PEER-TO-PEER BIOMETRIC VERIFICATION USING NFC-ENABLED SMARTPHONES

by

Charl Anton Opperman

Submitted in partial fulfilment of the requirements for the degree
Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

February 2014

SUMMARY

PEER-TO-PEER BIOMETRIC VERIFICATION USING NFC-ENABLED SMARTPHONES

by

Charl Anton Opperman

Supervisor:	Dr G.P. Hancke
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Mobile identification, biometrics, speaker recognition, face recognition, smart sensing, NFC, RFID, Android, smartphones, ubiquitous computing

The current approach for person identification consists of handing over a paper ID document which consists only of a picture of the person's face, usually at a much younger age. Human intuition provides the means to make a comparison between the person in the picture and the person handing over the document. With the rise in personal electronic devices such as smartphones and tablets, alternative solutions such as mobile biometrics have become feasible especially since these security mechanisms are also being studied as solutions to securing the mobile devices themselves from unauthorised login. Instead of presenting a paper ID document, a person may present a contactless smartcard or even an NFC-enabled smartphone on which biometric data is securely stored. The data may be read from the card or the phone using another phone with NFC capability. The sensors of the receiving smartphone can then be used to take biometric measurements of the person to verify the biometric template that the person presented. In this way, the sensors and processor of a phone perform the recognition instead of a human, and extra layers of security can be added over paper ID documents by allowing traits such as voice, teeth, or hand geometry to be identifiable, instead of only the person's face. Additionally, the biometric data can easily be updated on an electronic device as a person ages, whereas the same photo on ID documents are currently used for many years to avoid the time-consuming reprinting and issuing processes.

A study was performed to determine the feasibility of a novel identification solution using NFC and biometrics on a smartphone. The solution allows the user to use a smartphone or an RFID tag as a replacement for a paper ID document without adding overhead to the identification process. NFC is a close-proximity communication technology and is utilised as the enabling technology for this solution by allowing users to touch phones together or to touch an RFID tag with a phone to read the required biometric data. NFC does not require any connection setup or device pairing, which means the “handing over” of an ID document is not substituted with a laborious electronic process, but rather by the simple touch of a phone. For the biometric system, the main focus was on speaker recognition using the built-in microphone of a smartphone, although a basic face recognition system was also developed using the built-in camera. The Google Nexus S was used as the platform for the smartphone implementation. The Nexus S runs the Android operating system which was used to develop biometric algorithms in both Java and native C/C++. Various open-source libraries were ported to Android mainly to provide pattern recognition algorithms.

For the first experiment a Java-based speaker recognition system was developed. The system was trained with a database of 27 speakers under various environmental conditions. In terms of performance, the smartphone could perform biometric enrolment and classifier training in about 400 milliseconds when distance-based classification algorithms were used. This excludes the time of the recording, which depends on the length of the sentence that is spoken. Verification of the biometric template can be performed in about 3 seconds. The processing time was compared with a standard PC and it was found that the phone was about 30 times slower, which is to be expected, but the processing time of most algorithms was still found to be very reasonable for the implementation of a user-friendly system. The identification accuracy of the system reached 82.76% for the top performing algorithm. Similarly, in the second experiment, it was found that the phone could perform training and verification in about 150 milliseconds for a face recognition application in C++. This was found to be about 21 times slower than a PC. In both experiments, no reduction in identification accuracy was observed when comparing the phone to the PC. A database with 40 subjects was used to train the system and identification accuracy reached 89.17%.

A third experiment was conducted to study a complete end-to-end proof-of-concept implementation of a smartphone-based peer-to-peer biometric system. A second speaker recognition application was developed and different variables were studied under varying conditions in the biometric system. These included the effect of audio sample rate, text-independence, and feature vector size on system accuracy, processing time and NFC transmission time. The system was trained with only a single user and 4 intruders attempted to verify themselves as the legitimate user. Upon changing the audio sample rate from 8 kHz to 44.1 kHz, no noticeable improvement could be observed in accuracy since the human voice does not generally extend to frequencies above 4 kHz. Feature vector size and text independence generally had a significant effect on the accuracy of the system. Text independence in this context refers to the user of the system being verified when speaking a sentence that was not used for enrolment. NFC transmission time was shown to vary from approximately 700 to 4000 milliseconds for different feature vector sizes stored on various RFID tags as well as peer-to-peer transmission between phones. A comparison was also made between two different programming languages, Java and C++, which are available for Android development. It was found that the processing time could be decreased by a factor of 9 when developing a native C++ application instead of Java, which requires applications to run on the Dalvik virtual machine in Android. Both the Java and C++ libraries of Android are quite limited (to suit the ARM architecture) and many modifications are required to port open-source desktop software to Android, even though Android is Linux-based. It was found, however, that the porting of C++ software to Android generally requires many more code modifications than Java which was expected since Java is generally considered very portable.

In South Africa, as in many other countries, paper ID documents are systematically being replaced by contactless smartcards with a stored biometric template on the card. NFC phones could theoretically read the biometric template from these cards and compare with a measured trait provided that the phone supports the applicable sensor. In the future, ID information could be stored in a secure element embedded in a smartphone or based in the cloud. The results of the experiments in this dissertation show that smartphones and tablets have become powerful enough to allow for user-friendly implementations of the identification solution proposed. The accuracy of voice and face biometrics are generally lower when compared to established technologies such as fingerprint biometrics. Fingerprint scanners are currently entering the mobile space and when combined with other modalities such as voice and face, very secure verification will be possible. The combination of biometric modalities, allows a higher level of certainty when identifying people. Multimodal biometrics does however increase the processing requirements, which is why performance analyses are important for mobile devices.

OPSOMMING

EWEKNIE-TOT-EWEKNIE BIOMETRIESE BEVESTIGING MET BEHULP VAN NFC-GEMAGTIGDE SLIMFONE

deur

Charl Anton Opperman

Studieleier:	Dr G.P. Hancke
Departement:	Elektriese, Elektroniese en Rekenaaringenieurswese
Universiteit:	Universiteit van Pretoria
Graad:	Magister in Ingenieurswese (Rekenaar Ingenieurswese)
Sleutelwoorde:	Mobiele identifisering, biometrie, sprekerherkenning, gesigsherkenning, slim sensors, NFC, RFID, Android, slimfone, onbeperkte verwerking

Die huidige benadering vir persoonsidentifisering behels die oorhandiging van ŉ papier ID dokument wat slegs uit ŉ foto van die persoon se gesig bestaan, gewoonlik by ŉ baie jonger ouderdom. Menslike intuïsie verskaf die metode om ŉ vergelyking te tref tussen die persoon in die foto en die persoon wat die dokument oorhandig. Met die styging in persoonlike elektroniese toestelle soos slimfone en tablette het alternatiewe oplossings soos mobiele biometrie haalbaar geraak, veral sienend dat hierdie sekuriteitsmeganismes ook bestudeer word as ŉ oplossing om die mobiele toestelle self te beskerm teen ongemagtigde toegang. In plaas daarvan om ŉ papier ID dokument te vertoon kan ŉ persoon ŉ kontaklose slimkaart vertoon of selfs ŉ NFC-gemagtigde slimfoon waarop biometriese data veilig gestoor is. Die data kan vanaf die kaart of foon gelees word deur middel van ŉ ander foon met NFC vermoë. Die sensors van die ontvangende slimfoon kan dan gebruik word om biometriese metings van die persoon te neem om die biometriese templaat te verifieer wat deur die persoon vertoon was. In dié manier voer die sensors en verwerker van die foon die herkenning uit in plaas van ŉ mens en ekstra vlakke van sekuriteit kan bygevoeg word oor papier ID dokumente deur toe te laat dat eienskappe soos stem, tande, of handafmetings identifiseerbaar is, in plaas van slegs die persoon se gesig. Daarbenewens kan die biometriese data maklik opgedateer word op elektroniese toestelle terwyl ŉ persoon

verouder, terwyl dieselfde foto op ID dokumente tans vir baie jare gebruik word om tydsame herdrukking en uitreikingsprosesse te vermy.

ŉ Studie was uitgevoer om vas te stel of ŉ nuwe identifiseringsoplossing wat gebruik maak van NFC en biometrie haalbaar is op ŉ slimfoon. Die oplossing laat die gebruiker toe om ŉ slimfoon of RFID merker te gebruik as ŉ plaasvervanger vir papier ID dokumente sonder om bo-koste by te voeg tot die identifiseringsproses. NFC is ŉ kortafstand kommunikasietegnologie en word gebruik as die bemagtigende tegnologie vir hierdie oplossing deur die gebruiker toe te laat om fone in kontak te bring of ŉ RFID merker te raak met ŉ foon om die benodigde biometriese data te lees. NFC vereis nie konneksie opstelling of toestel afparing nie, wat beteken dat die oorhandiging van ŉ ID dokument nie vervang word met ŉ moeisame elektroniese proses nie, maar eerder deur die eenvoudige aanraking van ŉ foon. Die hooffokus van die biometriese stelsel was op sprekerherkenning deur middel van die ingeboude mikrofoon van die slimfoon, alhoewel ŉ basiese gesigsherkenningsisteem wat gebruik maak van die ingeboude kamera, ook ontwikkel was. Die Google Nexus S was gebruik as die platform vir die slimfoon implementering. Die Nexus S hardloop die Android beheerstelsel wat gebruik was om die biometriese algoritmes te ontwikkel in beide Java en inheemse C/C++. Verskeie oopbron sagteware was gepoort na Android hoofsaaklik vir patroonherkenningsalgoritmes.

