rely on ML to construct user models and verify their identities. For instance, a user can be rejected when his/her analyzed behavior differs from his/her normal one, described in his/her model.
For this kind of tasks it has to be taken into account that the accuracy of the techniques used is much more relevant than in other scenarios of data analysis. False positive predictions, e.g. a suspicious behavior that is not detected as an intrusion, or a false negative one, e.g. an authorized user that is classified as an intruder, have to be limited, since they can lead to unpleasant outcomes.

Extending a state of practice authentication platform

Let us consider as a use case scenario a **messaging application** for smartphone devices (e.g. Whatsapp, Telegram, Messenger etc.). The security goal of the application is that user accounts cannot be violated. In addition, users may want to have the certainty about the identity of the other users with whom they are messaging. Thus, the authentication has to be personal but also mutual. To design the authentication service to accomplish these goals, different solutions can be explored. Here we propose two configurations that rely on the **client/server** architecture, and both provide **continuous authentication** based on **behavioral verification**.
In this conformation two parties are involved: the **client**, that is the application used by the user, and the **server**, which is the identity that provides the authentication mechanism (and possibly the messaging services). The client is an **Android** application that interacts with the server through HTTP requests.
In our first solution, the client sends authentication information throughout the entire usage of the application. This information is constantly processed by the server which has to detect whether the user is the legitimate one. In the second solution we propose, the processing of the authentication information is split both on the client application and onto the server, such that less information is sent during the session and the server is much less overloaded. For both the solutions, **Keystroke dynamics** is exploited as first behavioral data to perform the authentication.

Figure 2: ©*coolestech.com*

Keystroke Dynamics

Keystroke dynamics or **typing dynamics** refers to the automated method of identifying or confirming the identity of an individual based on the manner and the rhythm of typing on a keyboard. Keystroke dynamics is a behavioral biometric, this means that the biometric factor is 'something you do'[3].
Two measurements can be taken: dwell time and flight time.

- The **Dwell Time**, or **Duration**, is the time duration that a certain key is pressed.
- The **Flight Time**, or **Latency**, is the time duration in between pressing two consecutive keys.

In order to calculate these values, the application has to intercepted two events, which are the *KeyDown* and *KeyUp* events. The first is generated when a key is pressed, the latter when the key is released. The dwell time is measured as the time elapsed between the *KeyDown* and *KeyUp* events on the same key. The flight time is measured as the time elapsed between two consecutive *KeyDown* events.

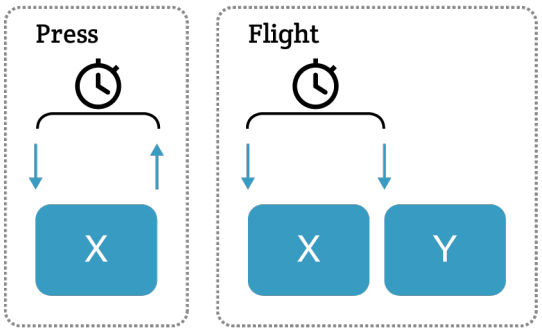


Figure 3: ©*behaviosec.com*

Other events could be recorded along with keystroke dynamics. For mobile devices the data intercepted

from the Operating System may help characterizing more precisely the user interactions with the application. For instance, **touch events**, such as *hit zone*, *pressure* applied on the screen and *multi-touch gestures* could be considered for this analysis, as well as data collected by the sensors of the smartphone devices. This last kind of data could be useful to profile the user's behavior based on the contextual usage of the application. For instance, *gyroscope* and *accelerometer* measurements could be used to determine whether the user is walking, and use this information to adapt the verification of his/her typing pattern [5].

Data Preprocessing

Data preprocessing and sanitation has to be done on the input received in order to eliminate outliers. These outliers are data that has no meaning for the analysis and that may be due to variability in the measurement and compromise the system reliability. For keystroke dynamics to compute the Flight Time between two keys, the *digraph* data type is used. This *digraph* is a graph with two nodes, the two consecutive keys pressed, and the edge that links them weighed with the latency. Some filters can be applied on the generated digraphs to eliminate outliers. As a first crude filter, digraphs with times lesser and greater than given thresholds should not be considered. Digraph with non-printable ASCII characters should also be excluded from the analysis [6].

Possible algorithm configurations

For the modeling and analysis of user data different algorithms can be explored. The techniques used belong mainly to two families, that are: *classification* techniques and *clustering* techniques.

