



HAROKOPIO UNIVERSITY

MASTER THESIS

Keystroke Dynamic Authentication as a Service

Author:

Loukas Avramidis

Supervisors:

Nikolaidou Mara

Rizomiliotis Panagiotis

Tsadimas Anargyros

*A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Computational and Internet
Technologies and Applications*

in the

Informatics and Telematics Department

March 2014

Contents

Contents	i
List of Figures	iii
1 Introduction	1
1.1 Biometric Technologies	1
1.2 An Introduction to Keystroke Dynamics	2
1.3 Contribution	2
1.4 Contents	3
Chapter 1	3
Chapter 2	3
Chapter 3	3
Chapter 4	3
Chapter 5	3
Chapter 6	3
Chapter 7	3
2 Related Work	4
2.1 Implementations	4
2.1.1 Local Authentication	4
Academic Approach	5
Commercial Products	5
2.1.2 Web Authentication	5
Academic Approach	6
Commercial Products	6
2.2 Statement of Problem	6
3 Keystroke Dynamics	8
3.1 Science of Keystroke Dynamics	8
3.2 The Operational Phases of Keystroke Dynamics	9
3.2.1 Raw Data Collection	9
3.2.2 Training and Classification	10
3.3 Types of Errors in Keystroke Dynamics	11
FAR	11
FRR	11

	FMR	12
	FNMR	12
	EER	12
4	Web-based Keystroke Dynamic Authentication Mechanism	14
4.1	Basic Methodology	14
4.2	Client-Server Communication	15
4.3	Classification	17
4.3.1	Euclidean Algorithm	17
4.3.2	Manhattan Algorithm	19
4.3.3	Scaled Manhattan Algorithm	20
5	The Implementation of KeyDy	23
5.1	Programming Environmental and Diagram	23
5.2	Usage	24
5.3	Client Side Functions	25
5.4	Server Side Operations	26
5.5	Database	27
5.6	Web-site Usability and Design	28
5.7	Example Screenshots	29
5.7.1	Documentation website	30
5.7.2	Example's website	33
6	Case Study and Evaluation	38
6.1	Case Study	38
6.2	Testing Environment	40
6.3	Results	42
7	Summary	44
7.1	Conclusion	44
7.2	Food for Thought	45
7.3	Future Work	45

List of Figures

3.1	Latencies between keystrokes when writing the word “password” by three different people. [1]	9
3.2	Some of the most important features of a keystroke sensitive password.	10
3.3	Informations and results from different studies. [2]	10
3.4	Relationship between FAR, FRR, and EER. [3]	13
4.1	Client-server communication. [4]	15
4.2	Detailed client-server communication.	16
4.3	Contours of the decision surfaces for the Euclidean classifier.	18
4.4	Contours of the decision surfaces for the Euclidean classifier. [2]	18
4.5	Contours of the decision surfaces for the Manhattan classifier.	19
4.6	Contours of the decision surfaces for the Manhattan classifier. [2]	20
4.7	Contours of the decision surfaces for the Manhattan classifier.	21
4.8	Contours of the decision surfaces for the Manhattan classifier. [2]	21
5.1	Process Diagram.	27
5.2	Database structure.	28
5.3	Top menu bar, logo and download section.	30
5.4	What, how and some usage information.	30
5.5	Continuation of usage information.	31
5.6	Basic information about KeyDy’s implementation its developer.	32
5.7	Contact form.	32
5.8	Contact form with empty email field error.	33
5.9	Example login screen.	34
5.10	Example login screen while communicating with servers.	34
5.11	Example login screen under training along with the keystroke request content in JSON form.	35
5.12	Example login screen after training is complete.	35
5.13	Example login screen with previous sample accepted by all algorithms.	36
5.14	Example login screen with previous sample denied by all algorithms.	36
5.15	Example login screen with previous sample partially accepted by two out of three algorithms.	36
5.16	Example login screen with training data erased by a click on the training bar.	37
6.1	KeyDy’s script role in a website.	39

6.2	Simple log in form created by the group.	40
6.3	Communication with the servers.	41
6.4	Dialog prompted by unregistered user entry.	41
6.5	Sample denial.	42

Abstract

Information security has always been an issue in the world wide web, which led to countless of researches in order to improve that security. Biometric systems and more specifically keystroke dynamics is the result of some of those researches, a technology which enables us to use typing rhythms and unique typing patterns as the means for authentication and/or identification.

There have been quite a few implementations of keystroke dynamics, but none of the provided as a service in order for website owners to easily integrate into their own authentication forms, without the use of any additional infrastructure or software. All data is parsed in the application's server and send to the websites through requests and responses, communication which is based on the client-server model.

The implementation is open source, in order for developers all around the world to contribute to the project or even make their own versions of it.

Chapter 1

Introduction

In this chapter biometric technologies and keystroke dynamics are introduced while research conducted in the field of keystroke dynamics is briefly reviewed along with the thesis contribution and its basic structure.

1.1 Biometric Technologies

Biometric features are certain qualities and characteristics that a human being possesses. The term biometric derives from the Greek words “βίος” and “μετρικός”, which mean life and measure respectively. Biometrics have been used since the early years of humanity and have provide the ability to recognize and distinguish someone through his characteristic features.

Biometric technologies allow us to distinguish and identify human beings between each other after a certain extraction and analysis of their features. A system that extracts and analyzes features like that is called a biometric system. According to Jain et al [5] , “a biometric system is essentially a pattern recognition system that operates by acquiring biometric data from an individual, extracting a feature set from the acquired data, and comparing this feature set against the template set in the database”.

Needless to say, the importance of biometrics in the field of information security is huge, especially in the form of authentication. Biometric authentication consists of certain phases including data gathering, through any kind of sensors, feature extraction from the collected data, and creation of patterns with the use of certain pattern recognition algorithms.

In conclusion, biometric systems consist of automated methods to authenticate a person based on a physiological or behavioral characteristic, as mentioned by the Biometric Consortium [6].

1.2 An Introduction to Keystroke Dynamics

For the past two decades, computers have played a very important part of our lives and society. With the rapid increase of their use, information security becomes all the more essential. Each day computers are being used to access emails, browse the online markets, play video games, or to manage bank accounts and other sensitive data. Thus, it is understandable that a compromised computer or account could cause a lot of damage, and not only to its owner. That is where the keystroke dynamics step in.

Researches have shown that each person has certain unique features which can be calculated through his keyboard typing rhythm [7]. The delay between each keystroke or the duration that he holds a certain key pressed while typing can distinguish him between other users. It is a behavioral biometric that can be used in many ways such as authentication, identification or even emotion detection, similarly to hand writing.

1.3 Contribution

The basic reason behind this whole experiment is to get people to know more about keystroke dynamics, its multiple uses and advantages. There have been quite a few researches on the subject, but there has never been a keystroke dynamics application used publicly in order for people to get in touch, understand and appreciate the advantages that keystroke dynamics and behavioral biometrics in general could offer to our modern society in terms of security and so much more.

Basically, by igniting the public's interest, companies and institutes will be intrigued to provide sufficient assets and funding in order to further develop these very promising technologies.

1.4 Contents

Chapter 1 Introduction

Biometric systems and keystroke dynamics are introduced, with a brief presentation of related work on the field, the contribution of the application developed, and the contents of the thesis.

Chapter 2 Related Work

A short overview of the research and commercial products currently existing on the field of keystroke dynamic authentication, along with a statement of the problem.

Chapter 3 Keystroke Dynamics

Keystroke dynamics are more thoroughly analyzed in order to readers to understand how exactly this technology works before venturing in application and implementation specifics.

Chapter 4 Web-based Keystroke Dynamic Authentication Mechanism

An introduction to the developed application and the architecture behind, along with a mathematical point of view of the classification algorithms used.

Chapter 5 The Implementation of KeyDy

A more thorough explanation of the implementation and technologies used for the development of the mechanism, which was named KeyDy and guidelines for it to be applied on a website's login form.

Chapter 6 Case Study and Evaluation

A case study performed on the mechanism and the difficulty of its actual implementation on a login form according to the documentation on the website along with the results and user evaluation.

Chapter 7 Epilogue

A short epilogue on the conclusions after the experiment, thoughts of behavioral biometric applications in ubiquitous technologies, and possible future work.

Chapter 2

Related Work

User authentication is probably one of the most important applications of keystroke dynamics. This chapter reviews some local and web application developed either for research or commercial use and the problem statement.

2.1 Implementations

Authentication with keystroke dynamics can be achieved by training a certain algorithm with the typing pattern of a person and excluding all samples that do not meet certain anomaly criteria which will be more thoroughly analyzed in the following chapters. In this section, some of the most famous implementations are reviewed and separated into categories depending on their functionality, local or web, and the scope of their development, academic or commercial. An academic approach suggests that the research has been done by some kind of institute or university, while a commercial product on the other hand suggests an industry or company implementation with profit as their main objective.

2.1.1 Local Authentication

Keystroke dynamic local authentication suggests a locally installed program or some sort of mechanism, with all computations and data storage taking place locally, that might authenticate user logins, text typing etc.

Academic Approach Authentication through keystroke dynamics has mostly been achieved with the help of pattern recognition systems, with the most common of them listed below.

- Statistical Models [8][9][10].
- Neural networks [11].
- Fuzzy logic [12][13]
- Support-vector machines[14]

Commercial Products There is not much knowledge about the commercial products of keystroke dynamic authentication, because they are all closed source. Below is a list referring to some known products, with some of the information known about their implementations.

- TypeWATCH, released by Watchful Software [15], free text typing patterns software.
- Intensity Analytics [16], uses statistical weights and measures
- BioTracker, released by Pluriloc [17], also tracks mouse movements.
- KeyTrac [18], analyzes any text input in the background.

2.1.2 Web Authentication

Web keystroke dynamic authentication on the other hand refers to authentication on websites, web applications etc. The data and the computations might take place either on a remote server which can only be contacted through some kind of connection such as a network or the Internet or locally through an installed program.

Academic Approach Sadly, there have not been as many researches in web authentication as in local authentication. Some of them will be listed below, depending on their pattern recognition approach.

- Statistical Models [7]
- Neural Networks [19]

Commercial Products Just like in local authentication, there is limited knowledge on the mechanics of commercial products and patents. Below is a list referring to some of them, with some of the information provided about their mechanisms.

- Trustable Passwords, released by iMagic Software [20], is used for both web authentication and large-scale enterprise authentication.
- bioChec [21], uses keystroke dynamics for ubiquitous web-based login.
- behavioSec [22], includes keystroke, mouse and environment dynamics.

2.2 Statement of Problem

Password are very easy to be compromised. Brute-force attacks, online scams or just ill placed trust, are all very common security threats that computer users face daily. Indeed, measures have been taken such as "Completely Automated Public Turing test to tell Computers and Humans Apart" also known as CAPTCHA, verification code generators, security questions etc. But they all have downsides, CAPTCHA is able to distinguish people from computers which can counter brute-force and most program-generated attacks, but has been criticized, especially from disabled people who could not understand the letters. Verification code generators can be very impractical and security questions can easily slow down everyday work.

Keystroke dynamics on the other hand do not interfere with the authentication procedure itself. The user just has to sit comfortably, and type a certain passphrase the way he always does, which works as his digital hand signature. FRR can sometimes be a nuisance, but as long as the user accepts his pass-phrase entry not

just as a personal secret knowledge, but also as a behavioral authentication, just like writing his hand signature, those error rates will drop. Let it be noted, that keystroke dynamics can be applicated to any kind of text, not just pass-phrases. Quite a few keystroke dynamic algorithms and mechanisms have been introduced through the past 20 years, but mostly on a research level. As mentioned before, there have been quite a few implementation attempts, but none of them has been used widely while the technology is still far away from becoming a standard. Nevertheless, there has never been an *open source* implementation providing keystroke dynamic authentication as a *RESTful*(Representational State Transfer) web-service [23], available for anyone who wants to use it in order to increase login authentication security on their web-site without any infrastructure or complex commercial products.

According to W3C, a web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [24]. With RESTful style web services, the data is passed directly through HTTP operations(POST, GET, PUT, DELETE) which makes them a lot faster than traditional SOAP(Single Object Access Protocol) web services which, in contrast to the first, use the an additional XML format when sending data or communicating.

Chapter 3

Keystroke Dynamics

Before going into detail about the application developed, the technology of keystroke dynamics needs to be perfectly understood. In this chapter, the science of keystroke dynamics is analyzed, their basic operational phases and the methods used to evaluate an authentication system like that.