Vir die eerste eksperiment was ŉ Java-gebaseerde sprekerherkenningstelsel ontwikkel. Die stelsel was opgelei met ŉ databasis van 27 sprekers onder verskeie omgewingstoestande. In terme van werkverrigting kon die slimfoon biometriese inskrywing en klassifiseerderopleiding uitvoer in omtrent 400 millisekondes toe afstandsgebaseerde klassifiseringsalgoritmes gebruik was. Dit sluit nie die tyd in vir opname nie, wat afhang van die lengte van die sin wat uitgespreek word. Verifiëring van die biometriese templaat kan uitgevoer word in ongeveer 3 sekondes. Die uitvoeringstyd was vergelyk met ŉ standaard persoonlike rekenaar en dit was gevind dat die foon omtrent 30 keer stadiger was, wat te verwagte is, maar die uitvoeringstyd van meeste algoritmes was steeds bevind om baie redelik te wees vir die implementering van ŉ gebruikersvriendelike stelsel. Die identifiseringsakkuraatheid van die stelsel het 82.76% bereik vir die top presterende algoritme. Eweneens was dit in die tweede eksperiment gevind dat die foon opleiding en verifiëring kon uitvoer in omtrent 150 millisekondes vir ŉ gesigsherkenningstoepassing in C++. Dit was bevind dat hierdie toepassing 21 keer stadiger was as op ŉ persoonlike rekenaar. In beide eksperimente was geen afname in identifiseringsakkuraatheid waargeneem toe die foon met die persoonlike rekenaar vergelyk is nie. ŉ Databasis met 40 onderwerpe was gebruik om die stelsel op te lei en die identifiseringsakkuraatheid het 89.17% bereik.

ŉ Derde eksperiment was uitgevoer om ŉ punt-tot-punt bewys-van-konsep implementering van ŉ slimfoon-gebaseerde eweknie-tot-eweknie biometriese stelsel in geheel te bestudeer. ŉ Tweede sprekerherkenningstelsel was ontwikkel en verskeie veranderlikes was bestudeer onder variërende omstandighede in die biometriese stelsel. Dit het ingesluit die invloed van monsteringstempo, teks-onafhanklikheid, en kenmerkvektor lengte op stelsel akkuraatheid, verwerkingstyd en NFC oordragtyd. Die stelsel was opgelei met slegs ŉ enkele gebruiker en 4 indringers het gepoog om hulself te verifieer as die regmatige gebruiker. Na afloop van ŉ verandering in die oudiomonsteringstempo vanaf 8 kHz na 44.1 kHz, was geen opmerkbare verbetering waargeneem in akkuraatheid nie, aangesien die menslike stem gewoonlik nie frekwensies hoër as 4 kHz bereik nie. Kenmerkvektor grootte en teks-onafhanklikheid het oor die algemeen ŉ aansienlike invloed op die akkuraatheid van die stelsel gehad. Teks-onafhanklikheid in hierdie konteks verwys na die gebruiker van die stelsel wat geverifieer word wanneer ŉ sin uitgespreek word wat nie gebruik was tydens inskrywing nie. NFC oordragtyd was bewys om te varieer vanaf ongeveer 700 tot 4000 millisekondes vir verskillende kenmerkvektor groottes, gestoor op verskillende RFID merkers, en ook eweknie-tot-eweknie oordrag tussen fone. ŉ Vergelyking was ook getref tussen twee verskillende programmeringstale, Java en C++, wat beskikbaar is vir Android ontwikkeling. Dit was bevind dat die verwerkingstyd met ŉ faktor van 9 verminder kon word wanneer C++ toepassings ontwikkel word in plaas van Java, wat benodig dat toepassings op die Dalvik virtuele masjien hardloop in Android. Beide die Java en C++ sagtewarebiblioteke in Android is taamlik beperk (om by die ARM argitektuur aan te pas) en baie aanpassings is nodig om oopbron sagteware na Android te poort al is Android op Linux gebaseer. Dit was egter gevind dat om C++ sagteware na Android te poort baie meer kode aanpassings benodig oor die algemeen as Java, wat te verwagte was aangesien Java oor die algemeen as baie poortbaar beskou word.

In Suid-Afrika, soos in baie ander lande, word papier ID dokumente stelselmatig vervang met kontaklose slimkaarte met ŉ gestoorde biometriese templaat op die kaart. NFC fone kan teoreties die biometriese templaat vanaf hierdie kaarte lees en vergelyk met ŉ gemete eienskap solank die foon die toepaslike sensor ondersteun. In die toekoms kan ID informasie gestoor word in ŉ sekuriteitselement wat in ŉ slimfoon ingebed is of in die

wolk gebaseer is. Die resultate van die eksperimente in hierdie verhandeling wys dat slimfone en tablette kragtig genoeg geword het om gebruikersvriendelike implementerings van die voorgestelde identifiseringsoplossing toe te laat. Die akkuraatheid van stem- en gesigsbiometrie is in die algemeen laer wanneer dit vergelyk word met gevestigde tegnologieë soos vingerafdrukbiometrie. Vingerafdrukskandeerders is tans besig om die mobiele spasie in te tree en wanneer dit gekombineer word met ander modaliteite soos stem en gesig, sal baie veilige verifiëring moontlik wees. Die kombinasie van verskeie biometriese modaliteite laat ʼn hoër vlak van sekerheid toe wanneer mense geïdentifiseer word. Multimodale biometrie verhoog egter die verwerkingsvereistes, wat die rede is waarom werkverrigtingsanalises belangrik is vir mobiele toestelle.

LIST OF ABBREVIATIONS

AAR	Android application record
ANN	Artificial neural network
API	Application programming interface
ATLAS	Automatically tuned linear algebra software
CCS	Core card services
CSV	Comma-separated values
DEP	Data exchange protocol
FAR	False accept rate
FFT	Fast fourier transform
FMR	False match rate
FNMR	False non-match rate
FRR	False reject rate
GCC	GNU compiler collection
GPRS	General packet radio service
GUI	Graphical user interface
HMM	Hidden markov model
HSPA	High speed packet access
IR	Identification rate
JNI	Java native interface
JVM	Java virtual machine
LPC	Linear predictive coding
LTE	Long term evolution
MARF	Modular audio recognition framework
MCU	Microcontroller unit
MNO	Mobile network operator
NDEF	NFC data exchange format

NFC	Near field communication
OEM	Original equipment manufacturer
OS	Operating system
OTA	Over-the-air
PC	Personal computer
PCA	Principal component analysis
PCM	Pulse code modulation
PGM	Portable graymap
PIN	Personal identification number
RFID	Radio frequency identification
SE	Secure element
SDK	Software development kit
SNR	Signal-to-noise ratio
SVM	Support vector machine
TSM	Trusted service manager
WEKA	Waikato environment for knowledge analysis

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

After the invention of radio frequency identification (RFID) technology, sensing applications utilising RFID soon followed. Sensors were provided with the ability to communicate wirelessly at a close distance, with no effect on power consumption [1]. When the data generated by these sensors required processing, a computer was utilised, which meant that a link was necessary between a computer and the sensor. Traditionally, a specialised RFID reader would be used to gather data from the sensor, after which the reader would be physically or wirelessly connected to a computer in reasonably close proximity [2]. Another approach would be to fit the sensor with a microprocessor or microcontroller, to enable local processing of data, but this meant that the sensor would consume much more power. Mobile devices can offer a quicker and more portable approach.

High-end smartphones have recently started to feature near field communication (NFC) capabilities. NFC, a successor to RFID, is a close-range communication technology that is compatible with RFID, but also features a peer-to-peer communication mode between active devices. This means that NFC phones can read and write data on RFID tags (or RFID sensors) and also transmit data to other NFC phones by simply touching devices together. It therefore eliminates the need for the time-consuming connection setup that is required by other similar peer-to-peer communication technologies.

As the processing power and memory capacity of smartphones gradually increase, they become more suitable for processing-intensive applications that were previously limited to specialised hardware, desktop computers and laptops [3]. Mobile phones have undergone a metamorphosis to a general-purpose, mobile computer with a plethora of sensors, input-output devices and wireless connection capabilities. A wide scope of applications have emerged involving mobile devices, as is evident from the numbers of mobile “apps” that are available for each platform on their respective “app markets”. One area that benefits from the rise of mobile devices is in measurement and processing applications, such as biometric recognition. This project studies the feasibility of using smartphones as biometric devices.

1.1.2 Research gap

In applications where the security requirements are very strict, a forgeable paper or plastic identity card may not be adequate. Electronic identification documents are already being introduced in many countries, including South Africa. Many of these documents contain an RFID chip with stored biometric data (in South Africa a thumb print biometric template is stored on the card). The biometric template can be read from the card and compared to the template generated from measuring the applicable trait of the person to determine if the card is legitimate. Authenticity and integrity of the biometric template on the RFID tag can be ensured using a digital signature of the issuing authority.

RFID tags can be made very small and it is possible to insert these tags physically into the human body. Although many argue that there would be numerous benefits to such an approach, such as increased national and personal security and reduced medical risks, others argue that this approach is an extreme invasion of privacy and allows people to be tracked everywhere they go. Nevertheless it has been found that the overall acceptance of implanted RFID chips have been steadily increasing due to three issues: terrorism, identity theft, and the convenience of not having to physically carry around documents and not having to memorise personal identification number (PIN) codes.