Classification

By using the *classification*, the objective is to divide the input data into two or more classes. Then, for each new data analyzed, this data has to be assigned to one of these classes, or to different classes, i.e. *multi-label classification*. The most common usage of classification is the *binary classification*. It divides the input space in two classes, a positive and a negative one, and assigns the new data to one of them.
In the context of authentication, *multi-class classification* and *binary classification* are both suitable. *Multi-class classification* can be exploited by instantiating a new class for each user of the system. Then, for each new input from a specific user, if it is not predicted as belonging to correct user, an alert can be generated. *Binary classification* behaves differently, since a classifier is used for each user. Once the model is built, if new data are classified as negative, probably it does not belong to the legitimate user. The latter example sounds much better in terms of efficiency, since no other data belonging to other users is involved in the analysis. Moreover, in the first scenario, the general model represents the **single point of failure (SPOF)**; once it is broken, the whole system will not work anymore.
In practice, the situation is different for *binary classification*, since, in order to construct the model, the algorithm needs negative instances, otherwise the system will accept all the new data, legitimate and not. To circumvent this problem, data generated from other users could be used to provide negative examples when a model has to be built. This will decrease the efficiency of the system, but it will avoid the SPOF intrinsic of the of *multi-class classification* method.
Another method to solve the problem is using *clustering* techniques.

Clustering

Clustering algorithms group a set of elements into subsets or clusters, so that similar elements are assigned to the same cluster [2].
The idea of using clustering is to build a cluster for each user. This cluster has to circumscribe the user's typical behavior. When new input are analyzed, the algorithm checks, whether or not, the new actions reflect user usual behavior. If the new input does not belong to the user cluster, e.g. its distance from the cluster is above a certain threshold, it is considered as suspicious behavior and the user could be locked out of the system.
The advantage of using clustering, besides that a definition of distance has to be introduced, is that it can work correlating only the data belonging to the particular user analyzed, without involving other users.

Preserving Privacy and Reliability

Implementing a behavioral authentication system in this way, implies that the service providers needs access to all the user keystrokes in order to perform the authentication. In particular for the first archi-

tecture presented, all the events generated by the users, i.e. what he/she typed, are sent to the server and analyzed. Privacy concerns could arise, and in order to mitigate this problem, privacy by design choices should be adopted. This is why the second configuration has been proposed. In such a configuration the objective is to send less relevant information to the server, migrating part of the computation to the client side. In this new architecture the server receives only significant information that are then processed and analyzed. It has to be taken into account that an attacker could tamper the information sent to the server and the system could be compromised more easily. For this reason, robust technique of validation of the data sent by the client has to be developed. Strong cryptography could be investigated as one possible solution. Another solution is to mutate the data that has to be analyzed before sending it, e.g. adding some noise to the data. However, by doing this, the data sent has to remain valuable for the analysis in order to provide a reliable authentication.

Evaluation

The evaluation of the two different architectures proposed, along with the algorithms used, will be carried out by testing the system with some publicly available datasets [7]. Testing the system will help identifying the filters for the data sanitation and the most appropriate feature selection. The filters will be helpful to separate outliers from the data for the keystroke dynamics analysis in order to built a robust and reliable authentication system. This evaluation will be also useful to evaluate the machine learning techniques from a privacy perspective, and to verify whether adding noise would weaken the accuracy of the system.

In conclusion

Keystroke dynamics authentication seems to be an interesting method for user authentication, but it has its drawbacks. Privacy and scalability are the obvious ones. The former especially, since using these techniques means building a model for each user by recording their behavior. This models could be exploited by service providers, but also by malicious users, in case the user personal information is lost for any reason, e.g. data breaches. The other possible usage of keystroke biometric algorithms is to identify users even if they are surfing the web anonymously. This could become another tool for companies for their targeted advertisement purposes, but also for governments for their mass surveillance programs. Hence, exploring techniques of obfuscation to protect individuals' privacy is necessary to actually use them in mainstream applications, but also in order to weaken the tracking techniques used by adversaries.

References

[1] Raza, M., Iqbal, M., Sharif, M., & Haider, W., (2012), 'A survey of password attacks and comparative analysis on methods for secure authentication', World Applied Sciences Journal, 19(4), 439-444.

[2] Tan, P.N., Steinbach, M., Kumar, V., (2006), 'Introduction to Data Mining', Addison Wesley.

[3] http://www.biometric-solutions.com/keystroke-dynamics.html (2006).

[4] https://www.behaviosec.com

[5] Abdulaziz Alzubaidi and Jugal Kalita, (2015), 'Authentication of Smartphone Users Using Behavioral Biometrics', Journal of IEEE Communications Surveys and Tutorials,

[6] Maximiliano Bertacchini, Carlos E. Benitez1 and Pablo I. Fierens, (2010), 'User Clustering Based on Keystroke Dynamics'.

[7] Vinnie Monaco, Keystroke Dynamics Datasets, http://www.vmonaco.com/keystroke-datasets

[8] Yampolskiy, R.V. and Govindaraju, V., (2008), 'Behavioural biometrics: a survey and classification', *Int. J. Biometrics*, Vol. 1, No. 1, pp.81-113.