3.1 Science of Keystroke Dynamics

As briefly mentioned in the introduction, keystroke dynamics is the technology of harvesting a person's typing rhythm, transforming it into useful information, and using it in some way. The main principle behind this technology is that each person has certain unique features which can be calculated by his keyboard typing rhythm [25]. Those features can tell us many things while also distinguishing him from others, similarly to handwriting. It is a behavioral biometric that can be used in many ways such as to detect emotions, but most importantly, to authenticate or even identify a person.

In the figure above, the delay between each keystroke and the duration that he holds a certain key pressed are displayed, in order to better understand some of the features that a person's typing rhythm consists of.

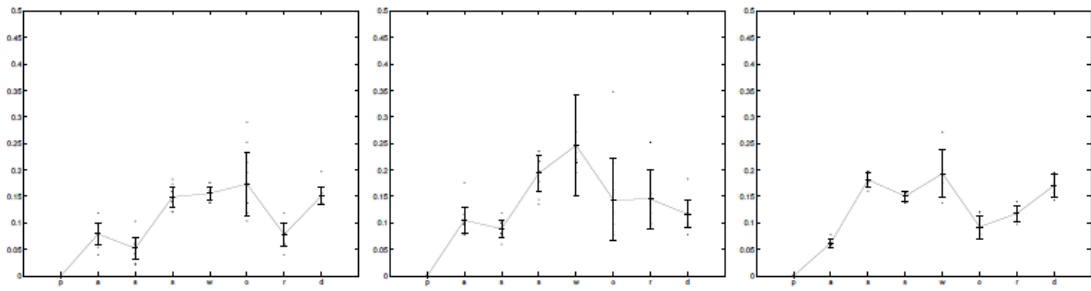


FIGURE 3.1: Latencies between keystrokes when writing the word “password” by three different people. [1]

3.2 The Operational Phases of Keystroke Dynamics

Keystroke dynamics, just like all biometric systems, consist of three operational phases, the data gathering, the training and the classification procedure which are all analyzed in the subsections bellow.

3.2.1 Raw Data Collection

Data collection is the first phase of a keystroke dynamic system. It is essentially the phase where the mechanism collects characteristic data and computes the features of one or more individuals.

More specifically, in keystroke dynamics, data collection refers to the process of saving keystroke timings, such as the press and release time of a user. The data is useless in its raw form, and needs to be transformed into durations between key events, such as the duration between two keystrokes or the duration a key is held pressed. The figure bellow shows two basic features used in keystroke dynamics.

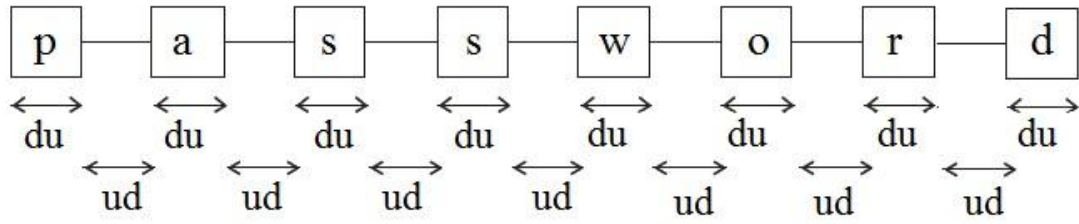


FIGURE 3.2: Some of the most important features of a keystroke sensitive password.

Other features might include the duration of the release of two keystrokes (up-up) etc. This procedure lasts long enough to gather the required amount of samples needed to efficiently train a pattern recognition model.

3.2.2 Training and Classification

The amount of samples needed in order to train a pattern recognition model depends on the model and the approach. The following table, analyzes the results of some researches, along with the number of inserted password repetitions.

Source Study	Classifier	Feature Sets				Password	
		Return key	Down-down	Up-down	Hold	Length	Reps
1 Joyce and Gupta (1990)	Manhattan	✓	✓			N/A	8
2 Bleha et al. (1990)	Euclidean		✓			11-17	30
	Mahalanobis		✓			11-17	30
3 Cho et al. (2000)	Mahalanobis k -NN	✓		✓	✓	7	75-325
	Auto-Associative Neural Net	✓		✓	✓	7	75-325
4 Haider et al. (2000)	Outlier Count		✓			7	15
5 Yu and Cho (2003)	SVM	✓		✓	✓	6-10	75-325
6 Araujo et al. (2004)	Scaled Manhattan		✓	✓	✓	10+	10
7 Kang et al. (2007)	k -Means			✓	✓	7-10	10

Source Study	Filtering		Testing		Results (%)		
	Users	Times	#Attempts	Updating	Threshold	Miss	False Alarm
1 Joyce and Gupta (1990)		✓	1		heuristic	0.25	16.36
2 Bleha et al. (1990)			1	✓	heuristic	2.8	8.1 ^(a)
			1	✓	heuristic	2.8	8.1
3 Cho et al. (2000)	✓	✓	1		zero-miss	0.0	19.5
	✓	✓	1		zero-miss	0.0	1.0
4 Haider et al. (2000)			2		heuristic	19.	11. ^(b)
			2		heuristic	22.	20.
			2		heuristic	13.	2.
5 Yu and Cho (2003)	N/A	N/A	1		zero-miss	0.0	15.78
6 Araujo et al. (2004)			1	✓	heuristic	1.89	1.45
7 Kang et al. (2007)	N/A	N/A	1	✓	equal-error	3.8	3.8

FIGURE 3.3: Informations and results from different studies. [2]

A pattern recognition model calculates the anomaly score of each sample and creates the threshold from those anomaly scores depending on the security level that needs to be achieved.

The classification phase, consists of the evaluation of a newly inserted test sample. The features are extracted, the anomaly score is calculated, and if it is lower than the threshold calculated in the training phase, it gets accepted by the model. If not it is rejected, along with its user. The "update" variable that is seen on the figure above refers to the ability of an algorithm to improve itself. An update mechanism for example, would replace an newly acquired sample which was accepted with the oldest in the training data, and retrain the model to create an updated pattern. A mechanism like that is quite useful because user typing pattern might change in time.

3.3 Types of Errors in Keystroke Dynamics

There are always statistical errors in recognition patterns, thus any algorithm used in keystroke dynamic authentication or identification. Those error rates define the quality of the application of a keystroke dynamic system and a biometric system in general and its effectiveness in distinguishing people from each other. A set of metrics have been defined, in order to calculate the accuracy and the efficiency of biometric system. Those types of errors are listed as follows.

FAR False Acceptance Rate

The false acceptance rate represents the probability of an authentication system providing access to an impostor. The probability of that can be very low in physiological biometrics, but in the case of behavioral biometrics such as keystroke dynamics, it is something very common and it depends on the strictness of the system.

FRR False Rejection Rate

The false rejection rate represents the probability of an authentication system denying access to the legitimate user. Just like in FAR, such a risk is expected to be higher in a behavioral biometrics based system; especially if the person is not in its normal mental or emotional state, such as being tired, distressed, or under

the influence of substances etc. Obviously, it also depends on the strictness of the system.

FMR False Match Rate

The false match rate represents the probability of an identification system having a false positive match for a certain individual. It mostly depends on the algorithms used, but also on the complexity of the biometric features that are being processed.

FNMR False Non-Match Rate

The false non-match rate represents the probability of an identification system not being able to correctly identify an individual, despite it possessing the required biometric data. Just like the false match rate, it strongly depends on the algorithm used, the complexity of the biometric features and the external conditions when data was captured.

EER Equal Error Rate

The strictness of a the pattern recognition system can be changed by the statistical error rates and algorithms that are used, depending on what needs to be achieved, and what is deemed important in an authentication or identification procedure. In the authentication procedure, if the priority is high security, FRR will rise. On the other hand if the system's safety is not of much concern, the system's strictness can be relax by decreasing the FRR. Same goes for the identification procedure. The probability at which both types of errors FAR, FRR and FMR, FNMR respectively are equal, is called the equal error rate.

An important thing that should be mentioned is that FAR, FRR and FMR, FNMR respectively, have inverse relationships. In an authentication system for example, the securer it is, hence the lower the FAR is, the higher the FRR will be as well, and vice versa.

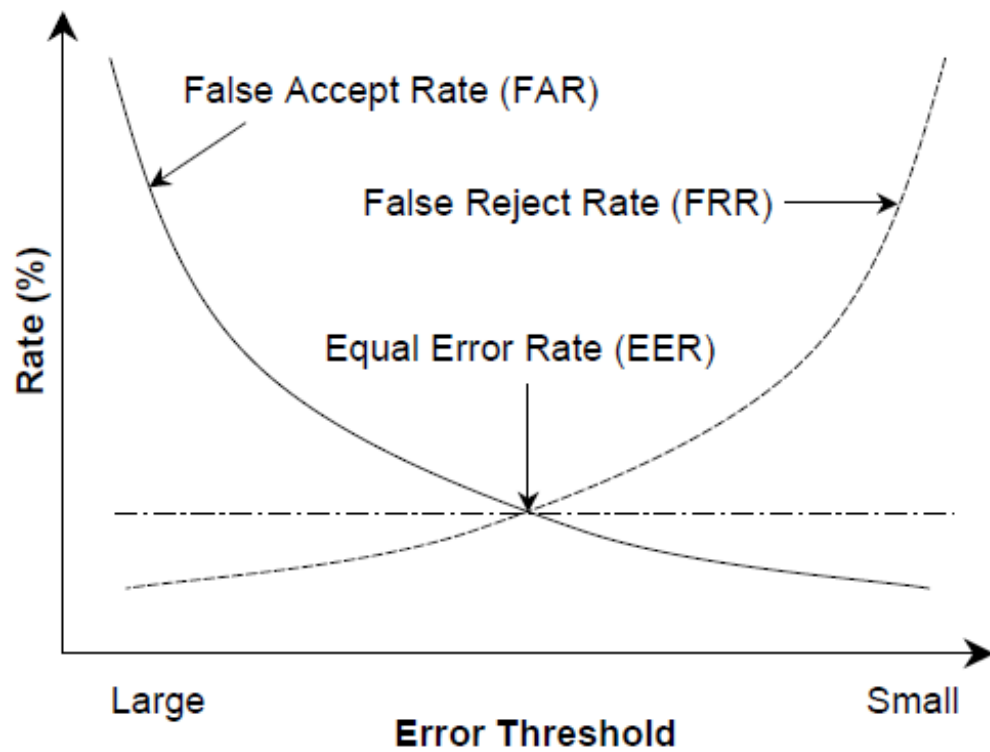


FIGURE 3.4: Relationship between FAR, FRR, and EER. [3]

Chapter 4

Web-based Keystroke Dynamic Authentication Mechanism

In this chapter, the developed web application of keystroke dynamic authentication is introduced, which was named **KeyDy**, along with the reasons behind its creation and a basic methodology.

4.1 Basic Methodology

KeyDy consists of core script that communicates with a set of RESTful web-services in order to train and authenticate users who attempt to login on a website. The script can be included in any web-site with a user-password authentication form.

Whenever a login is attempted, after the user gets verified by the authentication server, a request is send to the web-service to determine if the user wants to use keystroke dynamic authentication. If he accepts, the training begins. After each successful login, a keystroke sample will be send to the web-service and saved in the database until the required amount of samples are collected. Obviously, no password information except the timings of keys pressed is saved. So that means, the times at which each key was pressed and released. Those values have no use at their current form, which is why the server after a set a calculations generates the *down-down*, *down-up*, *up-down* and *up-up* durations, better explained in Figure 3.2. From the point where the required sample amount is available, 3 patterns are created from the metrics above, one for each of the 3 different classification

algorithms used in the mechanism, and each keystroke sample received is compared to those patterns.

If at least 2 out of the 3 algorithms accept a new keystroke sample, the user is verified and the oldest training sample is replaced by the new one. This way, the pattern is updated in case the users typing rhythm changes in time, which is usually the case. Obviously, if the keystroke sample does not meet certain criteria, the user is denied access. The patterns are basically anomaly scores, which will be more thoroughly analyzed in 4.3.

4.2 Client-Server Communication

The communication between the script and the web-server is a basic part of the mechanism, and is based on the client-server model. The web-site that runs the script acts as the client while the web-service as the server as shown bellow.