Previous works have studied the implementation of biometrics on smartphones for the purpose of authenticating users of the devices. In these approaches, biometric recognition would be used to replace a PIN. It is however also possible to use biometrics on smartphones for the purpose of identifying peers. In this approach, smartphones can be used to replace paper-based identification documents. RFID-based identification documents are already in use and the use of NFC phones would ensure compatibility with these documents. Since NFC is currently enabling smartphones to replace wallets, keys and tickets, this project proposes that NFC-smartphones could also be used to replace identification documents.

The study of a peer-to-peer biometrics application on smartphones has various ramifications, such as enabling user-friendly data transfer for the casual user, keeping the recognition time low while providing adequate accuracy, mitigating the possibility of fraud, etc.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The main objective of this project was to quantitatively study the feasibility of a security approach based on smartphones and biometric recognition, and to analyse various ramifications and practical issues of such a system. The viability of mobile biometrics in conjunction with NFC was researched with specific focus on usability, speed, accuracy and security to determine if such an approach could be adopted instead of paper-based identification documents. A peer-to-peer biometrics system was developed and tested. The objective of the system is to enable any user in possession of an NFC-enabled smartphone, to be able to identify another individual by touching phones together (or touching the RFID tag of an individual with the phone) and then taking measurements of biometric traits of the individual using the sensors of the phone (such as a camera, microphone, accelerometer).

The following research questions were posed.

- Can smartphones be utilised as a platform for biometric recognition with reasonable performance in terms of recognition accuracy and processing time? Do general purpose mobile devices consequently provide a viable alternative to specialised biometric equipment?
- How will a high-end smartphone perform in comparison with a standard personal computer (PC) running a desktop Linux distribution?

- Will open-source software libraries for feature extraction and pattern classification algorithms be portable to a smartphone platform, and specifically to the Android platform? If so, will these libraries provide adequate verification accuracy?
- How will system factors such as feature vector size affect NFC transmission time, identification accuracy, and processing time?
- How will the performance differ between non-native Java and native C++ implementations of similar biometric applications on a mobile platform? Considering other factors, such as code availability and portability, will Java or C++ provide a more feasible solution for the implementation of biometric algorithms?

1.3 HYPOTHESIS AND APPROACH

It was hypothesised that a current high-end smartphone would possess adequate memory and processing capabilities, as well as the necessary sensor availability and acuity, to be utilised as an off-the-shelf biometric verification device. A uni-modal or multi-modal biometric application can be executed on such a phone by utilising pattern recognition algorithms provided by open-source software libraries, which should be portable to mobile platforms when edited and recompiled. Short range data transmission may be facilitated by the built-in NFC capability of a smartphone, which can act as an enabling communication technology for the implementation of an interactive biometric application that ultimately aims to replace identification documents with mobile devices. NFC also provides the means to read and store data on RFID tags, which is a technology that is already used in identification documents.

The research approach was to develop biometric software and to benchmark the performance thereof for both a PC and an Android smartphone, in C++ and Java. Open-source libraries were ported to Android and compared to the identical implementations on the PC. The main focus was on speaker recognition, although an initial face recognition application was also implemented to study the possibility of multi-modal biometrics. Two applications, one in Java and one in C++, were implemented on both a standard PC and a Google Nexus S smartphone to analyse the relative processing power of each and a third application was developed only on the phone, but in both Java and C++ to compare native and non-native code on Android, as well as to explore the effect of various system configurations on system performance.

1.4 RESEARCH GOALS

At the onset of this research project, the following goals were set.

- To develop a security system in which users in possession of a smartphone should be able to identify one another dynamically by touching smartphones and then taking one or more biometric measurements using for example the phone’s camera or microphone. Algorithms running on the phone can then compare the measured trait with the stored trait.
- To determine whether current smartphones possess enough processing power for the execution of a complex sensing and processing task, such as biometric recognition. Consequently, a conclusion could be reached on whether general purpose mobile devices provide a viable alternative to specialised equipment in the biometrics space.
- To determine which biometric system configurations provide the best performance on a smartphone and which trade-offs should be taken into consideration when developing such as system.
- To analyse the portability of open-source desktop software to the Android platform in both C++ and Java, and to compare the porting process between the two languages.
- To analyse the difference in processing time that native C/C++ code provides on an Android phone when compared to standard Java code.
- To determine, using the processing time, accuracy and NFC transmission time, whether a peer-to-peer biometric system on a smartphone is practically feasible.
- To study the feasibility of biometric data storage and retrieval on RFID tags.

1.5 RESEARCH CONTRIBUTION

Although the use of biometric authentication on smartphones has been explored in literature, the work has mostly been directed at user log-in authentication and not at peer-to-peer identification. This project explored the use of smartphones for peer-to-peer biometric authentication and considered smartphones as a portable and ubiquitous alternative to specialised biometric equipment, as well as an electronic alternative to paper identification documents.

In this project, the RFID based identification method is extended with its successor technology NFC. Instead of still having to carry around the electronic RFID version of identification documents, a mobile phone may be used by itself, providing all the required processing, communication and security requirements. This approach negates the need for any other equipment. A mobile biometrics application can benefit from using NFC for the passing of data in the form of biometric templates or identification information, such as business, cards between phones at close physical proximity. NFC is very quick when passing only a small amount of data because there is no set-up required to initiate communication as with other short range wireless technologies. The compatibility between NFC and RFID also enables the use of RFID tags as small storage devices for biometric template data and other applicable identification data.

The main quantitative research contributions of this research project were to measure the processing capabilities of mobile devices in the context of biometric pattern recognition under various system configurations and to study code portability from desktop devices to mobile devices. Processing time and application simplicity from the user’s perspective were used to analyse the usability of the proposed system.

1.6 OVERVIEW OF STUDY

This dissertation discusses the implementation of biometric recognition applications on a high-end Android smartphone. Existing open-source software was ported to the Android platform in both C++ and Java and then interfaced with the built-in sensors of the phone using the Android application programming interface (API). These applications serve both as processing benchmarks as well as a proof-of-concept implementation.

Three experiments were carried out for this research project, in addition to a literature study on NFC, RFID sensing, biometric recognition in general, and various specific biometric modalities such as face and speaker recognition. For the first experiment, a speaker recognition application was developed in Java, using the MARF (modular audio recognition framework) open-source library. This application was used to compare the processing power between a PC and a high-end smartphone when running Java applications, as well as to set an initial standard for the biometric verification accuracy on a smartphone. For the second experiment, OpenCV (open-source computer vision) was used for the implementation of a simple face recognition application in C++. The application was used for a performance comparison of C++ code between a PC and a smartphone. In the third experiment, a second speaker recognition application was implemented to study a complete end-to-end biometric system with NFC communication. System performance was studied under various configurations and a comparison was also made between a Java and a C++ implementation on an Android smartphone. In Java the library jAudio was used and in C++ the SPro library was used for the implementation of linear predictive coding (LPC) feature extraction. For classification, a simple distance-based algorithm was implemented to measure the distance between feature vectors.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OBJECTIVES

This chapter will present a summary of the literature study that was carried out for this research project. An overview of NFC communication technology will firstly be given in Section 2.2, followed by a summary of application scenarios using NFC phones in Sections 2.3 and 2.4. A high-level discussion of biometric recognition will then be given in Section 2.5, and the chapter will conclude with previous similar work and the novelty of this research in Section 2.6.

2.2 NEAR FIELD COMMUNICATION

Near field communication or NFC, a successor to RFID, was standardised by the NFC Forum, which was founded in 2004 [4]. The NFC standard is defined in ISO 18092 [5] and the equivalent ECMA-340 [6] standard and is compatible with RFID tags that comply with the ISO 14443 standard. NFC devices are also specifically compatible with the well-known RFID tag brands MIFARE and FeliCa, by Philips and Sony, respectively [7].

NFC provides the following advantages over legacy RFID:

- NFC provides a new peer-to-peer transmission function in addition to the standard RFID tag reading and writing functions. This means that NFC "reader/writer" devices can also communicate with each other [8].
- NFC can be used to initialise faster connections such as Bluetooth and Wi-Fi seamlessly. This is known as "Connection Handover" [9].
- NFC is incorporated into mobile phones, which are ubiquitous. This means simple, low-power RFID sensors can connect to mobile phones to gain internet access indirectly via general packet radio service (GPRS) or high speed packet access (HSPA). These low-power sensors can also take advantage of the processing power of NFC-enabled smartphones [10].
- NFC reader/writer devices can also emulate RFID tags. This function is used for electronic keys and ticketing, as well as mobile payments. Data is stored on a secure element on the phone [8], [11].

NFC operates across a very short range and is unique for its user-friendly "touch" or "tap" interaction, even though it is a wireless technology. Low power consumption means that NFC is very applicable in mobile devices.

2.2.1 Technical details of NFC

An active NFC device (such as an NFC phone) features 3 modes of operation; active communication mode, passive communication mode and card emulation mode [8].

2.2.1.1 Active communication mode

In active mode both the NFC initiator and target devices are self-powered devices and both devices generate and modulate their own oscillating (13.56 MHz centre frequency) magnetic fields to transmit data, successively. This is shown in Figure 2.1.