Appendix B

IEEE article

Privacy-preserving Keystroke Authentication on Smartphones

Gabriele Vassallo
Department of Computer Science, KU Leuven
Leuven, Belgium
gabriele.vassallo@student.kuleuven.be

***Abstract*—The functionalities offered by smartphones are increasing, along with the demand of stronger security mechanisms. Password based authentication methods for access control are the only defenses that separate illegitimate users to access sensitive information stored on these devices or online services. Our research investigates the effectiveness of using keystroke dynamics to authenticate smartphones users continuously and at the same time transparently. By using these techniques, the identity of a user is verified based on his/her way of typing on the device keyboard. We propose a keystroke dynamics authentication framework whose services can be integrated into a contemporary Identity and Access Management (IAM) system to provide continuous authentication capabilities. We implemented privacy-preserving techniques for our solution – i.e. *permutation*, *substitution* and *suppression* – that can be used in order to prevent service providers from reconstructing the text originally typed by users. We evaluate our solution measuring the authentication accuracy by using real user data. We also measured and compared the impact on this accuracy when the privacy-preserving techniques are applied. The Equal Error Rate (EER) obtained by applying the *permutation* technique remained around 16% for the 'user classification' and 18% for 'user clustering', the same as the one registered without using this technique. Whereas, by applying the *substitution* technique, we registered an increase of 15% for the first task, and of 11% for the second one. For the third *suppression* technique, the EER increased as much as the number of keystrokes suppressed.**

I. Introduction

The usage of smartphones and mobile devices during our day-life has become widely common. The capabilities of these devices have increased as well, allowing users to perform disparate kinds of tasks. Rather than simply placing calls, they are used to store personal information or access online sensitive data, such as bank accounts, mobile payments, social networks, etc. For this reason, securing wisely these personal devices is vital. The access control mechanisms to access a device and the authentication methods adopted to access online services rely mostly on password based approaches. This poses a great risk to smartphones users. Attackers need only to know a single piece of information, e.g. a passcode, a PIN code, a locking pattern or a password, in order to access illegitimately a device and gather sensitive information. In addition, some of these methods can no longer be used on devices with limited interaction capabilities, e.g. smartwatches.

These are some of the reasons that motivate the growing interest in alternative authentication strategies. This is the case for what is called *active* or *continuous* authentication. With this new paradigm of authentication the identity of a user is continuously verified. The authenticity of a user identity is verified by means of behaviometric techniques that are used to monitor transparently the user interaction with the device. The aim of this research is to extend a state-of-the-art authentication system with continuous authentication capabilities, by using keystroke dynamics [1], [2], [3] as a behaviometric technique. Keystroke dynamics refers to identifying a user based on the rhythm of typing on a keyboard.

Previous research has shown that these techniques achieved good results in terms of accuracy for the authentication of users on desktop and laptop computers. However, our research investigates whether keystroke dynamics are suitable for the authentication of smartphones users given the virtual nature of the keyboard. We evaluated our solution that builds on top of state-of-the-art keystroke dynamics techniques with three different datasets.

A second challenge we address in this work is that privacy concerns could arise when this technique is used to continuously authenticate against online services. *Honest but curious* service providers can use the keystrokes collected for authentication purposes, to reconstruct the original text typed by the users. Hence, we researched and implemented privacy-preserving techniques in order to mitigate these privacy threats. The rest of the paper is organized as follows: in Section II we present the related work on keystroke dynamics and continuous authentication, giving the reader some understanding of already existing techniques that we have used for our solution, which is described in Section III. Section IV illustrates the results obtained from the experiments that we have performed in order to evaluate our solution, followed by the conclusion of the paper in Section V.

II. Related work

The metrics that measure human behavior to recognize or verify the identity of a person are referred to as behaviometrics. Several studies have investigated the application of using behaviometrics in order to provide an authentication that is (a) continuous, during an entire user session, and (b) non-intrusive, since the normal user interaction with the system is analyzed. It has been demonstrated that a user identity can be recognized and verified by means of several behaviometrics, such as keystroke dynamics, mouse movements (together with display resolution) [4], CPU and RAM used [5], stylometry