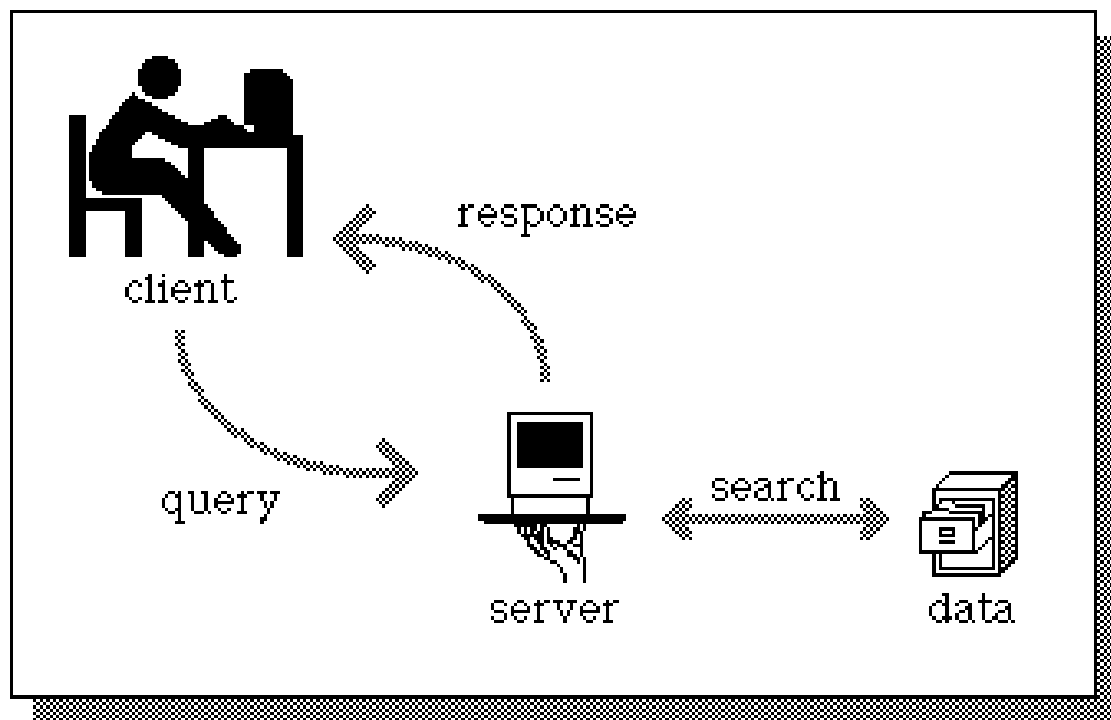


FIGURE 4.1: Client-server communication. [4]

There are 3 basic requests that are prompted in order for the keystroke dynamic authentication to work. The first one determines if the user and password are legitimate, by contacting the web-site's authentication server. If the user gets verified, a request is send a web-service that determines if the user wants to use

keystroke dynamic authentication or has chosen to do so in the past. If so, the third request is send, which contains the keystroke timings of the user to an other web-service which will handle them accordingly, depending on his training status. The figure bellow shows each communication as a different step.

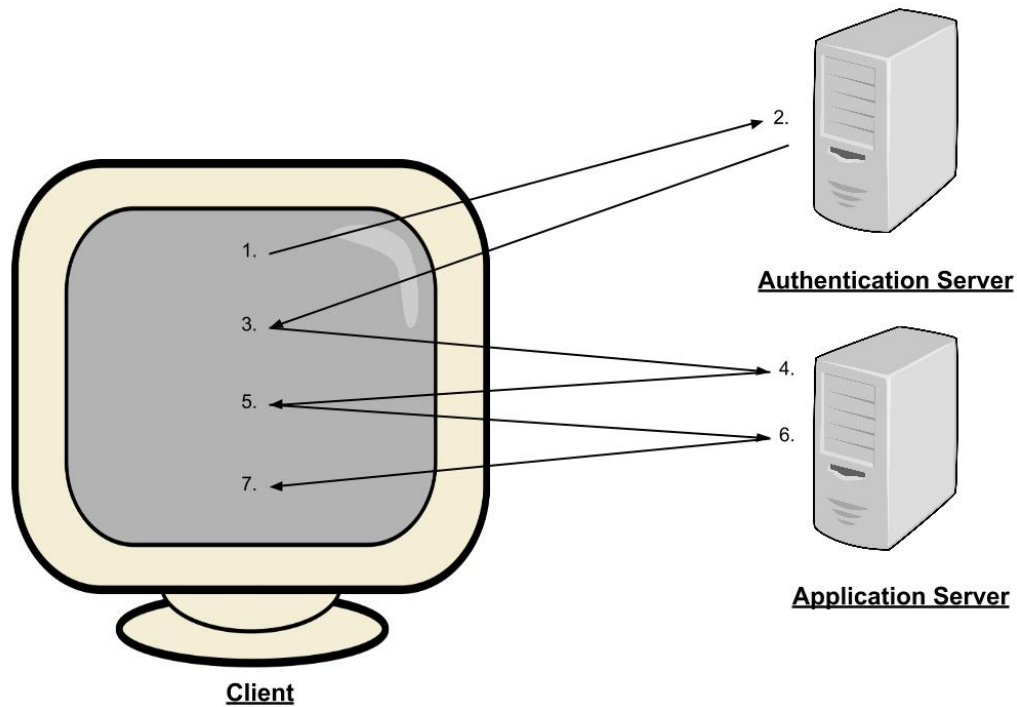


FIGURE 4.2: Detailed client-server communication.

1. Authentication request to web-site server.
2. Authentication response to client.
3. If user verified, send security request to App Server.
4. Check database if user has *KD* enabled and respond.
5. If *KD* enabled, send keystrokes.
6. Receive and handle keystrokes depending on the training progress.
7. Get accepted, denied or continue training.

4.3 Classification

The classification algorithms that are used are based on distance models in order to distinguishing legitimate and illegitimate attempts by representing the samples in a vector space model. The classification begins after all requires samples are collected, with a training phase in which the class's centroid is calculated in a p -dimensional space, where p is the amount of features generated for each sample, such as *down-down*, *down-up* etc. The distances between each training sample and the centroid are then calculated in order to define a threshold that will server as the point at which samples will be accepted or rejected. The other phase of the classification is the testing phase, in which samples inserted in the trained algorithm in order to be compared and evaluated. If the total anomaly score of the distance between a sample and the centroid created in the training process is greater than the threshold, the sample and thus the user, will be denied access. The distances between the centroids and the samples are represented as anomaly scores. The way those thresholds are calculated is explained bellow, depending on each algorithms used.

4.3.1 Euclidean Algorithm

The Euclidean distance algorithm creates a p -dimensional space, with p being the number of features used and each sample being a point in the space created, as mentioned before. Hence, the training data is modeled as a cloud of points, with the mean vector acting as the center of the cloud. A presentation of the vector space is shown bellow.

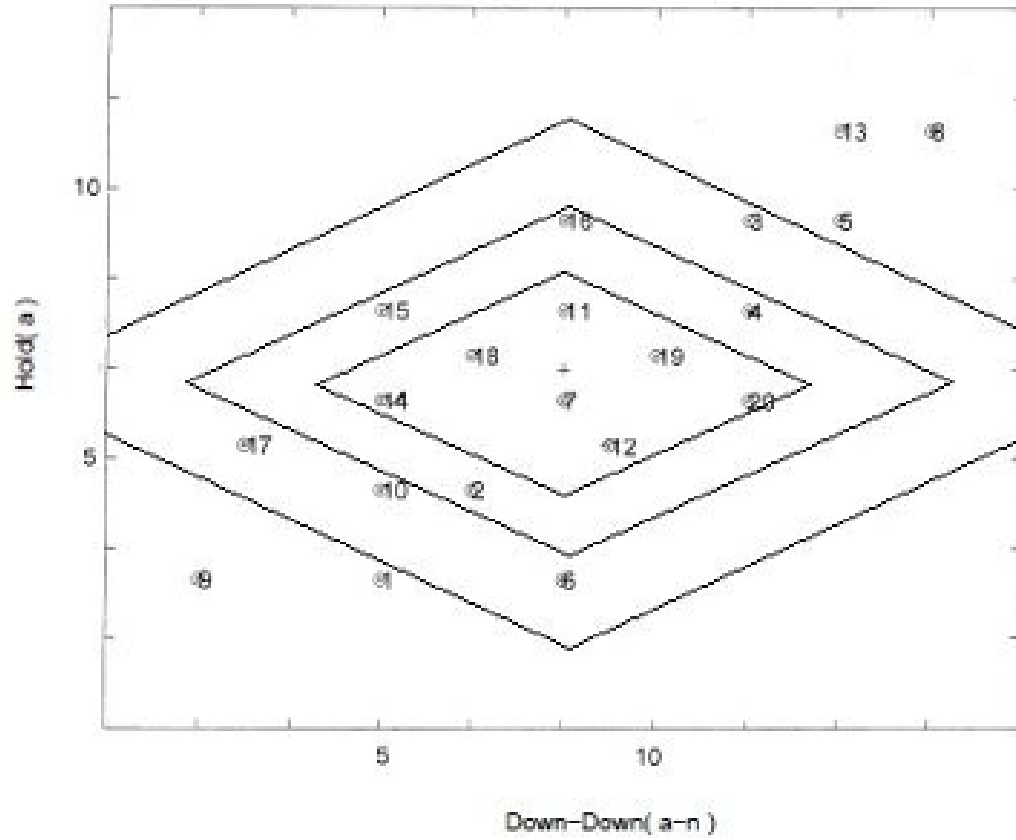


FIGURE 4.3: Contours of the decision surfaces for the Euclidean classifier.

Note that the mechanisms actually uses 4 features, so 4 dimension, but in Figure 4.3, only 2 dimensions are shown, the down-up(hold) and the down-down, in order to better understand the distance model's role.

In the classification phase, the Euclidean distances between the test vectors and the mean vectors are calculated. The sample is accepted or denied, depending on if the mean distances exceeds the mean distance threshold. The x_i value is representing the i -th timing feature of the mean vector and y_i is representing the i -th timing feature of the test vector, the anomaly score is calculated as shown below.

$$\sqrt{\sum_{i=1}^p (x_i - y_i)^2}$$

FIGURE 4.4: Contours of the decision surfaces for the Euclidean classifier. [2]

4.3.2 Manhattan Algorithm

The difference between the Euclidean and the Manhattan classification algorithms is only the way they calculate the distance. The difference is clearly visible in 4.5. In the training phase, the mean vectors of all samples are calculated for each pair, along with the distances between the training data and those mean vectors. [2]

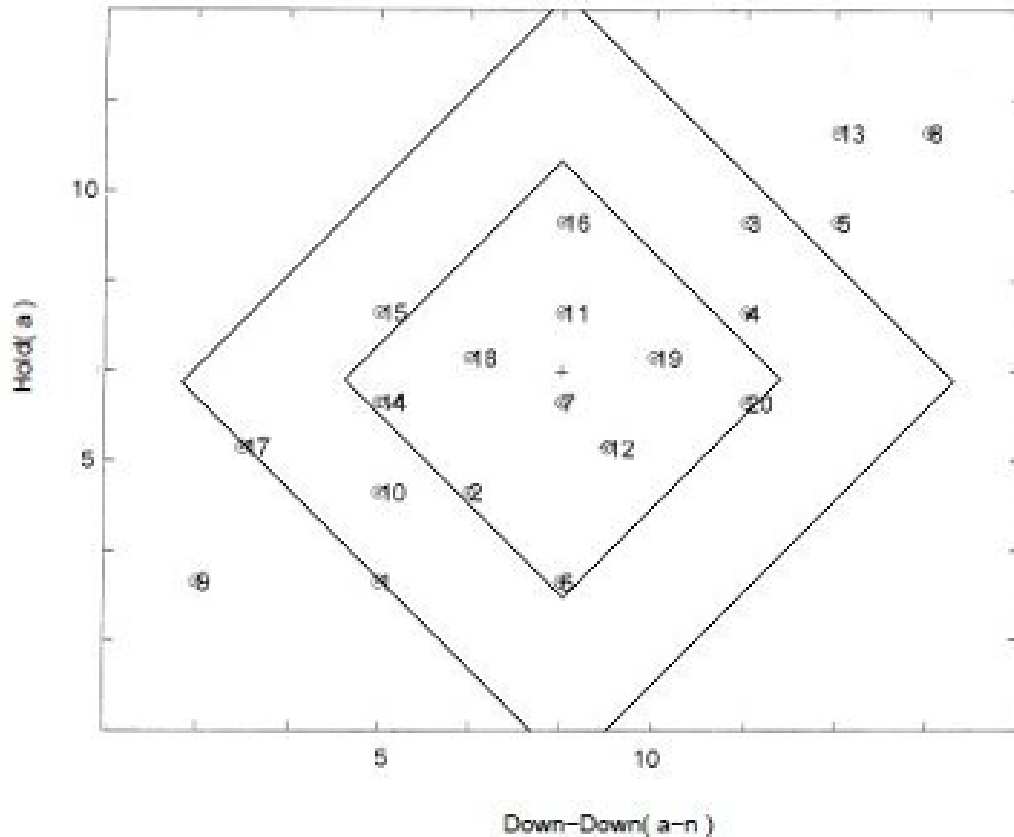


FIGURE 4.5: Contours of the decision surfaces for the Manhattan classifier.