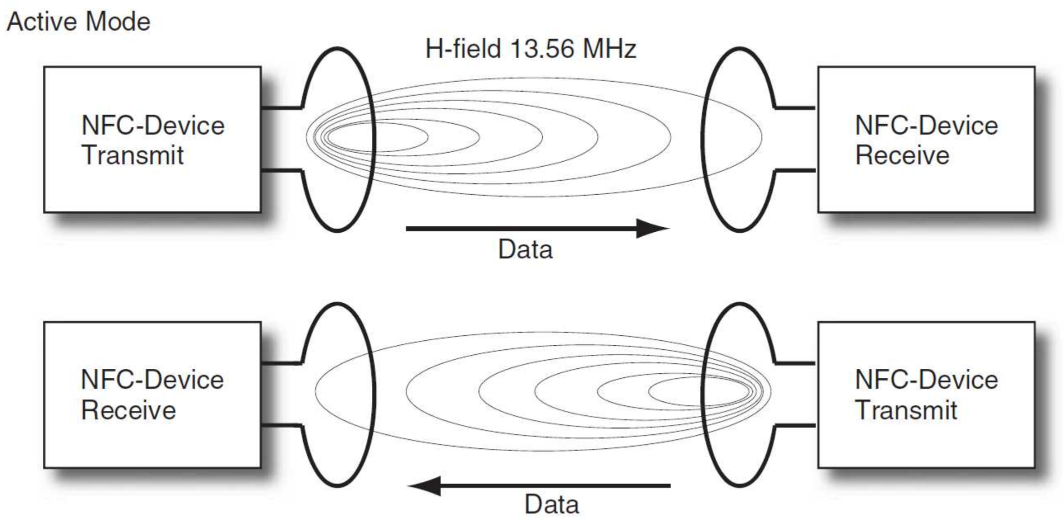


Figure 2.1. Peer devices successively generate oscillating magnetic fields to send data in NFC active communication mode [12]

Devices can communicate at data rates of 106, 212 and 424 kbps. To initiate communication between two devices they simply need to be brought into close proximity of one another. A simple handshake is performed between two active devices at initialisation to determine each other's IDs, data rate capabilities and other connection-specific information before commencing data transfer. The data exchange protocol (DEP) defined in ECMA-340 is then activated to exchange data between devices [5], [6]. Data may be exchanged in the NFC data exchange format (NDEF), which allows the transfer of images, URLs, text, and other high level data [13]. Raw data may also be transferred, in which case custom commands may be defined.

2.2.1.2 Passive communication mode

In passive mode, the NFC initiator communicates with an RFID tag or with another active NFC device operating in card emulation mode. The initiator generates a 13.56 MHz oscillating magnetic field which is induced in the circuitry of the target device to power the device and to transmit data to the device as illustrated in Figure 2.2. The target device responds by load-modulating the magnetic field generated by the initiator. This is performed by varying the impedance of the target in accordance with the data sequence intended for transmission [8]. Changes in the impedance of the target cause slight variations in the magnitude of the magnetic field, which can be detected by the initiator.

NFC supports RFID tags that operate in accordance with the standards ISO 14443 (proximity cards) [14] and ISO 15693 (vicinity cards) [15].

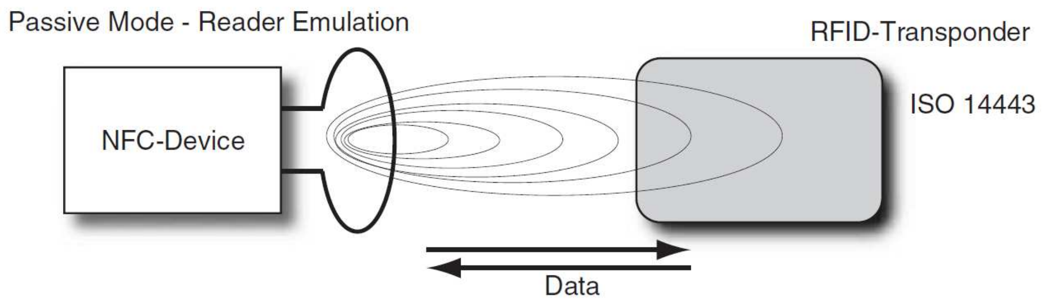


Figure 2.2. An NFC reader communicates to an ISO 14443 compliant RFID tag in passive communication mode [12]

2.2.1.3 Card emulation mode

Card emulation mode is essentially an active NFC device "pretending" to be a passive RFID tag as shown in Figure 2.3. Even though the device is self-powered, it does not generate a magnetic field but instead utilises load modulation to communicate data to an

NFC/RFID reader. The term "card emulation" is mostly used for NFC devices which are also able to support reader emulation and peer-to-peer modes, while the term "semi-passive RFID device" is used for RFID devices that are self-powered but always communicate passively [16]. Semi-passive RFID tags are useful for cases in which the device needs to be active when a reader is not present to provide energy.

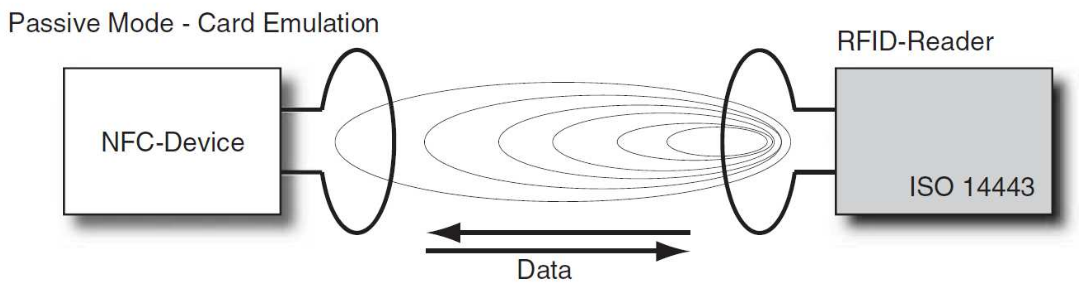


Figure 2.3. An NFC device in card emulation mode being read by an ISO 14443 compliant RFID reader [12]

2.2.2 Comparison of NFC with other short-range wireless technologies

There are two main points which differentiate NFC from other short range wireless technologies. The first is the extremely short range over which NFC acts and the second is the fact that NFC devices can communicate to both active and passive devices, which opens up a myriad of applications that are not accessible to other short range wireless technologies. Some advantages of the short range include the inherent security that such a short range provides against eavesdropping and the natural user friendly touch interaction, which is easily understood by non-technical users. Table 2.1 gives a simplistic comparison of several aspects between NFC, Infra-red, Bluetooth technologies.

Table 2.1. A comparison of short range communication technologies

	NFC	**Infrared**	**Bluetooth**
Operating distance	Very small	Small	Small
Implementation cost	Low	Relatively low	Relatively high
Power consumption	Passive or very low	Relatively high	Relatively high
Usability	Simple. Only a touch or tap is required	Simple	Relatively complex. Requires device pairing

2.2.3 Applications of NFC

NFC is an attractive technology because of its inherent user-friendly touch-based interaction. It is intuitive to use and does not require any form of protocol set-up. The integration of NFC in mobile phones means that an NFC device will be available to users any time to serve as an electronic interaction proxy for services that are expected to become NFC-enabled in the future. Some of the main applications in which NFC is currently applied are as follows.

- Mobile payments or m-payments [17], [18]. This is discussed in detail in Section 2.3.
- Electronic ticketing [8]. RFID tags are often used as tickets for events and public transportation. However, NFC phones provide the card-emulation function with which a phone may be used as a virtual ticket-holder.
- Location-based services [19]. Objects may be touched with a phone to gain useful information based on the location of the touched objects.
- Smart posters and large displays. Posters with embedded RFID tags can be touched with phones to get additional information on certain subjects [20]. Large displays can also be touched with the phone if a large screen is required temporarily to display content that cannot be viewed on the phone itself [21].

- Sensing and digital control applications [10], [22].
- Shopping applications (RFID tags may be used as a replacement for barcodes, for example) [8].
- Health and medical applications [23].
- General data transfer between phones and reading and writing of RFID tags. The slower data rate of NFC can be overcome by using NFC for connection handover to faster technologies such as Bluetooth and Wi-Fi. In this way the simple tap-action of NFC can be used to initiate Wi-Fi or Bluetooth communication, instead of the usual device-pairing that is required.
- Identification of animals, objects and people [8], [24]. Electronic passports or e-passports and electronic ID documents or eIDs are already being issued in many countries [25], [26]. In South Africa, a pilot project will commence in late 2012 to systematically replace ordinary ID documents with smart-IDs [27].

2.3 MOBILE PAYMENTS

Google Wallet was the first major deployment of m-payments using NFC based payment systems with wide retail functionality [28]. A secure element (SE) in the phone is used to store payment cards electronically and this allows contactless payments to be made by tapping a phone against contactless acquiring devices with support for technologies such as Mastercard PayPass and Visa payWave. The advantage that is gained over plastic contactless payment cards is that over-the-air (OTA) services can be provided such as reloading prepaid cards, downloading payment cards, and other value added services such as coupons can be managed remotely. This is done by a trusted service manager (TSM) that facilitates the connection between the phone and the financial institution, whilst managing the secure element remotely.

Widespread adoption of mobile payments has been stifled by a general dispute regarding the form factor of the secure element that is used to store payment cards. Three form factors are available, an embedded SE in the phone can be sold as part of the phone and access to the element will be owned by the original equipment manufacturer (OEM), SIM card based SE can be used where the mobile network operator (MNO) has ownership of the SE, and lastly financial institutions that prefer ownership of the SE can issue SD cards with an embedded SE. Ownership of the SE has been a hot topic and consensus has not yet been reached on which form factor to pursue. In fact, across the globe different MNOs, financial institutions, TSMs, and other institutions have piloted and rolled out all the different form factors. In South Africa, Absa bank is currently piloting SD card based NFC payments on BlackBerry and Samsung phones. A global shift may have been made towards embedded SEs since the very popular Samsung Galaxy S4 was recently released with an embedded SE and the Visa payWave payment applet preloaded.