[6], Web browsing behavior [7], etc. Keystroke dynamics are the techniques used to identify a user based on his/her typing behavior. Previous research [8] [5] demonstrated that keystroke dynamics gave better results w.r.t. other behaviometrics. The first research on keystroke dynamics dates back to 1980 [9]. Despite all these years, this topic is still appealing for research. Lot of studies have focused on keystroke dynamics with computer keyboards [1] [2], whereas in the last years the interest is growing on using this behaviometric for authentication on smartphone devices [10] [11] [12]. As it has been already mentioned in the introduction, the focus of this research is on this last mentioned application of keystroke dynamics. Several algorithms have been proposed for the analysis of keystroke dynamics on smartphones using machine learning techniques. We refer to a survey [13] for the comparison of the accuracy rates when using these different algorithms. Despite the high number of techniques investigated, good results, in terms of accuracy, have been registered also by applying statistical techniques [14] [15]. In particular, Gunetti et al. proposed in their work [16] a new measure, named *degree of disorder*, that can be used to distinguish whether two typing samples are likely to have been typed by the same person. This distance measure can then be exploited to build several authentication strategies. Our solution is built upon their work, even if their techniques were not developed having in mind that smartphones would have come into play.

III. Implementation of a privacy-preserving keystroke dynamics authentication system

The goal of this work is to investigate the effectiveness of using keystroke dynamics as a behaviometric for the continuous authentication of smartphones users. In order to collect and analyze this behavioral data, a client and a server component are required. The former collects the keystrokes whenever the user types on the smartphone keyboard, whereas the latter collects the keystrokes and performs the authentication operations, continuously. We adopted a design strategy such that an integration with an Identity Access Management (IAM) system can be straightforward. In particular, we analyzed the possible integration with the ForgeRock's OpenAM authentication system. OpenAM is a platform that provides out-of-the-box authentication modules, along with the possibility of making use of external authentication services. By having this integration in mind, we developed a continuous authentication framework which exposes RESTful APIs that can be contacted by the OpenAM system.

The authentication strategies implemented by this framework rely on a distance measure, proposed by Gunetti et al. [16], that is used to compare the similarity between typing samples generated by users. Thus, it is required to comprehend how this distance is calculated in order to understand how the authentication is performed. At the end of this section we present three privacy-preserving techniques that we have implemented, that can be used to prevent untrusted service providers to reconstruct the text typed by users.

A. Measuring the distance between two typing samples

A user typing sample is composed of a sequence of keystrokes. Following the approach proposed by Gunetti et al. [16], these typing samples can be reorganized in sequence of *n-graphs*, for a certain value of *n*. The *n-graph* is the data type that represents a sequence of *n* keys and the latency between them, i.e. the time duration between the pressure of the first and the *nth* one. Once the value of *n* is chosen, a distance **d** can be calculated between two typing samples **E1, E2**. This distance is computed by measuring the *degree of disorder* and the *(dis)similarity* between the two samples. To compute the *disorder* the *n-graphs* of the two typing samples are ordered by increasing time duration and filtered in a way that the two samples share only common *n-graphs*. The *disorder* can than be computed as the sum of the distances between the position of each element in the two arrays. For instance, the *disorder* between two arrays A=[2,5,1,4,3] and A'=[3,1,2,5,4] will be computed as: (2+2+1+1+4)=10. The idea behind this measure is that even if a user may type differently in terms of speed, he/she would type still faster certain combinations of keys w.r.t. others that usually are typed more slowly. The number obtained is then normalized, dividing it by the maximum value of *disorder*. The *similarity* measure, instead, takes into consideration the absolute typing speed of the different combinations of keys. To compare two *n-graphs*, if we indicate with d1 and d2 the time durations of the same *n-graph*, then two *n-graphs* are recognized as similar if $1 < max(d1, d2)/min(d1, d2) \leq t$, for a constant $t > 1$. The similarity between two samples is then calculated as: 1 - (number of similar n-graphs between S1 and S2)/(total number of n-graphs shared by E1 and E2). The final distance is produced combining these two measures described.

B. Authentication strategies

In order to perform the authentication operations, the framework has to create and store user typing models. Once this first *training* phase is completed, the verification of the user identities can be executed by comparing the user typed keystrokes against their reference models. This user model is a NxM matrix, where each row of the matrix is a user typing sample, which is composed of M *digraphs*. Once the N typing samples are collected by the system during the training phase, the mean distance between all the training typing samples is computed. For instance, suppose we have a three samples for a user A that are A1, A2, A3, and that we have:

$$\text{d(A1,A2)} = 0.312378;\ \text{d(A1,A3)} = 0.304381;\ \text{d(A2,A3)} = 0.326024;$$
$$\text{m(A)} = (0.312378 + 0.304381 + 0.326024)/3 = 0.314261$$

This *m(A)* is a number that represents how a certain user A types on a keyboard. After the training of the model, the next keystrokes received from user A will be used for the authentication. In order to verify the identity of a user, two modes of operation have been implemented, i.e. user (1) *classification* to recognize an individual in a set of users, and (2) *clustering* to verify the identity of a given user.