In the classification phase, the Manhattan distance between the mean vectors and the samples are calculated. The difference between the two vectors is calculated across each of the features collected, and the absolute values of these differences are summed, just like the euclidean distance. Again, the x_i value represents the i -th timing feature of the mean vector and y_i represents the i -th timing feature of the test vector. In the figure bellow, you can see the mathematic formula that was explained so far.

$$\sqrt{\sum_{i=1}^p |x_i - y_i|}$$

FIGURE 4.6: Contours of the decision surfaces for the Manhattan classifier. [2]

In Figure 4.5, we see the 2-dimensional space created by the down-up(hold) and the down-down durations, just like in the euclidean, but the difference in the contours of the decision surfaces are clearly different. As with the Euclidean classifier, the closer to the center a point is, the smaller the anomaly score gets. Unlike the Euclidean classifier, points on the left edge of the cloud have relatively low anomaly scores. Because of the diamond shape of the Manhattan contours, it is easy to understand why a sample could have been accepted by the Manhattan algorithm, but not by the Euclidean or the other way around. [2]

4.3.3 Scaled Manhattan Algorithm

The Manhattan distance classifier works just like the two before, but the distance is calculated with a different formula. In the training phase, the mean vectors of all samples are calculated for each pair and the distances between those samples and the mean vectors. The limit contours as shown bellow, are similar to the ones of the Manhattan distance algorithm.

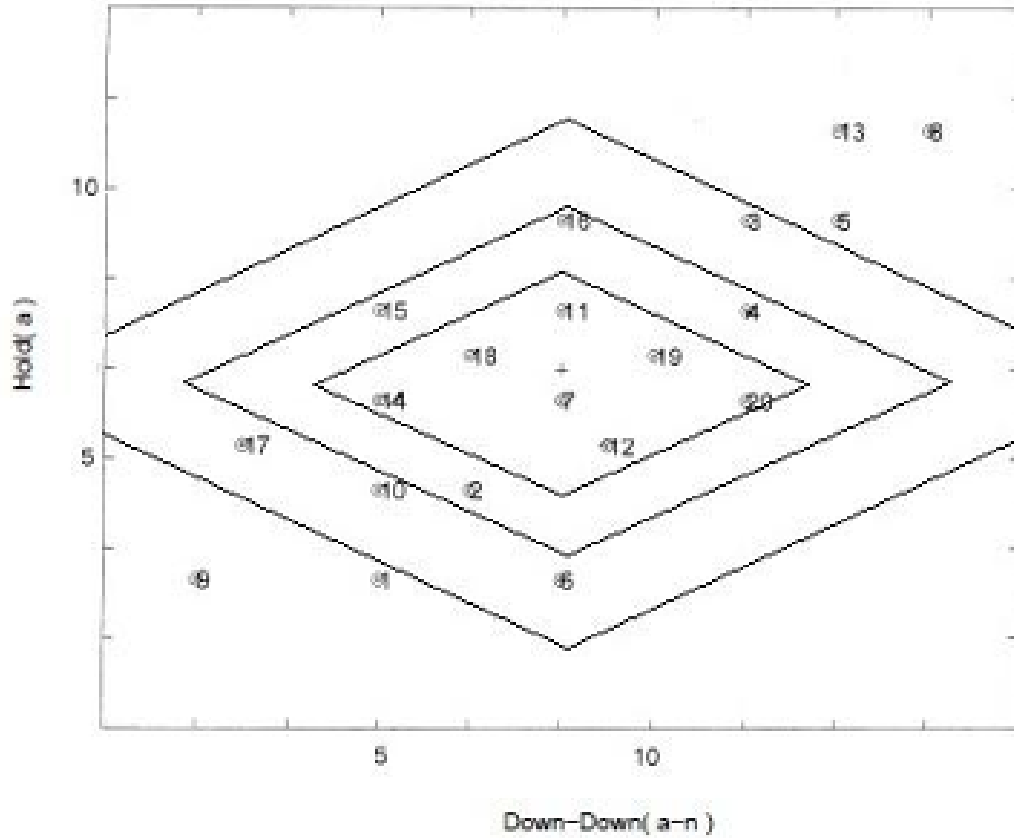


FIGURE 4.7: Contours of the decision surfaces for the Manhattan classifier.

In the classification phase, the Manhattan distances between the mean vectors and the test vectors are calculated. The difference between each two vectors is calculated for each features, and the absolute values of these differences are summed up. Again, the x_i value represents the i -th timing feature of the mean vector and y_i represents the i -th timing feature of the test vector. The anomaly score formula is as follows.

$$\sum_{i=1}^p |x_i - y_i| / s_i$$

FIGURE 4.8: Contours of the decision surfaces for the Manhattan classifier. [2]

In Figure 4.7, we see again the 2-dimension space created by the down-up(hold) and the down-down durations, just like before, with the threshold contours shaping a little differently because of the different distance formula. As with the Euclidean classifier, points in the center of the cloud have low anomaly scores, and points outside of the cloud, in the lower left, have high scores. Unlike the Euclidean

classifier, points on the left edge of the cloud have relatively low anomaly scores. Because of the diamond shape of the Manhattan contours, points that differ from the center in only one dimension are assigned lower anomaly scores than those that differ in multiple dimensions. [2]

Chapter 5

The Implementation of KeyDy

In the following chapter, the implementation of the **KeyDy** application will be thoroughly analyzed, in order to perfectly understand its usage and functionality. A few run-time screen shots will be presented along with the application's website interface.

5.1 Programming Environmental and Diagram

The *client side*, consists of the following:

- **Javascript**, which is most the commonly used dynamic programming language for web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. [26]
- **jQuery**, a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. [27]
- **AJAX** (Asynchronous JavaScript and XML), a group of interrelated web development techniques used on the client-side to create asynchronous web applications. [28]

The *server side*:

- **CodeIgniter**, an open source PHP framework developed by EllisLab, with a very small footprint, a simple and elegant toolkit to create full-featured web applications. [29] It is based on the model-view-controller(MVC) pattern, which divides a given software application into three interconnected parts, in order to separate internal representations of information from the ways that information is presented to or accepted from the user. [30]
- **Phil Sturgeon's CodeIgniter Rest Server**, a fully RESTful server implementation for CodeIgniter using an additional library, configuration file and controller, which makes using RESTful web services a lot more easier with CodeIgniter. [31]
- **MySQL** ("My S-Q-L", Structured Query Language), a relational database management system (RDBMS) [32], used to store all data needed in order for the mechanism to run.

5.2 Usage

In order for someone to implement KeyDy to his web-page, certain steps need to be taken. As mentioned before, the whole client-server communication works around a script which has to be included to the web-page where the authentication is taking place in order to record the keystrokes. But in order for it to work, a few other libraries have to be imported along.

- KeyDy folder, for the keystroke collection, communication and user interface
- jQuery [27] for basic jQuery functionality
- jQuery UI [33] for jQuery User Interface support
- jQuery CSS [33] for the jQuery UI style sheet

After importing the libraries above, the following alterations have to be made in the HTML element classes in order for the functions to work properly.

- The class "username" has to be added to the *username field*.
- The class "keylogger" to the *password field*.

- The class "ready" to the *submit button*.
- The class "keydyForm" and "onsubmit='return false;'" to the *login form*.

5.3 Client Side Functions

The client-side functionality begins with the collection of the keystrokes. When the *password* field is focused, the timings of each key pressed are recorded, which basically means each key press and release time. Action keys are tricky, because they are recorded by the mechanism, while not affecting the password authentication itself, which is why most of them are ignored by the mechanism. Some may have other functionalities, such as the *backspace* key, which empties the password text field and all recorded data so far. That is because a typing mistake would render the whole sample.

When the user hits the login button, all keystroke data is processed and transformed into a JSON(JavaScript Object Notation) [34] format, in order to be sent to the KeyDy server. Before this happens, the username and password have to be verified. An AJAX call is made to the web-site's authentication server, containing the users credentials.

If the credentials get accepted, the next AJAX call is created, which contains the user's username. The request target is a RESTful web service on the KeyDy server, which accesses the database in order to specify the user's security status. If its return value is "neutral", a jQuery dialog window is prompted, asking the user if he wants to use KeyDy's enhanced security or not.

Most importantly, a last request is made containing his keystrokes in JSON format and a security field which can be "secure", "insecure" or "neutral", depending on the user's dialog choice. The response of that request will either be number, or three booleans. A number means that the training is still under process, with the number representing the amount of samples already collected. On the other hand, three boolean would represent the approval or denial of each of the three classification algorithms used.

5.4 Server Side Operations

The server operates through a set of RESTful web services that were created by Phil Sturgeon RESTful server implementation on CodeIgniter, which receive requests and respond accordingly. As mentioned before, the first request received, contains a user's username in order to specify his security options. For this to happen, the database has to be accessed, which is quite easy using the PHP CodeIgniter framework. All data concerning the keystrokes, the users, samples and everything needed by KeyDy is stored in the database. So, a query is sent to the database with the username along with the website's Internet Protocol(IP) so users from different websites can be distinguished, in order to get the security column which as mentioned before can be set to "neutral", "secure" or "insecure" and is sent back to the client as a response. In case the user doesn't exist, a new entry is made and returned, with the security value set to "neutral" as the default. A few moments later, a next request is received by a different RESTful web service, concerning the keystroke authentication itself. It contains the keystroke timings along with the security preference of the user. If it's set on "secure", the database is searched for the user's existing "samples". Before any actions take place, the sample keystrokes contained in the request need to be transformed into the features that we use. Right now, there are two timings for each keystroke, but in order to make any calculations, durations between pairs have to be computed into down-down, down-up, up-down and up-up metrics.

The first condition that needs to be met is that the password contained in the request and the ones saved in the database should be of the same length. If not, the sample is rejected straight away. Most commonly though, they are of the same length, which allows the mechanism to continue the operation. If the samples in the database don't exceed a certain amount dictated by the training options, the sample contained in the request is saved along with the others in the database.

When the samples reach the amount required for the training, they are sent to the classification algorithms in order to calculate the anomaly score of each sample. First of all, the mean vectors of the samples have to be calculated for each feature of every pair. After that, the anomaly scores can be calculated for every pair of each sample. In order to create the threshold, the mean anomaly score of each sample is calculated, by summing up the anomaly score of each pair and dividing them by the number of pairs, in contrast to the previous implementation, where where the one threshold was calculated for each pair. The results showed that

The threshold can now be picked from one of those values, depending on the security that needs to be achieved.

Obviously, we got a threshold for each of the three algorithms(Euclidean, Manhattan and Scaled Manhattan). From that point on, the anomaly score of every new sample received by the server needs to be calculated and compared to the existing thresholds. In the case where at least *two* out of *three* algorithms show positive results, the oldest sample is replaced by the new one. The results are then sent back to the client in the form of three booleans, which concludes the keystroke authentication procedure.

The following figure contains an overall *process diagram*, in order to better understand how all client and server components work with each other:

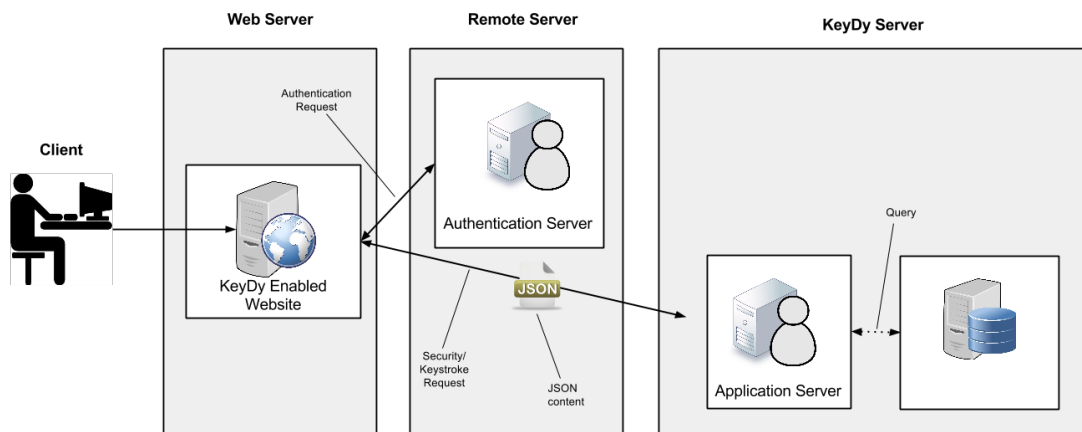


FIGURE 5.1: Process Diagram.