2.4 SENSING WITH NFC-ENABLED SMARTPHONES

There are two main sensing scenarios in which NFC phones are utilised. The first scenario involves the use of external sensors, in which an NFC phone is simply used to read stored data from the sensor or possibly power a passive sensor to take measurements [22]. In the second scenario, smartphones are utilised as the sensing devices and NFC is simply used for the passing of data between phones and possibly for the storage of data on RFID tags [29].

Connecting to external sensors by utilising NFC provides various advantages over other communication technologies. The following are some of these advantages.

- Passive sensing is possible. NFC and RFID allow the energising of passive targets wirelessly. The passive target can perform sensing, processing and communication tasks when an NFC phone comes into close proximity thereof.
- For low power sensors, a passive, low cost NFC interface can be built and fitted to existing sensors [22]. In this way low power sensors are given access to the processing capabilities of a smartphone as well as the broadband network access without adding any significant cost or power consumption to the sensors.
- Smartphones are already ubiquitous, which means special hardware and extra costs are unnecessary.
- NFC allows easy data storage on RFID tags, if required in the specific sensing application.

Three types of NFC sensor implementations are described in [16]. They are passive sensors, user controlled semi-passive sensors and stand-alone semi-passive sensors for long-term monitoring. Passive sensors only activate when an active NFC device, such as an NFC phone, is brought into close proximity of the sensor. An example is the use of a moisture sensor which could be built into walls to detect pipe leaks. Because there is no need for a battery, the device can be permanently embedded into a wall or floor. User controlled semi-passive sensors take measurements when a predetermined user action occurs like the push of a button or bringing an active NFC device into close proximity of the device. The sensor is powered externally but uses less power than a stand-alone sensor because it can enter a low power state in between user interactions. A stand-alone semi-passive sensor should be able to take measurements when activated by user interaction but also when external events occur or when internal timer intervals elapse. An example application is the use of an NFC enabled vibration sensor fitted onto fragile deliveries. When the delivery reaches its destination a mobile reader can be used to check if any harmful vibrations occurred during transportation. A general NFC interface was designed in [22] to interface with existing sensors. Figure 2.4 shows the prototype NFC-enabled heart rate monitor that was developed as a proof-of-concept. It is shown that this interface can connect to a wide range of existing sensors using the analogue and digital interfaces of a microcontroller unit (MCU).

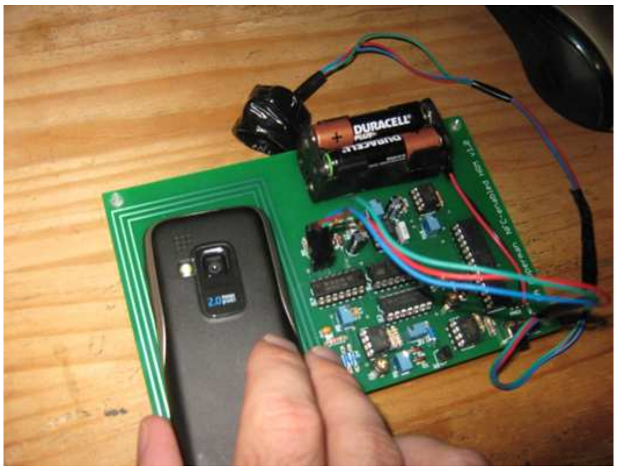


Figure 2.4. A proof-of-concept NFC-enabled heart rate monitor showing how external sensors may be fitted with NFC interfaces to enable communication with mobile devices

Smartphones with 3rd generation HSPA, and soon 4th generation LTE (long term evolution), allow the transmission of large amounts of data to remote locations for storage or processing. However, current smartphones are very adequate processing devices themselves and transmission is not necessary in many cases. Smartphones themselves are fitted with a wide range of sensors, which means the phone can be used for sensing applications with NFC used as an enabling communication technology. Figure 2.5 gives a graphical representation of a person's voice being recorded on a smartphone for subsequent processing (biometric verification in this project). Location-based smartphone technologies such as GPS and also NFC can help to give context to the sensing application by providing information regarding the current location of the phone. An example of such a context function for NFC is in electronic ticketing, where the phone will expect certain types of noise in voice recordings on a train or bus, for example. Location-based contextual information can help in interpreting sensing data more accurately.

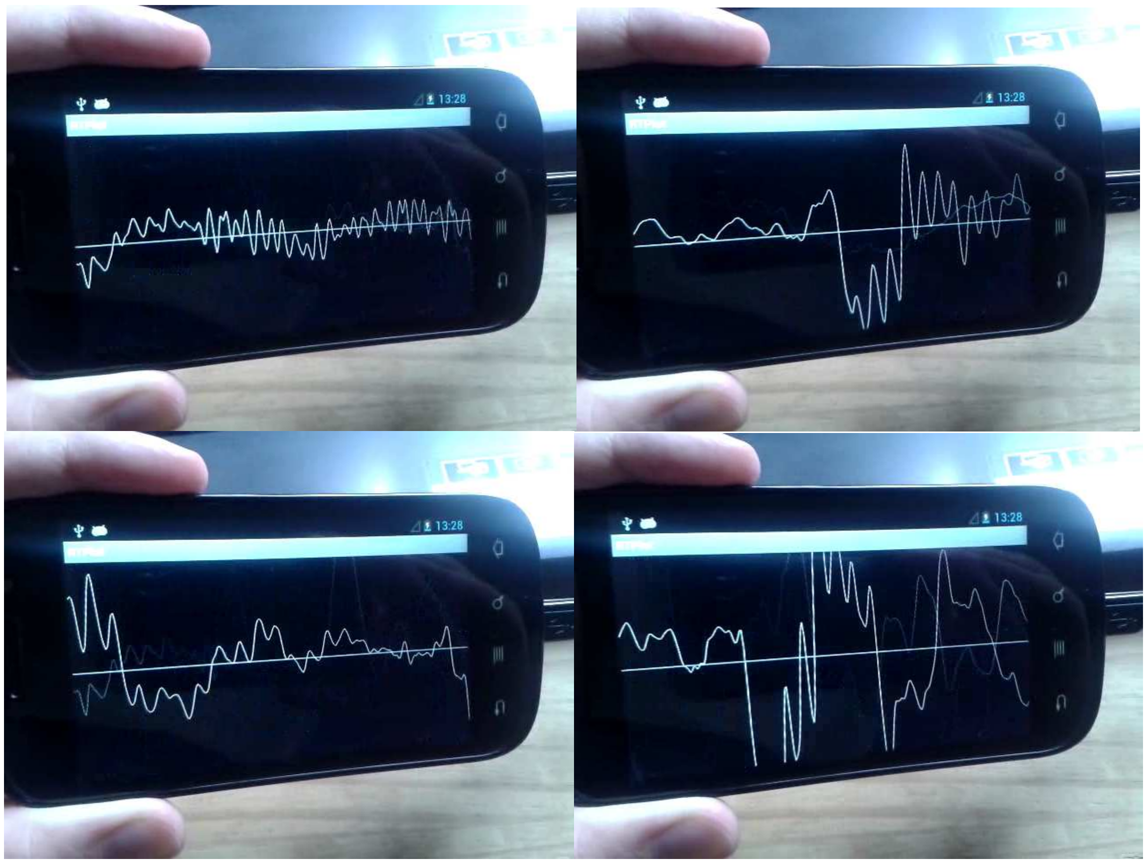


Figure 2.5. Voice recording on a smartphone for subsequent processing

The specific application scenario will always determine whether or not standalone sensors will be required or whether a smartphone alone is sufficient for taking measurements. Despite the processing capabilities of smartphones, some applications may still require a back-end computer if a large display is required for data visualisation, for example (however NFC-enabled tablet computers are also available or the large display function of NFC could be employed, as explained in Section 2.2.3).

2.5 BIOMETRIC RECOGNITION

Biometric recognition refers to any process that automatically identifies a person based on physiological or behavioural traits [30]. This definition may refer to the computerised processes that have started to gain popularity in recent years, but it also refers to the seemingly simple act of human beings recognising one another by the shape and colour of their faces, eyes and hair (physiological biometrics) and by the way they walk, sit and stand (behavioural biometrics) [31].

2.5.1 Biometrics in general

Biometric recognition or biometrics can either consist of methods that are used to verify the claimed identity of a person; or to determine the identity of a person by matching characteristics with a set of possible candidate identities. The former is also known as biometric verification and the latter is known as biometric identification. Identification and verification are different processes and should be dealt with separately when designing or studying a biometric system [32].

Physiological and behavioural traits of a person need to satisfy four conditions to qualify as a candidate biometric identifier. Firstly, the trait needs to be sufficiently complex or involve sufficient detail to be unique to each person (distinctiveness). Secondly, the trait should be something that every person possesses (universality). Thirdly, the trait should not change much over time (permanence). Lastly, it should be possible to measure the trait quantitatively (collectability). In practice, there are however several other factors to take into consideration when designing a computerised biometrics system. These include performance of the system, including accuracy and speed; acceptability by users, in other words the system shouldn’t undermine the privacy of users; and the system should also be secure against circumvention [32].