1) User classification: The *mean distance* (*md* for short) of a typing sample X from the user A is defined as:

$$md(A,X) = [d(A1, X)+d(A2, X)+...+d(An, X)]/n$$

where $\{A_1, A_2, ..., A_n\}$ are the typing samples collected during the training phase and d is the distance measure presented in III-A.

In order to authenticate user A correctly, two requirements have to be satisfied:

1) md(A,X) must be the smallest w.r.t any other md(B,X), where B is another legal user in the system;
2) md(A,X) is smaller than m(A), **or** md(A,X) is closer to m(A) than to any other md(B,X) computed by the system.

The first condition implies that the sample X, coming from the user A, must be the closer sample to A's samples. Moreover, the second requirement states that it has to be sufficiently close to A in order to accept it. This technique has been proposed in [16].

2) User clustering: By using the clustering technique, a user sample is compared only against his/her model. This technique is easier to implement, faster in execution speed, but less precise in terms of accuracy.

To assess that a sample X comes from the user A, the following condition has to be satisfied:

1) md(A, X) must be smaller than m(A).

In this case, no other users' samples are required. The sample must only be sufficiently close to A. This mode of operation should only be used in case there is no other user data to compare with, or when optimization, in terms of time and computation, is required, since only one comparison is required.

C. Privacy-preserving extensions

Privacy-preserving extensions have been investigated and implemented, with the aim of reducing the possibility for *honest but curious* service providers of gathering user behavior. Since the user typing habits are exploited to model and analyze their behavior, the implemented system should avoid the possibility of reconstructing the original typed text to preserve the privacy of the user. The fact that the *n-graph* data type has been adopted to represent the timing features of a typing behavior is already a privacy countermeasure. This is because the *n-graph* does not include the timestamp of when a certain key was pressed. Hence, three privacy-preserving techniques have been implemented:

(i) *Permutation*: keystrokes in a typing sample are shuffled before they are sent to the server component.
(ii) *Substitution*: each typed key is substituted with a close key on the keyboard.
(iii) *Suppression*: only a certain portion of keystrokes is sent to the server.

The effects on the authentication accuracy, when these techniques are applied, are presented in the next evaluation section.

IV. Evaluation

In order to evaluate the solution proposed, three datasets, made of keystrokes collected from real users, have been used to test the system. In particular, the first two were publicly available online, whereas for the third dataset the data has been collected through a deployed web-site. The first dataset[1] was composed of keystrokes taken from 54 users who typed easy and strong passwords on mobile devices. The second one[2] is constituted of samples taken from 20 users that typed on computer keyboards free and transcribed text. The last dataset was made of keystrokes generated by 10 users that transcribed text using smartphones. On all the datasets a first filtering operation has been carried out in order to eliminate keystrokes that could have represented *outliers* for our analyzes. These filters have been used to remove all the keystrokes that contained non printable characters. In addition, all the keystrokes with a time associated above a certain threshold have been discarded as well. The time thresholds used are: 620 ms for both the first and third datasets, and 750 ms the second one. We selected these numbers after a first round of experiments where we tested different values. Hence, we selected the values that produced the best authentication accuracies.

A. Evaluation metrics and model validation

To evaluate the accuracy for the authentication the following metrics have been used:

(i) *False Rejection rate (FRR)*: the number of falsely rejected user attempts against the total number of legitimate attempts.
(ii) *False Acceptance Rate (FAR)*: the number of falsely accepted user attempts against the total number of illegitimate attempts.
(iii) *Equal Error Rate (EER)*: the EER is used to evaluate the overall security of a biometric/behaviometric authentication system. It has to be noticed that the FRR and FAR are negatively correlated, thus the EER is used to measure the correlation between them. Its value can be found by intersecting the graphs of the FRR and FAR. A low value of EER means that the authentication is accurate.

Since there is not a predefined distinction of training and testing data in the datasets used, the *k-fold cross validation method* has been used to validate the results of the experiments. We opted for a 3-fold-cross validation because an high number of keystrokes was required in order to build enough typing samples for the testing phase.

B. Performance evaluation

The results of the experiments performed using the third dataset (freely texting users on smartphones) are shown in this subsection. These results are then compared with the ones obtained using the other two datasets. The two modes of operation of the proposed framework have been tested. The

[1] https://www.ms.sapientia.ro/~manyi/mobikey.html
[2] http://www.cs.cmu.edu/~keystroke/laser-2012

first results for the classification mode are presented in Figure IV-B and Figure IV-B. Figure IV-B shows that by increasing the number of *digraphs* that constitute a typing sample the overall performance increases significantly. The best increase in performance is registered when the typing samples are made of 40-50 *digraphs*, when FAR and FRR are between 15% and 20%.