5.5 Database

The database is a pretty important part of the whole service functionality, since it holds all needed information for the keystroke authentication to take place. The database structure is presented in the figure bellow.

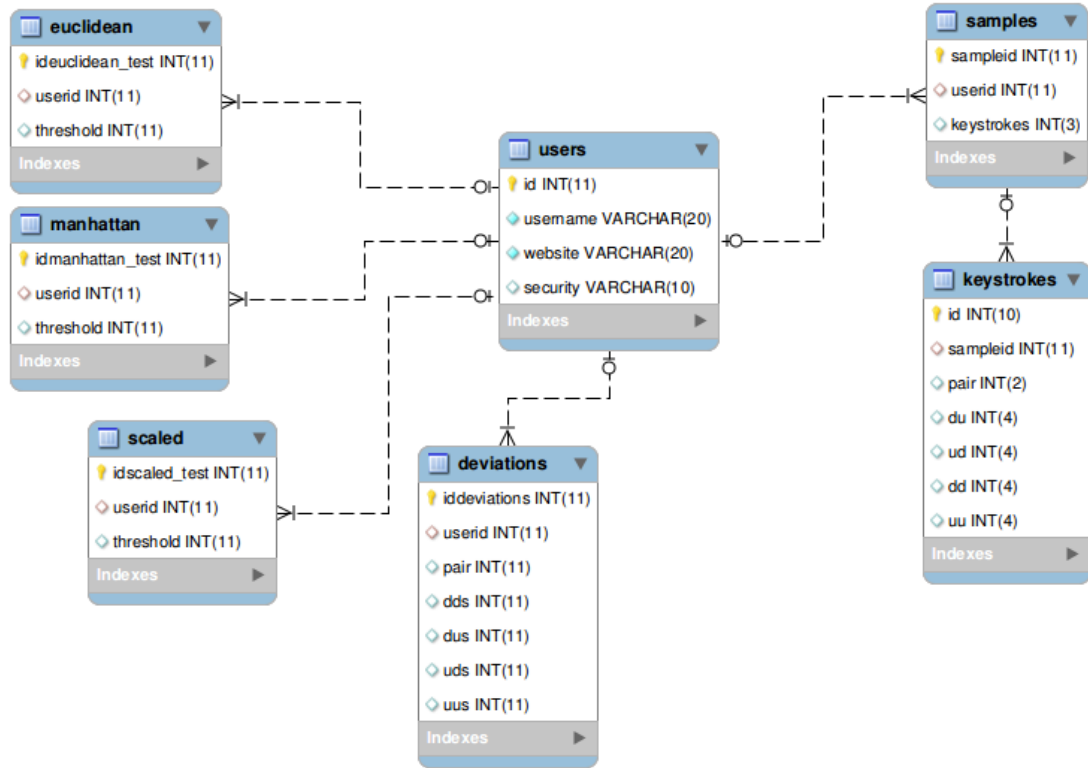


FIGURE 5.2: Database structure.

For each sample, a new entry is made in the "samples" table and one entry for each keystroke pair in the "keystrokes" table, with the "sampleid" working as the foreign key between them. So, naturally, each sample has multiple entries in the "keystrokes" table. There are three tables, one for each classification algorithm containing the threshold previously calculated. The "deviations" is only used in the *scaled Manhattan* classifier.

5.6 Web-site Usability and Design

A web-site [35] was developed containing information about KeyDy's purpose, functionality and an example so users can try out and experience the advantages of keystroke dynamic authentication on their own.

A lot of effort was given in the UI/UX (User Interface/User Experience), in order to make the mechanism more appealing to users.

The website was developed with Zurb's Foundation [36], a *responsive* front-end framework which contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, along with optional

JavaScript extensions. Responsive web design is an approach that aims to provide optimal viewing experience across a wide range of devices such as tablets and smart phones which was introduced by Ethan Marcotte [37].

The website also includes CSS3 features, such as animations, additional Javascript libraries which provide smooth scrolling effects, field validations in order to avoid empty or incorrect field errors, loading panels, pop ups, shadow boxes etc.

The website includes the following content:

- Downloads
- What is KeyDy?
- How does KeyDy work?
- Usage of KeyDy
- About KeyDy
- About Me
- Contact

While the example web-page contains the following:

- Basic Guidelines
- Login Form
- Result Panel
- Keystroke Panel

In the next section, all the above categories are displayed in screenshots.

5.7 Example Screenshots

In the section bellow, screen shots of the website and the example's run-time interface are displayed along with a short explanation for each figure.

5.7.1 Documentation website

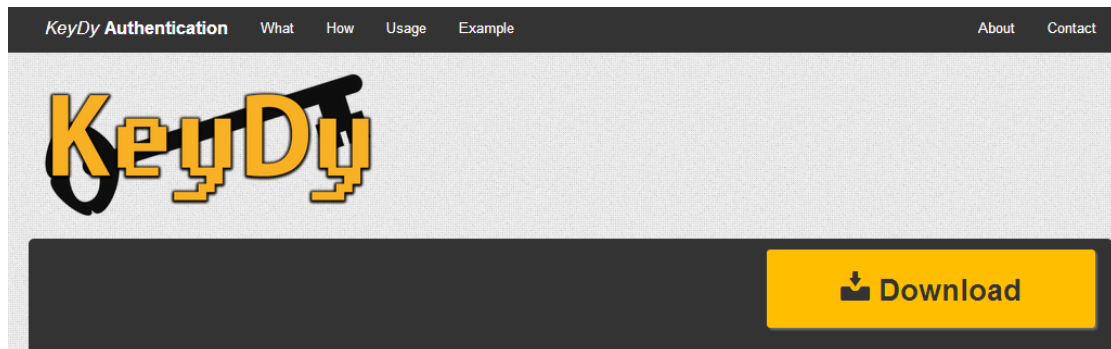


FIGURE 5.3: Top menu bar, logo and download section.

The top of the screen consists of a fixed top bar navigation which follows the screen along when scrolling, a logo and the download button.

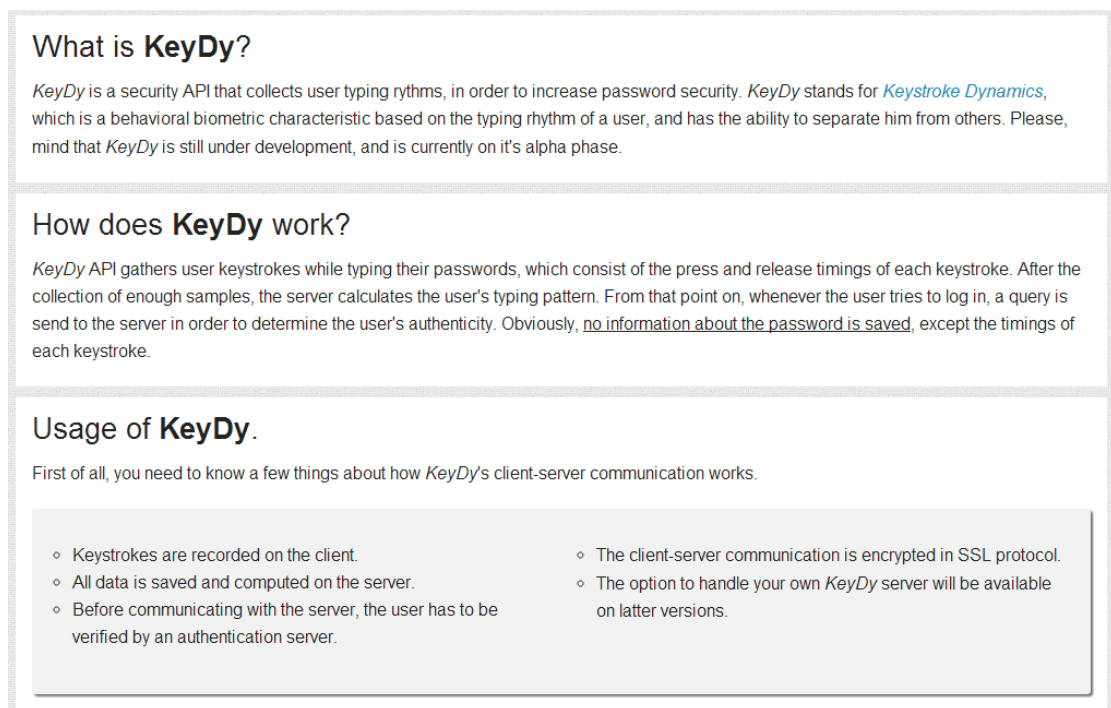


FIGURE 5.4: What, how and some usage information.

Basic information about what KeyDy does, how it works and basic guidelines about its usage.

KeyDy is designed to be as easy as possible to implement, nonetheless, you will need some basic Javascript/jQuery understanding in order to make it work for your website.

1. Download and decompress KeyDy and upload it on your web server. You will need to know the exact location it is saved.
2. Include the following files into your `log in` web page `<head>`:

- Basic jQuery library.

```
<script src="https://code.jquery.com/jquery-1.9.1.js"></script>
```

- jQuery UI.

```
<script src="https://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
```

- jQuery styles.

```
<link rel="stylesheet" href="https://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />
```

- KeyDy client-server communication.

```
<script src="[folder_path]/keydy.js"></script>
```

- jQuery styles.

```
<link rel="stylesheet" href="[folder_path]/overlay.css" />
```

- Insert this segment of code anywhere in your webpage.

```
<div id="dialog-message" title="Security Update" style="display:none;">
<p><span class="ui-icon ui-icon-circle-check" style="float:left; margin:0 7px 50px 0;"></span>
You can now enable <u><a href="https://en.wikipedia.org/wiki/Keystroke_dynamics"
target="_blank">keystroke dynamics</a></u> authentication.</p>
<p>Keystroke dynamics will decrease your account compromisation chance by infusing your typing rhythm
with your password security.</p>
</div>
<div id="overlay"></img></div>
```

3. You will also have to add the following:

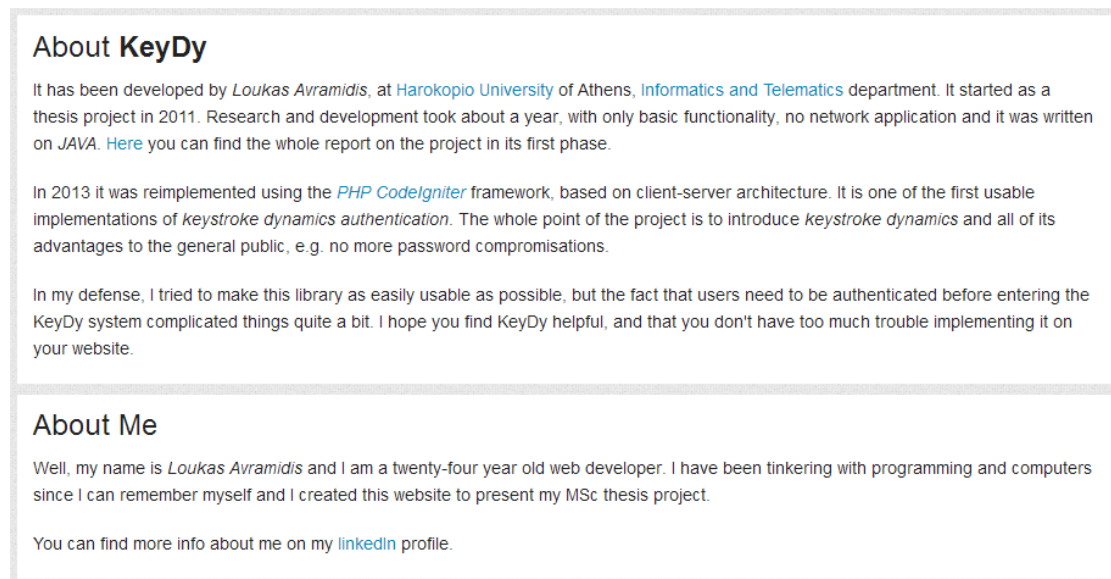
- `class="username"` to the `username` field.
- `class="keylogger"` to the `password` field.
- `class="ready"` to the `submit` button.
- `class="keydyForm"` and `onsubmit="return false;"` to the `log in` form.