A general biometric system consists of a sensor, a feature extractor, a matcher, and a database with templates of legitimate users [31], as shown in Figure 2.6. The sensor is used to convert a biometric trait of a person to an electronic form. From the electronic data, the feature extractor can extract distinctive information to create a biometric template. The template or feature vector can be compared to legitimate feature vectors using the matcher module. The matcher gives an output that quantifies the similarity between templates, usually in the form of a single number. When the number determined by the matcher exceeds a certain threshold *t*, the system positively identifies or verifies the user. When biometric verification is performed, as is done in Figure 2.6, the user will use their claimed identity as an index to select the corresponding template stored in the database. A one-to-one comparison is then performed between the selected template and the generated template to determine if the person is who he or she claims to be. On the other hand, when biometric identification is performed, the user does not claim any identity, but only allows the measurement of a trait to create a template which is compared with all the templates in the database to determine if any matches occur. In other words, the direct link between the user and the database disappears in the case of identification, and matching with templates in the database occurs until a match is found (when the matching score exceeds the threshold value *t*). In both cases of verification and identification, there may also be a link between the feature extraction module and the database, which represents the database templates being updated every time a successful match occurs. By updating the database regularly, future errors can be reduced by accounting for the gradual ageing of people, for example.

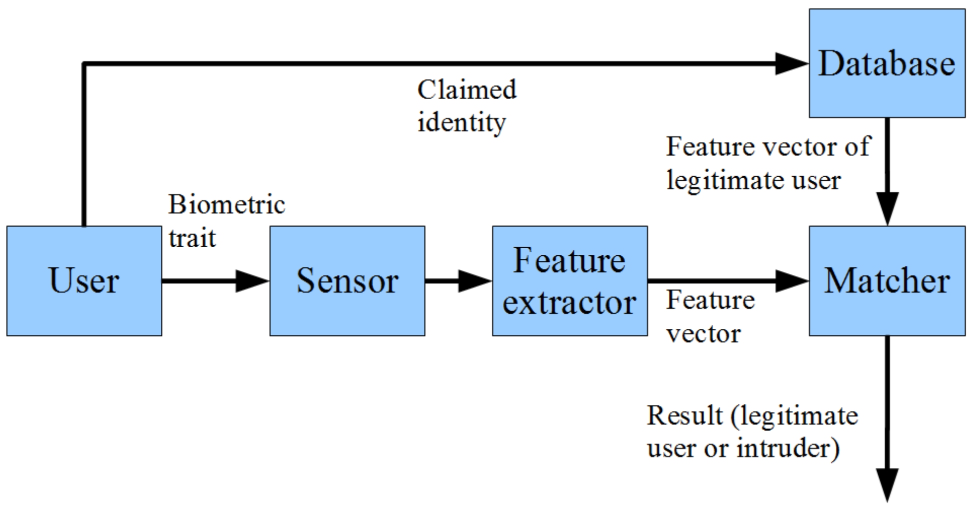


Figure 2.6. A general biometric verification system

The implementation of a biometrics system differs depending on whether the emphasis for a specific application is on usability or on security. A trade-off is made between the frequency of errors that identify intruders as legitimate users, also known as false match rate (FMR) or false accept rate (FAR), and the frequency of errors that identify legitimate users as intruders, also known as false non-match rate (FNMR) or false reject rate (FRR) [31]. Figure 2.7 shows how the threshold value *t* can be varied to perform this trade-off. The threshold value is the number that is chosen to be compared to the output of the matching module of the biometric system. If the output of the matcher is higher than *t*, the person is positively recognised and if it is lower than *t*, the person is classified as an impostor. For very secure applications the threshold is chosen very high while it is chosen much lower for civilian applications and also for forensic applications, for different reasons. In forensics it is acceptable to deal with many false matches while searching for the true match. If the threshold value were chosen very large to avoid false matches, the probability of a false non-match would increase and the guilty criminal may not be found which would be unacceptable. In civilian applications however, biometric authentication is usually required at regular intervals. Convenience is therefore given priority and the threshold value is also lowered, as in forensics, to avoid false non-matches, which would mean legitimate users are often identified as intruders. The price of this convenience is that false matches would increase and intruders could therefore enter the system more easily. In very secure circumstances this would not be acceptable and security is given priority over convenience by increasing the threshold value to avoid false matches.

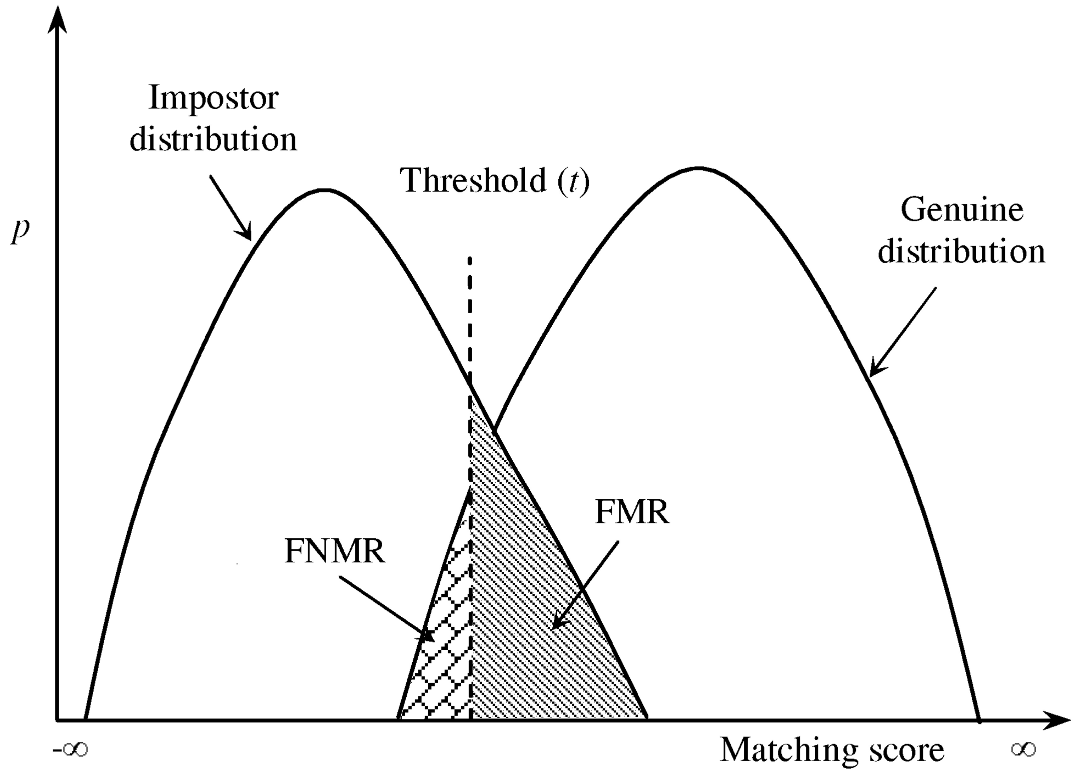


Figure 2.7. The overlapping of the FNMR and FMR rates. A trade-off can be made by varying the threshold *t* [33] (© IEEE 2004)

It should be noted that biometric identification has a higher FMR than verification because the input feature vector is compared to many templates whereas it is only compared to one template in the case of verification. The more users there are in the database, the higher the probability of a false match error becomes [34] and this presents a problem for large identification systems. In this research project the focus was on biometric verification because the system operates in a peer-to-peer fashion, which means that peers claim their identities to each other before being verified. Therefore, the identification problem in a system with a large database of users was not specifically addressed.

2.5.2 Commonly employed biometric methods

There are many different biometric methods that have been explored and new ones are discovered routinely as different hardware and software capabilities arise in computers and sensors. Table 2.2 summarises some of the best and worst biometric traits in terms of the four attributes of good biometrics [32].

Table 2.2. Some of the best biometric traits regarding the 4 attributes of a good biometric trait

Attribute of a good biometric trait	**Best examples of biometric traits**	**Worst examples of biometric traits**
Universality	Face, iris, retina, DNA, odour	Keystroke, signature
Distinctiveness	Odour, fingerprint, DNA, palm-print, retina, iris	Gait, keystroke, face, voice
Permanence	Hand gesture, odour, ear, fingerprint, DNA, palm-print	Gait, keystroke, voice, signature
Collectability	Face, teeth, voice, signature, hand geometry, gait, gesture	Odour, retina, DNA

From the table it is clear that many of the most popular biometric methods score high in terms of collectability. Collectability of biometric traits depends on the kind of sensors that are available to measure these traits, and how expensive and mobile these sensors are. Because of the widespread availability of cameras and microphones, face recognition and voice recognition are two of the most popular biometric methods in non-critical applications, even though these methods do not score particularly high in terms of distinctiveness and permanence. Currently, the most applicable biometric techniques for mobile devices are image based methods using the built-in camera, such as face recognition and hand geometry recognition, as well as speaker recognition using the built-in microphone. Gait and hand gesture recognition can also be implemented accurately using the accelerometer and gyroscopic sensors of a smartphone. Some smartphones also include a fingerprint reader for logging into the phone.