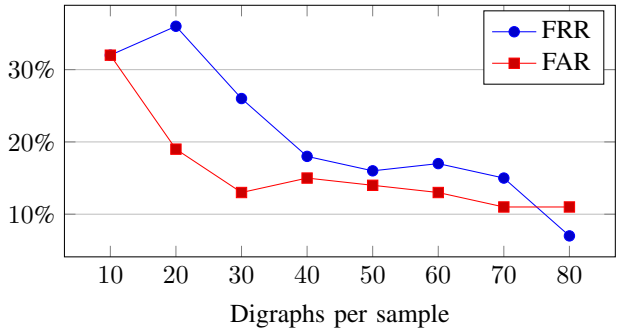


Fig. 1. Impact on authentication accuracy for *user classification*, when varying the number of *digraphs* per typing sample.

For the second experiment, samples of 50 *digraphs* are used to calculate the EER of the system. In order to calculate the EER a threshold parameter has been introduced. These threshold defines how near to the user model a typing sample has to be, in order to recognize it as belonging to that user. By modifying this parameter, different values of FAR and FRR can be obtained. In this second experiment, the EER can be found as the intersection of the FAR and FRR in Figure IV-B. It is equivalent to 16%.

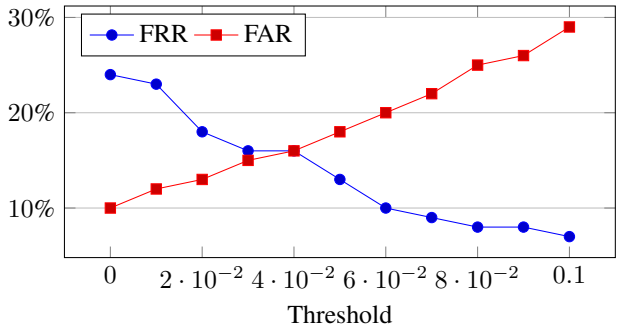


Fig. 2. Impact on authentication accuracy for *user classification*, when varying the threshold parameter.

We performed the same experiments using the clustering mode. The results obtained by changing the number of *digraphs* required for each typing sample show the same evidence that by increasing this number the overall accuracy improves as well. The best improvement was registered when 50 *digraphs* were used. The results are shown in Figure IV-B. We performed the second experiment using 50 *digraphs* per typing sample. Figure IV-B illustrates the results obtained when the threshold parameter is modified. The EER in this case is around 18%.

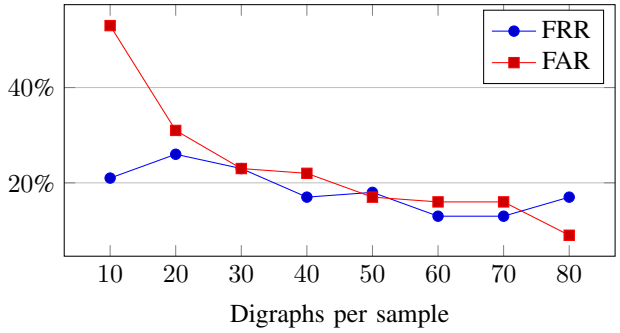


Fig. 3. Impact on authentication accuracy for *user clustering*, when varying the number of *digraphs* per typing sample.

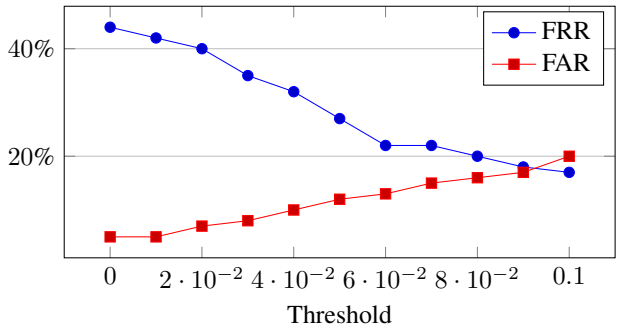


Fig. 4. Impact on authentication accuracy for *user clustering*, when varying the threshold parameter.

The insight of these results is that the when using other user data for the decision making process (for the *classification* method), the authentication accuracy increases. The same was obtained by testing the system with the other two datasets. The comparison of the results is shown in Figure IV-B.

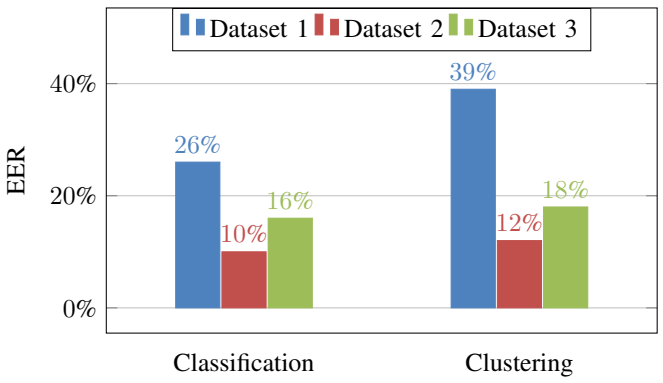


Fig. 5. Comparison of the EER for both *user classification* and *clustering* of the three datasets.