4. This part might be tricky. You will have to edit some fields of `keydy.js`, such as the `authServer` variable, depending on the authentication method that your server uses. Currently it is designed to authenticate through an LDAP server. `ldap.php` is included in the compressed folder, which is a script that authenticates via LDAP and returns `true` or `false` depending on the user's originality. If the user isn't correctly verified, the client will not communicate with the KeyDy web server. Don't get too frustrated, this is only a temporary solution and will be changed in the near future. If you meet any problems with this part, don't hesitate to [contact me](#) for further assistance.

5. Also, be mindful about the path of the content inside the KeyDy folder. You will probably have to change the `loader.gif` image element's path in the `HTML` code provided, the include path to `keydy.js` and possibly a few more.

FIGURE 5.5: Continuation of usage information.

Detailed information on how to implement KeyDy on any website containing a login form, as described in 5.2.



About KeyDy

It has been developed by *Loukas Avramidis*, at [Harokopio University of Athens](#), [Informatics and Telematics](#) department. It started as a thesis project in 2011. Research and development took about a year, with only basic functionality, no network application and it was written on JAVA. [Here](#) you can find the whole report on the project in its first phase.

In 2013 it was reimplemented using the [PHP CodeIgniter](#) framework, based on client-server architecture. It is one of the first usable implementations of *keystroke dynamics authentication*. The whole point of the project is to introduce *keystroke dynamics* and all of its advantages to the general public, e.g. no more password compromises.

In my defense, I tried to make this library as easily usable as possible, but the fact that users need to be authenticated before entering the KeyDy system complicated things quite a bit. I hope you find KeyDy helpful, and that you don't have too much trouble implementing it on your website.

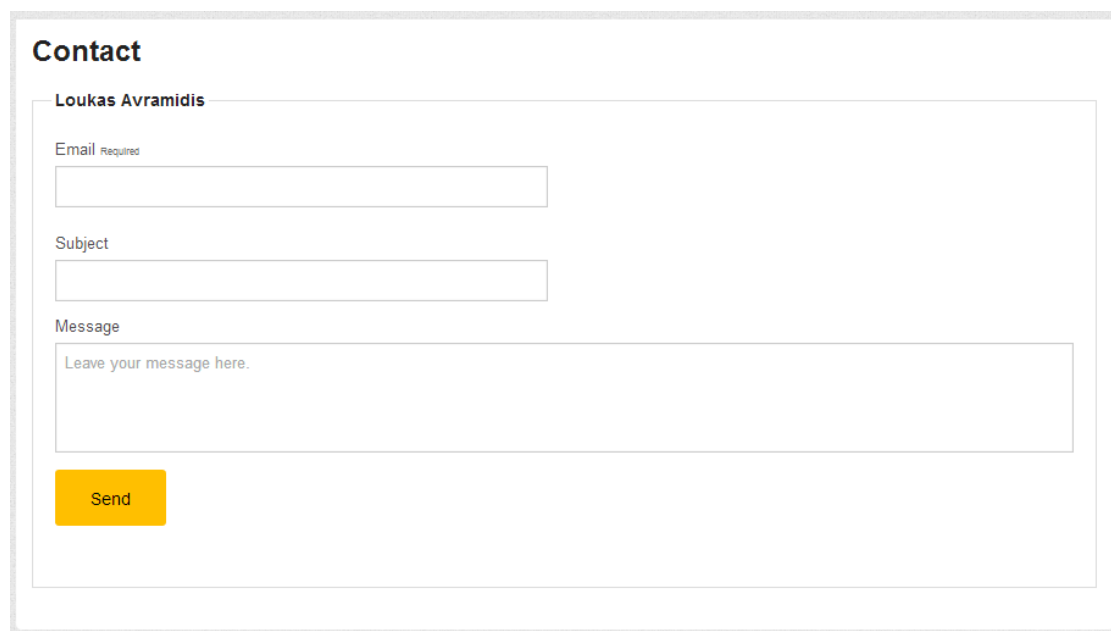
About Me

Well, my name is *Loukas Avramidis* and I am a twenty-four year old web developer. I have been tinkering with programming and computers since I can remember myself and I created this website to present my MSc thesis project.

You can find more info about me on my [linkedin](#) profile.

FIGURE 5.6: Basic information about KeyDy's implementation its developer.

Information about KeyDy's implementation environment and the reasons behind its creation, along with information about the developer.



Contact

Loukas Avramidis

Email Required

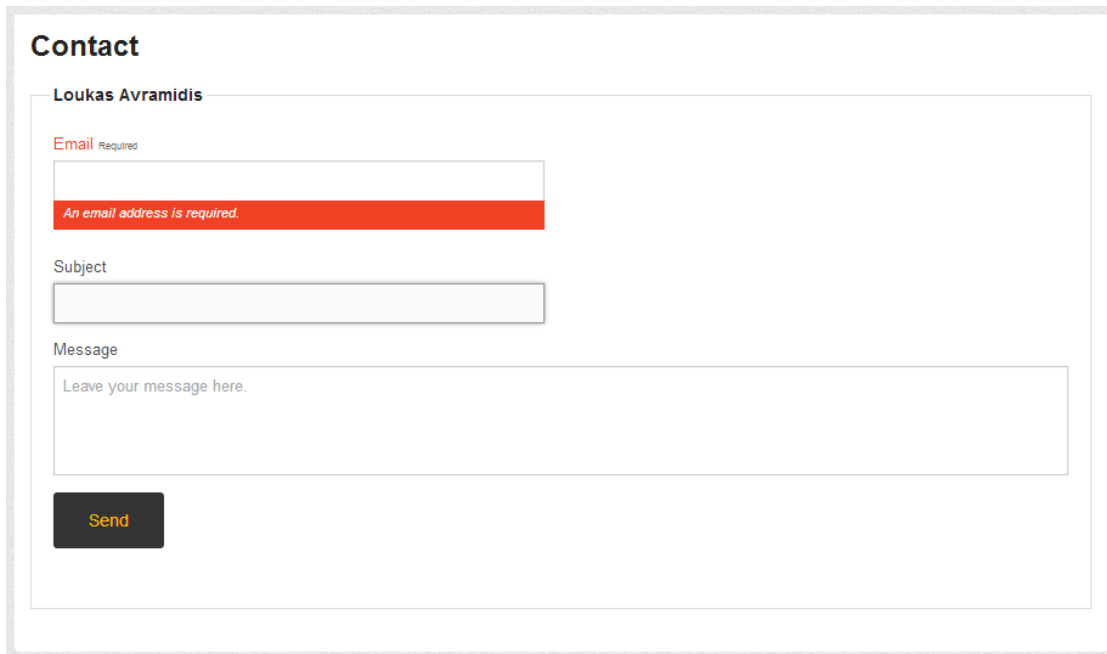
Subject

Message

Leave your message here.

FIGURE 5.7: Contact form.

A basic contact form in order for users interested in KeyDy to contact the developer.



The screenshot shows a web form titled "Contact". At the top, the name "Loukas Avramidis" is displayed. Below it, the "Email" field is labeled "Required" in red. The email input field is empty, and a red error message "An email address is required." is shown below it. The "Subject" field is also empty. The "Message" field contains the placeholder text "Leave your message here.". At the bottom left, there is a dark button labeled "Send".

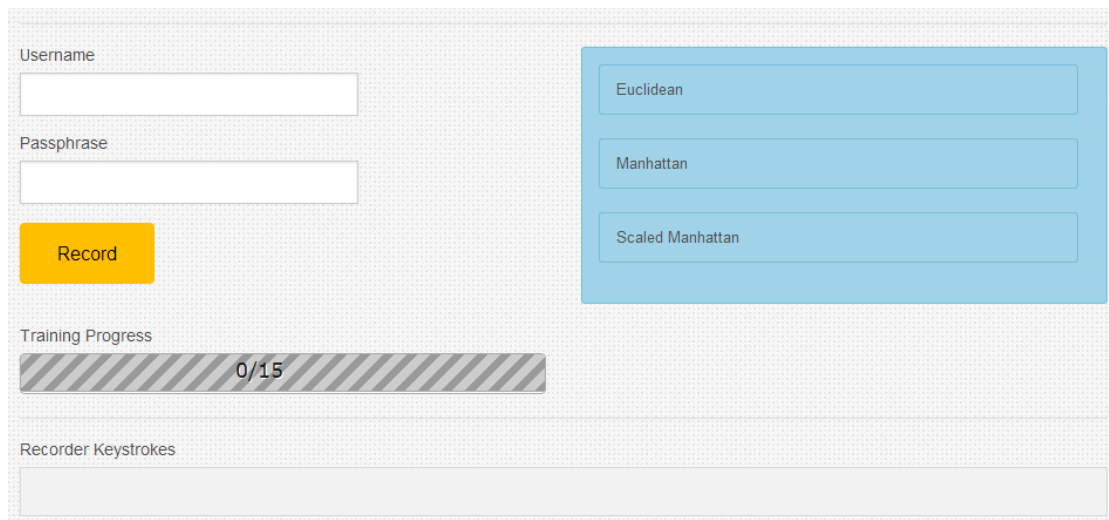
FIGURE 5.8: Contact form with empty email field error.

As mentioned before, the website uses field validation, which pops up if a field is empty or its content is inaccurate, f.e. no correct email format, in which case the buttons are disabled.

5.7.2 Example's website

On the top of the page, a few guidelines are displayed in order to help the user achieve better results and increase the effectiveness of the whole procedure which are the following:

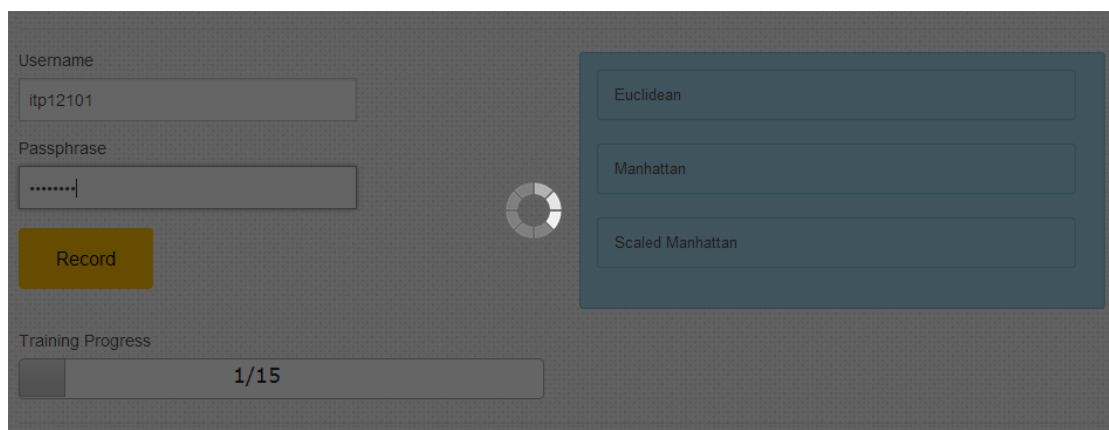
- Sit comfortable and type casually.
- Taking a small break between entries will improve the overall accuracy.
- If you make a mistake hit backspace, it will clear the field and data.
- Click on the progress bar to retrain your data.



The login screen features a light gray background. On the left, there is a form with two input fields: 'Username' and 'Passphrase'. Below the 'Passphrase' field is a yellow 'Record' button. To the right of the form is a blue box containing three buttons: 'Euclidean', 'Manhattan', and 'Scaled Manhattan'. Below the 'Record' button is a 'Training Progress' section with a gray progress bar showing '0/15'. At the bottom, there is a 'Recorder Keystrokes' section with a large, empty gray rectangular area.

FIGURE 5.9: Example login screen.

The login form is on the left side of the screen, along with the submit button and a progress bar showing the amount of samples so far, and the total progress. The required sample amount is set to 15, for a quick training procedure. On the right part of the screen, the algorithms used are displayed and on the lowest part, the recorded keystrokes after each attempt.