2.5.3 Multi-modal biometrics

Using mobile devices for biometrics requires measurements to be made under varying environmental conditions. This can cause significant performance degradations because of the difficulty of pattern recognition with varying measurement conditions and background noise. Multi-modal biometrics can be implemented to alleviate this problem by combining uncorrelated biometric procedures to increase the accuracy of the results. This is easily achievable given the wide array of sensors that are available in current smartphones. A disadvantage of such an approach is that the speed of the system is adversely affected because of the multitude of measurements that need to be carried out. There are however ways to combine biometric methods in such a way that all measurements can be performed simultaneously. An example of such an approach is to perform face recognition while a person's voice is being recorded for speaker recognition and his or her teeth photographed when opening their mouth while speaking [35].

In a multi-modal system, biometric methods may be integrated in various stages of the recognition process. This gives rise to following three fusion levels in multi-modal biometrics [36].

- Feature extraction level. Feature vectors of different biometric modalities are combined to form a new feature vector that will be classified as an intruder or legitimate user. Biometric fusion at the feature extraction level retains most of the information in each modality and should therefore deliver the best results. It is however difficult to normalise and integrate the feature vectors of different biometric methods.
- Matching level. At the matching level, a possible fusion approach would be to add the matcher output values of each biometric method using a weighted sum, where larger weights are given to more accurate methods. Fusion at the matcher level is used most often in the industry due to the limitation of access to the feature vectors imposed by the designers of biometric equipment [36].
- Decision level – Fusion at the decision level retains least information of each biometric method, but is the easiest to implement. A majority voting rule is usually

used at this level by counting the number of biometric modalities that determine a legitimate user versus the number of modalities that determine the user is an imposter and choosing the highest number of "votes".

2.6 SIMILAR WORK

In a similar research paper [25], NFC was utilised together with identification information stored electronically in a secure element on a mobile phone, to replace paper-based ID documents. This method enhances the privacy of persons by allowing only certain identification information to be requested and transmitted electronically, instead of having to present a full ID document with exhaustive information about a person that may be irrelevant in certain circumstances. Biometric procedures were included in the research, but the actual recognition was not performed using the built-in sensors of a mobile phone.

Various research projects have studied the feasibility of advanced sensing, and specifically biometric recognition applications, using mobile devices. However, as sensor acuity and processing capabilities continuously increase in these devices, so does the accuracy and processing speed increase in algorithms implemented on them. For the implementation of biometrics on mobile devices, many authors recommend the use of multi-modal biometrics for the mobile environment [35], [37], [38]. Intuitively, multi-modal biometrics introduces additional resource usage over uni-modal biometrics, which is not a trivial matter when it comes to mobile devices. This additional depletion of resources for multi-modal systems is however important in many scenarios, because mobile biometrics systems are less accurate than static systems in general, due to continuous changes in the surrounding environment and this causes a lower average signal-to-noise ratio (SNR). To increase the SNR, various biometric methods are combined in a single system and this is known as multi-modal biometrics. Similarly, in other advanced sensing applications, additional resources may often be employed to mitigate loss of accuracy, but this puts a strain on the limited resources of mobile devices. An example of a multi-modal system for the mobile environment may be to perform face recognition while a person's voice is being recorded for speaker recognition and his or her teeth photographed when opening their mouth while speaking [35]. In such a case all biometric measurements may be performed simultaneously and an adverse effect on the usability of the system is therefore avoidable, given that the processor can handle the additional data load swiftly.

Smartphones are equipped with a range of sensors, including light sensors, proximity sensors, accelerometers, microphones, and cameras. Recent studies of applications for these sensors have led to novel biometrics approaches to gesture and gait recognition using built-in accelerometers and gyroscopic sensors in smartphones. A person exerts certain unique acceleration patterns on a phone when walking with it in their pocket. This unique walking pattern is known as the gait of a person, which can be detected and matched to the legitimate owner of the phone [39], [40]. Acceleration sensors in smartphones can also be utilized for a gesture recognition system, which detects a certain pattern in which a person waves or moves their arm when holding the phone [41]. The owner of a phone may then, for example, unlock the phone by drawing a certain unique pattern with their arm in the air.

An evaluation of the security of uni-modal biometric methods of thumb-print recognition, face recognition and speaker recognition using mobile devices is performed in [42]. It is found that these methods are foiled quite easily using a fake thumb, a photo of a person, and a voice recording of a person, respectively. However, when using mobile phones for authentication between two individuals, the person performing the authentication of another person is physically present and is therefore in a position to detect and prevent such malicious activities.

Other example applications of mobile measurement have been developed in several different areas of research. A prominent application is health monitoring using mobile phones [22], [43]. Area mapping is commonly performed using GPS, and novel approaches have even seen indoor mapping by magnetic field using the magnetometer (compass) of a smartphone [44]. Audio, image and video processing can be performed using the camera and microphone of a smartphone. Mobility monitoring of the physically disabled has been proposed using the accelerometer and GPS capabilities of a phone, with additional context information provided by the camera [45].

A study in [46] compared the processing time of native C++ code and non-native Java code on Android by implementing real-time video processing software on the phone. It was found in that on average native code performed 2.4 times faster. In this dissertation, however, a much larger performance increase was measured for native code (see Chapter 4).

CHAPTER 3 METHODS

3.1 RESEARCH EQUIPMENT

A Samsung Google Nexus S smartphone was utilized as the testing platform. The Nexus S runs the Android mobile operating system (OS) which, in addition to Java programming, provides a framework for developing native applications in C/C++ code and interfacing this code with a Java application using the Java native interface (JNI). Some of the relevant specifications of the Nexus S are compared with that of the PC in Table 3.1 [47].

Table 3.1. Comparison of the smartphone specifications with that of the PC

Feature	**Google Nexus S smartphone**	**Desktop PC**
Operating system	Android v4.0.4	Ubuntu Linux v11.10
Processor type	ARM Cortex-A8	Intel Quad Core
Processor frequency	1 GHz	4 x 2.66 GHz
Instruction set	32-bit	32-bit
Memory	512 MB	3 GB

In addition to providing NFC communication abilities, the Nexus S is a powerful mobile computing device in terms of processing capability and memory, and also very high-resolution cameras and a range of other sensors. The Android OS is an open-source project created by Google. The project is well-documented for application developers and a thorough software development kit (SDK) is provided for rapid development. Both Java and C++ based pattern recognition algorithms were implemented and evaluated in this project. At the sensor level, native code is generally not as well-supported as non-native code and a combination of Java and C/C++ was therefore used, where sensing was always performed in Java but processing was performed in both languages, using JNI. Pattern recognition algorithms were not implemented from first principles, but open-source software libraries were rather ported to Android and linked to the developed applications. The porting of open-source libraries ensured that an accurate comparison could be made between the processing capabilities of the phone and the PC. The following libraries were ported to Android, although not all of these were directly applied in the biometric applications.

- MARF is a Java library for speech, sound and voice recognition, and natural language processing. The library was ported to Android and used for speaker recognition in a non-native application.
- The OpenCV library supports many pattern recognition algorithms, including artificial neural networks (ANNs) and support vector machines (SVMs), as well as feature extraction functions for images. A port to the Android native C++ interface is available for the library. OpenCV is a large, well-documented project.
- jAudio is an audio feature extraction library in Java.
- Waikato environment for knowledge analysis (WEKA) is a large, well-documented open source Java library for general machine learning and data mining tasks. Distance based classification algorithms, as well as ANN, SVM and many others are implemented.
- SPro is an audio feature extraction library in C.
- Shogun is a large-scale machine learning toolkit with a large range of classification algorithms.

WEKA and Shogun were ported for the future implementation of more complex classification algorithms than the distance-based algorithms that were implemented using MARF and OpenCV. These include among others ANN, SVM and hidden markov model (HMM) implementations. OpenCV also provides functions for these classifiers. The general nature of these libraries also provides the means to implement other biometric modalities such as palm-print and gesture recognition.

3.2 FIRST EXPERIMENT: JAVA SPEAKER RECOGNITION BENCHMARK

The MARF open-source library contains several algorithms for audio, speech, and natural language processing [48]. For this experiment, MARF was used to develop a speaker recognition system. The software was implemented identically on a PC and an Android phone. The porting of MARF to Android required several modifications, which included the removal of code with a dependence on Java graphical user interface (GUI) libraries (the AWT and Swing libraries of Java are not available in Android). MARF also uses standard Java libraries to load WAV audio files, which are not available in Android and various modifications were therefore required.

3.2.1 The MARF library

MARF is a text-independent speaker identification library, which means firstly that the system can identify a speaker by name from a group of possible speakers, and secondly that a speaker can be identified when pronouncing any phrase. The library is written in Java for portability between platforms, and to facilitate automatic memory management and other tasks which may require undue effort on the part of application developers. MARF is structured to consist of 4 distinct stages of processing when performing speaker recognition. These 4 stages are audio data loading, preprocessing, feature extraction and classification. The library includes, among others, fast fourier transform (FFT) filters and normalization for preprocessing; FFT, LPC and Min/Max amplitudes algorithms for feature extraction; and neural networks and distance-based algorithms (for example Euclidean, Manhattan and Minkowski distance) for classification.

3.2.1.1 Audio loading

The library can load audio files in the WAV format as well as in a textual format. The library provides a database of WAV files for testing purposes, and the WAV loader was therefore used. The loader is required to convert the pulse code modulation (PCM) data in the WAV files, to actual amplitude values that can be processed.

The WAV file loader of MARF utilises standard Java libraries in the "javax.sound" package to read audio files. This package is not present in Android, however, and had to be taken and modified from the OpenJDK open-source implementation of Java. Java also uses a method to obtain a concrete implementation of the WAV file reader class at runtime, and this was not possible using Android. A concrete implementation of the file could however be obtained from the GNU Classpath project (WAVReader.java), and this ensured that a WAV loader class could be obtained at compile time, for use on Android.