This comparison shows that the techniques used are not really suitable for the analysis of keystroke dynamics for fixed-length strings. The EER registered using the first dataset is 26% and 39% for *classification* and *clustering* respectively.

The experiments conducted using the second dataset, instead, shows that the algorithms used produce a higher accuracy when users type on computer keyboards free text, rather than when they type on smartphones. This is because the typing rhythm on a real keyboard is more personal than the one on a smartphone device. A computer keyboard is larger and each person writes on it using possibly different number of fingers. In addition, on computer keyboards there are neither autocompletions nor suggestions. Nevertheless, the performances are still acceptable in both cases if the keystrokes dynamics framework will be used along with other authentication methods, or perhaps by combining keystroke dynamics with other behaviometrics.

C. Privacy evaluation

The same experiments have been performed applying the privacy-preserving techniques proposed. The EER registered by applying *permutation* and *substitution* are compared to the ones obtained without using them. The results are shown in Figure IV-C.

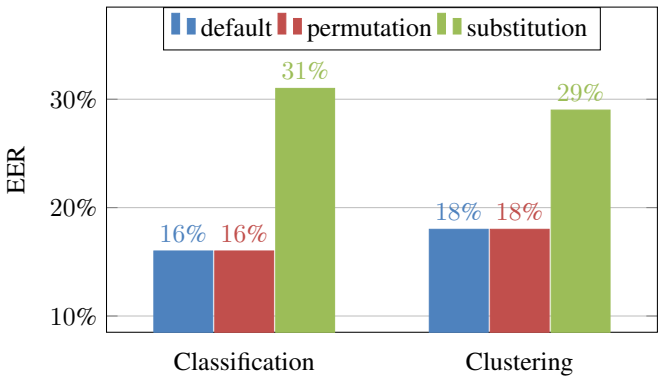


Fig. 6. Comparison of EER for both *user classification* and *clustering* when the *permutation* and *substitution* techniques are applied.

The results show that by applying the *permutation* technique, the EER remains the same. This is because the order of the *digraphs* in a typing sample does not influence the decision making process. Instead, applying the *substitution* technique increases the EER by 15% for *classification* and by 11% for *clustering*.

For what concerns the *suppression* technique there can be two modes of applying this technique. The first one avoids sending more than a certain percentage of keystrokes collected by the client. In this case we would have the same results obtained using the default configuration. The consequence will be that the system will take more time to collect the required number of *digraphs* to perform the decision making operation. Another possible application would be to send a fewer number of keystrokes and build typing samples that are shorter. To observe the impact of this second method on the authentication accuracy the reader can refer to Figure IV-B and IV-B.

D. Validity threats

The main validity threat, when biometric authentication systems have to be tested, is that a lot of data produced by different users is required. The problem is that there is not a specified threshold for the number of this data. In a realistic scenario, the system should be able to manage a large number of users. For this reason, the more data is available in the evaluation phase, the more the results will be accurate and realistic.

For each of the datasets used, the number of data samples per user was sufficiently high. Whereas, more realistic results would have been obtained with more users for each dataset. For the first two dataset, data from about 20 users has been tested. This number can be satisfactory, since other experiments have been performed with these same datasets. However, for the third dataset, the subject of the experiments were only 10. A system with only 10 users registered may not be realistic and, as a consequence, also the results obtained. However, the research of Messerman et al. [3] demonstrated that, after a certain number of users considered in the experiment, increasing this number did not influence their results. The techniques of authentication used in their work were similar to the ones used in the solution proposed. For this reason, we believe that the same reasoning could apply in the case of the third dataset. However, more experiments should be performed to prove that.

Another validity threat is that the data, used for the experiments, has been collected in a controlled environment. This is especially the case for the first and the third dataset. Smartphones could be used while the user is in motion. This will produce possibly different authentication results. To validate the results obtained, more experiments should be carried out in order to test the system in the different contexts in which it could be used.