This screenshot shows the login screen during server communication. The 'Username' field contains 'itp12101' and the 'Passphrase' field is masked with dots. The yellow 'Record' button is still visible. The 'Training Progress' bar now shows '1/15'. A central loading icon, consisting of a circle with eight segments, is displayed. The blue box on the right with the algorithm buttons remains visible. The 'Recorder Keystrokes' area at the bottom is not visible in this view.

FIGURE 5.10: Example login screen while communicating with servers.

The communication with the servers takes a few seconds, disabling the content of the web-page and prompting the loading icon on an semitransparent overlay layer.

Username
itp12101

Passphrase

Record

Training Progress
5/15

Recorder Keystrokes

```
{ "username": "itp12101", "keys": [{ "downkey": "1392561675902", "upkey": "1392561675992" },
{ "downkey": "1392561676093", "upkey": "1392561676178" },
{ "downkey": "1392561676387", "upkey": "1392561676461" },
{ "downkey": "1392561676569", "upkey": "1392561676632" },
{ "downkey": "1392561676643", "upkey": "1392561676733" },
{ "downkey": "1392561676822", "upkey": "1392561676890" },
{ "downkey": "1392561676963", "upkey": "1392561677058" },
{ "downkey": "1392561677141", "upkey": "1392561677221" } ], "security": "secure" }
```

FIGURE 5.11: Example login screen under training along with the keystroke request content in JSON form.

On the lowest part of the screen, the keystroke data sent to the server after each attempt is displayed in JSON form while the progress bar advances.

Username
itp12101

Passphrase

Record

Training Progress
Training Complete!

FIGURE 5.12: Example login screen after training is complete.

After 15 inserts, the training is completed, the progress bar's label changes into "Complete" and the classification phase begins.

A login interface with a light gray background. On the left, there are two input fields: 'Username' containing 'itp12101' and 'Passphrase' which is empty. Below these is a yellow 'Record' button. On the right, there is a green rectangular panel containing three sub-panels, each with a label: 'Euclidean', 'Manhattan', and 'Scaled Manhattan'. All three sub-panels are green, indicating acceptance.

FIGURE 5.13: Example login screen with previous sample accepted by all algorithms.

After the training is complete, each new sample is classified and compared to the pattern. The last attempt got accepted by all three algorithms. If two out of the three algorithms accept the sample, it is considered valid. Depending on a samples validity, the color of the classifier fields change, green is for positive and red for negative results.

A login interface similar to Figure 5.13. The 'Username' field contains 'itp12101' and the 'Passphrase' field is empty. The yellow 'Record' button is present. On the right, the green panel is now red. It contains three sub-panels labeled 'Euclidean', 'Manhattan', and 'Scaled Manhattan'. All three sub-panels are red, indicating denial.

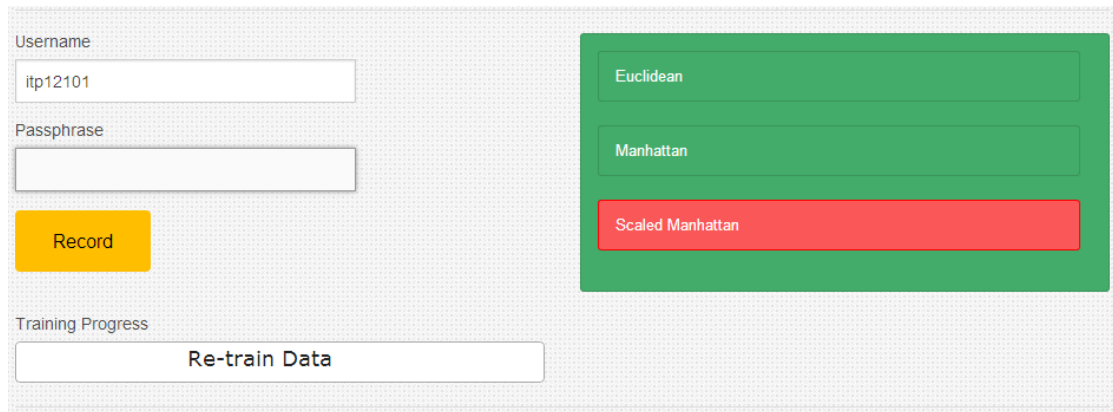
FIGURE 5.14: Example login screen with previous sample denied by all algorithms.

On this attempt, all algorithms denied the sample, because its anomaly score exceeded the threshold for each algorithm.

A login interface similar to Figure 5.13. The 'Username' field contains 'itp12101' and the 'Passphrase' field is empty. The yellow 'Record' button is present. On the right, the green panel contains three sub-panels labeled 'Euclidean', 'Manhattan', and 'Scaled Manhattan'. The 'Euclidean' and 'Manhattan' sub-panels are green, while the 'Scaled Manhattan' sub-panel is red, indicating partial acceptance.

FIGURE 5.15: Example login screen with previous sample partially accepted by two out of three algorithms.

As mentioned before, there is always a chance of a partial acceptance of a sample. In a case such as in the figure above, one of the three algorithms did not accept the sample, but it is still considered valid since the other two accepted it.



The screenshot shows a login interface with the following elements:

- Username:** A text input field containing the text "itp12101".
- Passphrase:** An empty text input field.
- Record:** A yellow button.
- Training Progress:** A section containing a button labeled "Re-train Data".
- Algorithm Results:** A green panel on the right containing three buttons:
 - Euclidean:** A green button.
 - Manhattan:** A green button.
 - Scaled Manhattan:** A red button.

FIGURE 5.16: Example login screen with training data erased by a click on the training bar.

On the figure above, the saved samples were erased after a click on the progress bar, in case someone wants to retrain the system with a different password.

Chapter 6

Case Study and Evaluation

In order to evaluate the mechanism in terms of simplicity to implement and function properly, a case study was made which is explained in the chapter along with its results.

6.1 Case Study

As mentioned mentioned before, a case study took place in order to evaluate the mechanism and its simplicity to use. More specifically, a few groups groups were selected at random and were asked to implement the libraries on their websites by following the guidelines uploaded on KeyDy's documentation website [35]. In the picture bellow, the way data is passed and the communication between the login website, KeyDy's script and the servers in order to better understand its role in a foreign website.

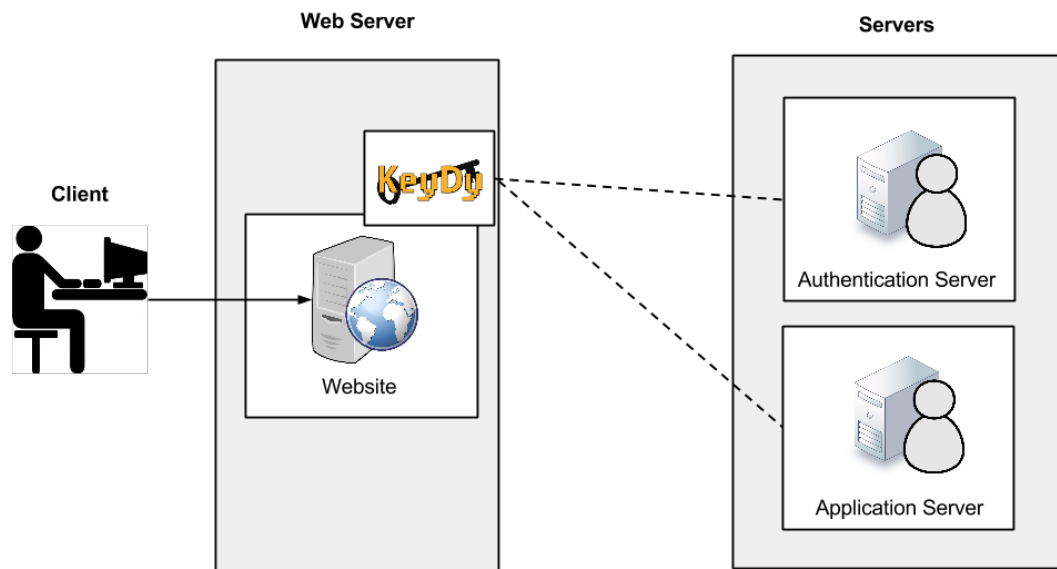


FIGURE 6.1: KeyDy's script role in a website.

In the first case, a simple HTML website was developed with a log in form and the proper libraries included. An error prompted while trying to communicate to the server which turned out to be caused by the same-origin origin policy. This is an important security concept for browser side programming languages such as Javascript, which is used for the scripts. The policy permits scripts running on pages originating from the same site to access each other's DOM with no specific restrictions, but prevents access to DOM on different sites [38]. In order to avoid this security policy, cross-origin resource sharing(CORS) was used. CORS is a mechanism that allows JavaScript on a web page to make XMLHttpRequests to another domain, not the domain the JavaScript originated from [39]. By adding a custom header with a few extra options, the browser and the server exchanged data through the AJAX calls without any conflicts and the implementation functioned correctly. In the second case a new error popped up, which yet again declined every single response from and to the server. After some research and debugging, it turned out that the problem was the SSL certificate that the KeyDy server used. If a browser had priorly verified the certificate the communication worked correctly, but in the case were browser accessed the server for the first time, communication was denied and the security and keystroke requests never reached the KeyDy

server. Mind that the authentication request that did not use HTTPS worked perfectly on both cases. So in order to solve this problem, the user needs to be prompted with the KeyDy certificate and verify it so that the communication can work as it was meant to be.

6.2 Testing Environment

In this section a few screen shots are displayed from the developed implementation by one of the groups, in order to get an idea about how KeyDy would look on a foreign website.

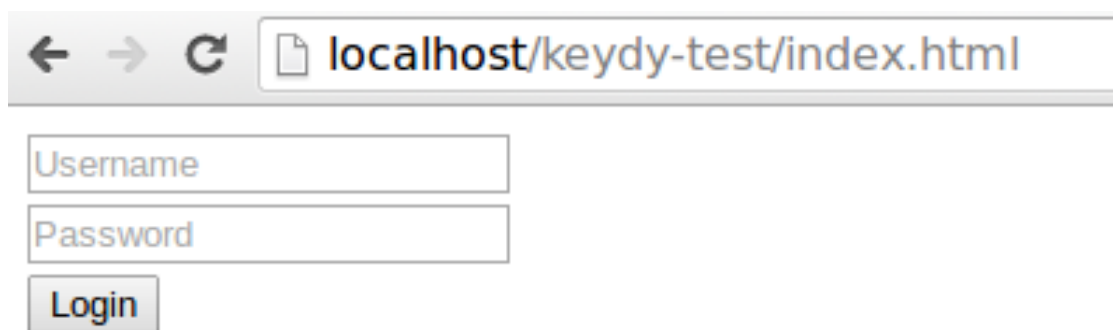


FIGURE 6.2: Simple log in form created by the group.

In the picture above, the simple log in web page is shown, that was created by one of the groups that partook in the case study.

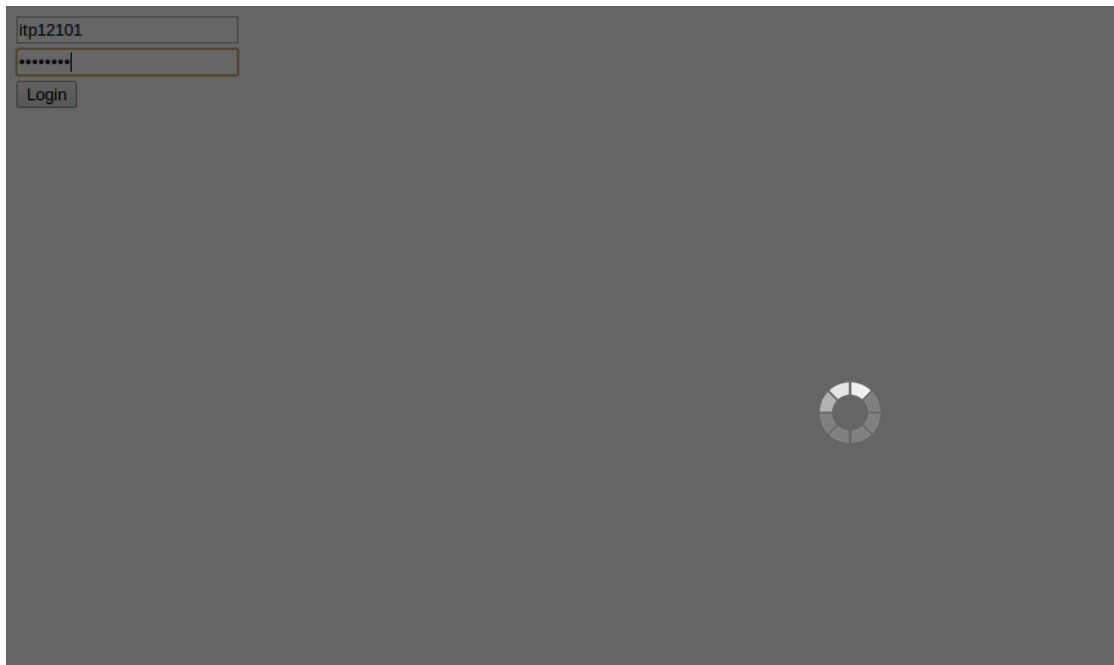


FIGURE 6.3: Communication with the servers.

The figure above displays the loading sign while the web page communicates with the servers.

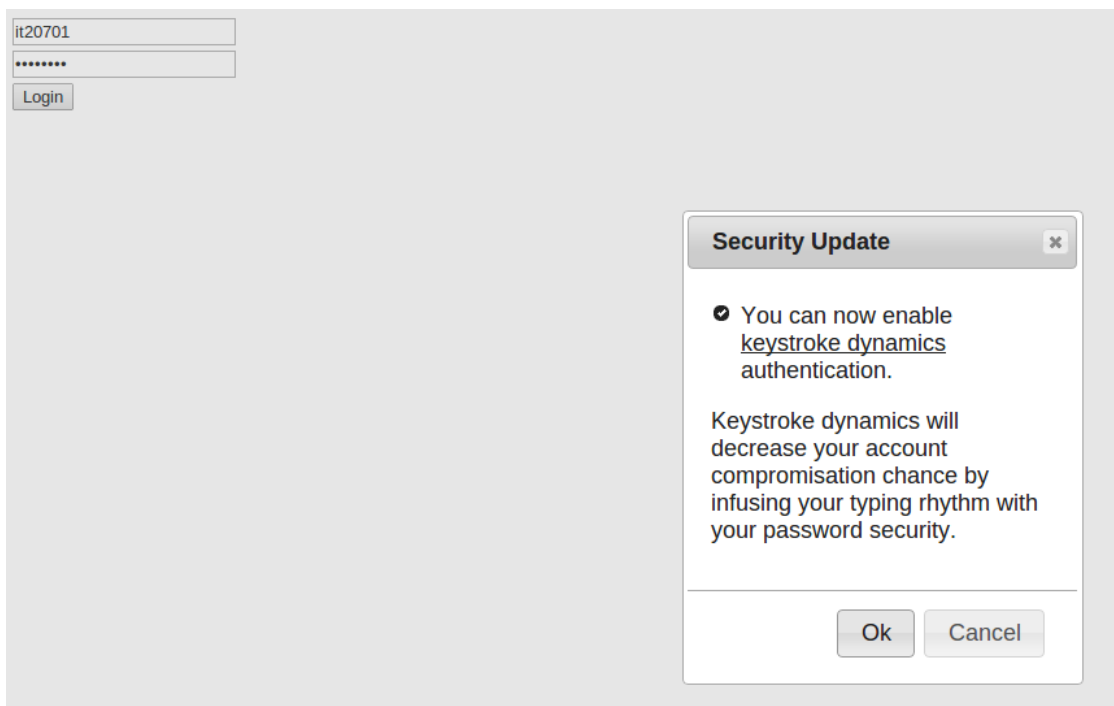


FIGURE 6.4: Dialog prompted by unregistered user entry.

In the case where an new and unregistered in KeyDy user enters the system, this dialog is prompted as usual.

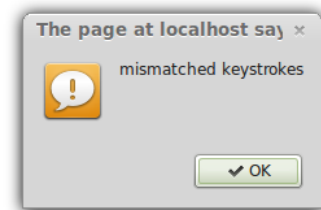
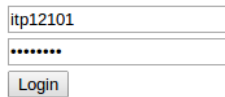


FIGURE 6.5: Sample denial.

In the case of denial, an alert is prompted by the script provided. On the other hand, if the sample is accepted by the algorithms, the procedure goes on, and the user logs in normally. Same goes for the training phase, the user is allowed access until the required amount of samples are collected.

6.3 Results

In conclusion, the keystroke authentication implementation of KeyDy is very easy to implement on any kind of website and browser, as long as the appropriate certificate is verified, which was actually the basic concern behind the implementation. Users were able to follow the guidelines on the website documentation without any further instruction from the developer, except the communication errors that showed up. Obviously a few alterations were made on the documentation in order to be more easily understood, after a short feedback from the two groups. The major problem of the implementation as it is, is the authentication request. The groups that implemented the mechanism on their website both used LDAP authentication servers, which made it a lot more easier for them. If someone does

not use LDAP, it might prove a lot more difficult to implement. That is the main concern, and will be the first thing to be changed in the near future updates.

Chapter 7

Summary

In this last chapter, a conclusion is made about everything that happened through the past chapters, while the reader is also provided with some food for thought in order to understand the advantages that keystroke dynamics could provide, along with future upgrades and ideas for the implementation.

7.1 Conclusion

Keystrokes dynamic authentication is an underrated authentication mechanism, which never received the proper media attention that it needs in order to flourish. "As a service" was a basic concept behind the implementation, so that users and websites have the opportunity to access these technologies without any additional infrastructure or costs, in hopes of it being more accessible to the wider public. A web service implies less to none computation being made on the web-server or client, which makes KeyDy an all around light and easy to use application of keystroke dynamic authentication for any kind of website.

KeyDy is also open source as mentioned before, which will attract developers to improve or even make their own versions of KeyDy hopefully better, which will also help the spread of keystroke dynamics technologies which can provide so much to our modern society.

7.2 Food for Thought

Keyboards dynamics are only the beginning. The same principles could easily be applied on smart-phones, video-game consoles and basically every electronic device or gadget that involves human-computer interaction, for example a car. There are definitely unique patterns in the way people drive their cars, so a mechanism like that would exclude the possibility of car thefts. To be more precise, there are unique patterns almost in every human-computer interaction, which could authenticate a person.

Yes, there are other ways of biometric authentication, but people are not usually happy to provide their characteristics to sensors directly. Behavioral biometrics could solve that, through simple "transparent" sensors which would not affect their everyday routine in any way, in contrast to using a password for example. The development of technologies like that could easily be combined with ubiquitous technologies, smart homes etc.

7.3 Future Work

The next step of considering the application would be to introduce it to the wider public through social networks. A security application that would recognize your personal typing rhythm and deny others from using your account in any way, preventing them to send messages, comments etc. even if someone happened to find an account logged in.

As mentioned before, the application was implemented based on a previously created mechanism, that has big room for improvement. If an application like that would to become commercial, improvement is necessary, in order to achieve acceptable standards in terms of lower error rates.

Bibliography

- [1] J. Ilonen. Keystroke dynamics, 1980.
- [2] Kevin S. Killourhy, Daniel P. Siewiorek, and Christos Faloutsos. A scientific understanding of keystroke dynamics, 2012.
- [3] N. L. Clarke and S. M. Furnell. Advanced user authentication for mobile devices. *Computers & Security*, 26(2):109–119, March 2007. doi: 10.1016/j.cose.2006.08.008. URL <http://dx.doi.org/10.1016/j.cose.2006.08.008>.
- [4] Dheeraj Sanghi. Computer networks. URL <http://www.cse.iitk.ac.in/users/dheeraj/cs425/lec17.html>.
- [5] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Trans. on Circuits and Systems for Video Technology*, 14: 4–20, 2004.
- [6] Introduction to biometrics, September 2012. URL <http://www.biometrics.org/introduction.php>.
- [7] Cheng huang Jiang. Keystroke statistical learning model for web authentication.
- [8] G.E. Forsen, M.R. Nelson, R.J. Staron, PATTERN ANALYSIS, and RECOGNITION CORP ROME N Y. *Personal Attributes Authentication Techniques*. Defense Technical Information Center, 1977. URL <http://books.google.gr/books?id=tbs40AAACAAJ>.
- [9] Rick Joyce and Gopal Gupta. Identity authentication based on keystroke latencies. *Commun. ACM*, 33(2):168–176, 1990. ISSN 0001-0782. URL <http://dx.doi.org/10.1145/75577.75582>.

- [10] L.C.F. Araujo, L.H.R. Sucupira, Jr., M.G. Lizarraga, L.L. Ling, and J.B.T. Yabu-Uti. User authentication through typing biometrics features. *Trans. Sig. Proc.*, 53(2):851–855, February 2005. ISSN 1053-587X. doi: 10.1109/TSP.2004.839903(410)53. URL [http://dx.doi.org/10.1109/TSP.2004.839903\(410\)53](http://dx.doi.org/10.1109/TSP.2004.839903(410)53).
- [11] M. S. Obaidat and B. Sadoun. Verification of computer users using keystroke dynamics. *Trans. Sys. Man Cyber. Part B*, 27(2):261–269, 1997. ISSN 1083-4419. doi: 10.1109/3477.558812. URL <http://dx.doi.org/10.1109/3477.558812>.
- [12] Bassam Hussien, Robert McLaren, and Saleh Bleha. An application of fuzzy algorithms in a computer access security system. *Pattern Recognition Letters*, 9(1):39–43, 1989. URL <http://dblp.uni-trier.de/db/journals/pr1/pr19.html#HussienMB89>.
- [13] Willem G. de Ru and Jan H. P. Eloff. Enhanced password authentication through fuzzy logic. *IEEE Expert: Intelligent Systems and Their Applications*, 12(6):38–45, November 1997. ISSN 0885-9000. doi: 10.1109/64.642960. URL <http://dx.doi.org/10.1109/64.642960>.
- [14] Christophe Rosenberger Romain Giot, Mohamad El-Abed. Keystroke dynamics with low constraints svm based passphrase enrollment. In *IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, pages 1–6, 2009.
- [15] Typewatch. URL <https://www.watchfulsoftware.com/en/home>.
- [16] Intensity analytics. URL <http://www.intensityanalytics.com/>.
- [17] Biotracker. URL <http://plurilock.com/>.
- [18] Keytrac. URL <https://www.keytrac.net/>.
- [19] Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung il Kim. Web based keystroke dynamics identity verification using neural network. *Journal of Organizational Computing and Electronic Commerce*, 10:295–307, 2000.
- [20] Trustable password. URL <http://www.imagicsoftware.com/>.
- [21] biohec. URL <http://www.biohec.com/>.

- [22] behaviosec. URL <http://www.behaviosec.com/>.
- [23] Leonard Richardson and Sam Ruby. *Restful Web Services*. O'Reilly, first edition, 2007. ISBN 9780596529260.
- [24] Web services glossary, February 2004. URL <https://www.keytrac.net/>.
- [25] S. J. Press R. S. Gaines, W. Lisowski and N. Shapiro. Authentication by keystroke timing: Some preliminary results, 1980.
- [26] David Flanagan. *JavaScript: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 3rd edition, 1998. ISBN 1565923928.
- [27] The jQuery Foundation. jquery, . URL <https://jquery.org/>.
- [28] Jesse J. Garrett. Ajax: A new approach to web applications.
- [29] EllisLab Inc. Codeigniter. URL <http://ellislab.com/codeigniter>.
- [30] Steve Burbeck. Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc), 1987. URL <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>.
- [31] Phil Sturgeon. Codeigniter rest server.
- [32] Robin Schumacher and Arjen Lentz. Dispelling the myths. URL <http://web.archive.org/web/20110606013619/http://dev.mysql.com/tech-resources/articles/dispelling-the-myths.html>. MySQL.
- [33] The jQuery Foundation. jquery user interface, . URL <https://jqueryui.com/>.
- [34] Douglas Crockford. Javascript object notation. URL <http://json.org/>.
- [35] Avramidis Loukas. Keydy website. URL <https://83.212.108.72>.
- [36] Zurb. Foundation. URL <http://foundation.zurb.com/>.
- [37] E. Marcotte. *Responsive Web Design*. A book apart. A Book Apart, 2011. ISBN 9780984442577. URL <http://books.google.gr/books?id=vhe4XwAACAAJ>.
- [38] Same origin policy. URL http://www.w3.org/Security/wiki/Same_Origin_Policy.

-
- [39] Arun Ranganathan. cross-site xmlhttprequest with cors. URL <https://hacks.mozilla.org/2009/07/cross-site-xmlhttprequest-with-cors/>.