V. Conclusion

The contributions of this research are (1) a server and client side implementation for keystroke dynamics authentication on mobile, (2) the evaluation of state-of-the-art keystroke dynamics techniques using real datasets, (3) the implementation of privacy extensions on top of existing algorithms. For contribution (1) a prototype of the continuous authentication framework has been developed. The authentication services are exposes by means of RESTful APIs that can be contacted by clients and IAM systems through simply HTTP requests. By adopting this design the clients are untied from the server application architecture, and it leaves room for the extension and integration with other authentication services. Contribution (2) has been achieved by evaluating our solution which makes use of state-of-the-art algorithms proposed for keystroke dynamics analysis. The evaluation has been performed using three different datasets, in particular real data was appositely acquired in order to test the system in its real usage environment (users who type freely on smartphones). The results obtained give us the evidence that our solution analyzes better the typing behavior of users when they type

free text rather than fixed-length strings. They also show that more users have a more particular typing behavior when they type on computer keyboard rather than on smartphones. In the first case the EER of the system was 10% and 12% when using the *user classification* strategy and the *clustering* one respectively. Whereas, higher values of EER has been obtained when users type on smartphones, the EER was 16% using *classification* and 18% using *clustering*. For contribution (3) three privacy-preserving techniques have been presented, which can be used in order prevent *honest but curious* service providers to reconstruct what users type. The impact of using these techniques on the authentication accuracy demonstrated that there is a trade-off between the security of the system and the privacy of the users. These trade-offs have to be regulated based on application requirements.

As a possible future work first of all a more exhaustive evaluation of our solution needs to be performed. This implies that more user data is required in order to validate the authentication accuracy we obtained in our experiments. This also applies for the experiments we performed with the privacy-preserving extensions. It could be also interesting to combine keystroke dynamics with other behaviometrics in order to characterize and verify better user behavior. A possible solution could be to implement another behaviometric authentication framework that can work in parallel with our proposed one. The final authentication decision would then be based on both the results obtained by the two systems, weighted proportionally w.r.t. to their respective accuracy.

REFERENCES

[1] H. Crawford, "Keystroke dynamics: Characteristics and opportunities," in *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*. IEEE, 2010, pp. 205–212.

[2] M. Karnan, M. Akila, and N. Krishnaraj, "Biometric personal authentication using keystroke dynamics: A review," *Applied Soft Computing*, vol. 11, no. 2, pp. 1565–1573, 2011.

[3] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *Biometrics (IJCB), 2011 International Joint Conference on*. IEEE, 2011, pp. 1–8.

[4] P. X. de Oliveira, V. Channarayappa, E. O'Donnel, B. Sinha, A. Vadakkencherry, T. Londhe, U. Gatkal, N. Bakelman, J. V. Monaco, and C. C. Tappert, "Mouse movement biometric system," *Proc. CSIS Research Day*, 2013.

[5] I. Deutschmann, P. Nordström, and L. Nilsson, "Continuous authentication using behavioral biometrics," *IT Professional*, vol. 15, no. 4, pp. 12–15, 2013.

[6] K. Calix, M. Connors, D. Levy, H. Manzar, G. MCabe, and S. Westcott, "Stylometry for e-mail author identification and authentication," *Proceedings of CSIS Research Day, Pace University*, pp. 1048–1054, 2008.

[7] M. Abramson and D. W. Aha, "User authentication from web browsing behavior," DTIC Document, Tech. Rep., 2013.

[8] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *International Journal of Biometrics*, vol. 1, no. 1, pp. 81–113, 2008.

[9] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, "Authentication by keystroke timing: Some preliminary results," DTIC Document, Tech. Rep., 1980.

[10] A. Alzubaidi and J. Kalita, "Authentication of smartphone users using behavioral biometrics," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1998–2026, 2016.

[11] M. Nauman, T. Ali, and A. Rauf, "Using trusted computing for privacy preserving keystroke-based authentication in smartphones," *Telecommunication Systems*, pp. 1–13, 2013.

[12] G. Kambourakis, D. Damopoulos, D. Papamartzivanos, and E. Pavlidakis, "Introducing touchstroke: keystroke-based authentication system for smartphones," *Security and Communication Networks*, 2014.

[13] P. S. Teh, N. Zhang, A. B. J. Teoh, and K. Chen, "A survey on touch dynamics authentication in mobile devices," *Computers & Security*, vol. 59, pp. 210–235, 2016.

[14] C.-J. Tasia, T.-Y. Chang, P.-C. Cheng, and J.-H. Lin, "Two novel biometric features in keystroke dynamics authentication systems for touch screen devices," *Security and Communication Networks*, vol. 7, no. 4, pp. 750–758, 2014.

[15] S. Dhage, P. Kundra, A. Kanchan, and P. Kap, "Mobile authentication using keystroke dynamics," in *Communication, Information & Computing Technology (ICCICT), 2015 International Conference on*. IEEE, 2015, pp. 1–5.

[16] D. Gunetti and C. Picardi, "Keystroke analysis of free text," *ACM Transactions on Information and System Security (TISSEC)*, vol. 8, no. 3, pp. 312–347, 2005.