3.2.1.2 Preprocessing

In the preprocessing stage, all algorithms normalise the input, except for the Raw algorithm, which directly passes the input data through without any modification. The Dummy algorithm, on the other hand, performs normalisation on the data but no other processing. Normalization of data is important in the mobile environment, because speakers do not speak at the same volume consistently, nor do they speak at the same distance from the microphone with each interaction. All other preprocessing algorithms in the library perform filtering or amplification functions on the data after normalisation. This enhances the quality of the speech signals embedded in the audio data, by isolating and amplifying the speech signals, while attenuating the non-speech frequencies.

The library provides a set of FFT filters, and they are available as low-pass, high-pass, band-pass, high-boost, and high-pass-boost filters. The difference between a high-pass filter and a high-boost filter is that the high-boost filter does not attenuate low frequencies, but instead only amplifies higher frequencies while keeping lower frequencies intact. The high-pass-boost filter is a combination of a high-pass and a high-boost filter. The cut-off frequencies of all filters are chosen appropriately to be applied to human speech. The endpoint preprocessing module inspects the speech signal for local maxima and minima throughout the signal. These maxima and minima points are then stored and the rest are discarded. Figures 3.1 and 3.2 show a speech sample before and after FFT bandpass preprocessing (including normalisation), respectively.

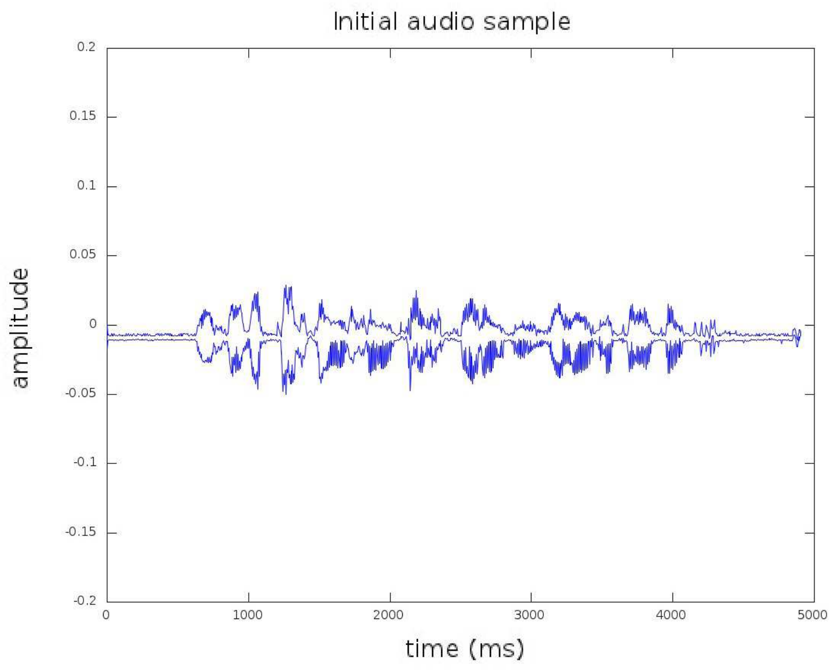


Figure 3.1. Original audio sample fed into the MARF pipeline

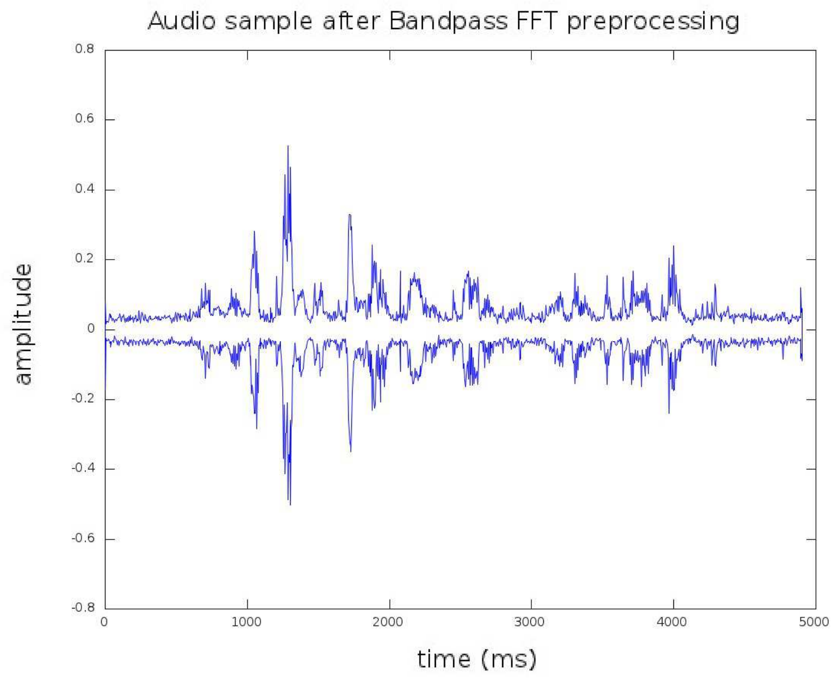


Figure 3.2. The audio sample of Figure 3.1 after normalisation and bandpass FFT preprocessing using MARF

3.2.1.3 Feature extraction

After preprocessing has been performed, a normalised array of numbers is available for processing. The goal of the feature extraction stage is to reduce the number of points while retaining as much speech information as possible. The feature extraction module therefore searches the data for salient points. Each data-point in the feature vector is considered a single dimension in a multi-dimensional hyperspace and by decreasing the number of data points in the vector, the number of dimensions of the vector is effectively reduced. Feature extraction is therefore also known as dimensionality reduction [49].

There are only 3 useful feature extraction algorithms that have been implemented in MARF. They are FFT, which uses the fast fourier transform to extract features; LPC, which is an algorithm used to obtain the spectral envelope of the signal [50]; and Min/Max, which stores an array of minimum and maximum values extracted from the signal.

The Random algorithm simply extracts random points from the feature vector and is therefore not a useful method for a real system. It does however provide a baseline implementation to compare other methods with. Another feature provided by MARF is that different feature extraction methods may be aggregated to further reduce the dimensions of the feature vector and thereby further discriminating between information-rich speech content and noise or unwanted data. In addition, the MARF library also provides a Raw feature extractor, but it is not recommended for general use due to the large dimensions of the resulting feature vector. Figure 3.3 shows the speech sample in Figure 3.1 after Bandpass-filter preprocessing (Figure 3.2) and FFT feature extraction.

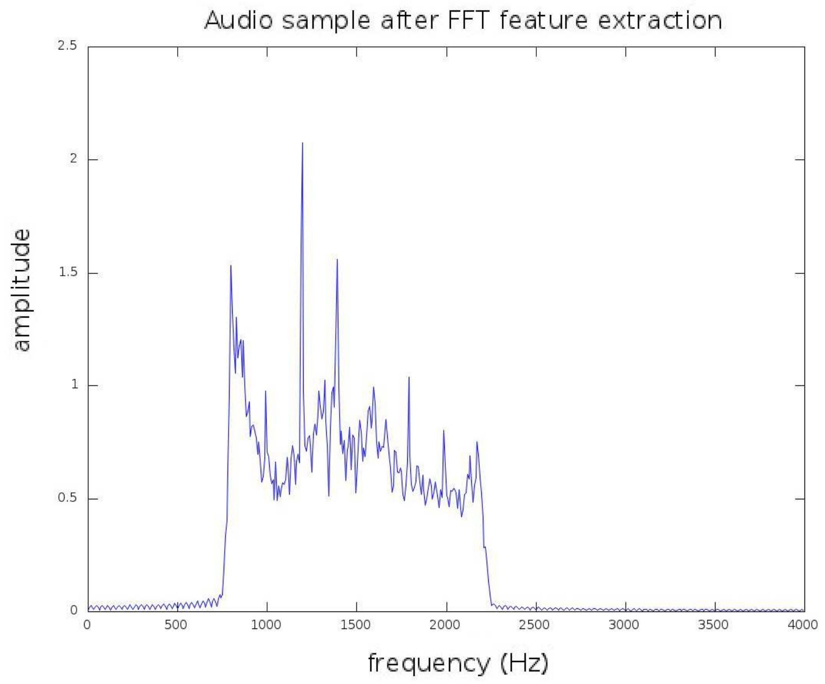


Figure 3.3. The preprocessed signal of Figure 3.2 after FFT feature extraction using MARF

3.2.1.4 Classification

The classification stage offers the actual recognition of the person speaking. In previous stages, the data have been processed and shaped to be in a proper format to be classified into predetermined classes (speakers). Up to the classification stage, MARF performs exactly the same functions regardless of whether the system is in training or in testing mode. In the classification stage, however, MARF will either train the classifier with the incoming processed data or classify the incoming data using the previously trained classifier.

MARF provides 6 useful classifiers in its latest stable version (version 0.3.0.5), and a random classifier which can be used as a baseline implementation for comparison. Of these 6 classifiers, 5 are distance-based classifiers. Each distance-based classifier uses a simple formula to measure the distance (or some function thereof) between feature vectors in an *n*-dimensional hyperspace (where *n* is the number of components or coordinates in the vectors) to determine their similarity. If the distance is larger than a pre-set threshold, the user is classified as an intruder. Distance-based approaches are also known as template matching [51]. MARF also provides a customisable neural network implementation.

3.2.2 Experimental set-up

The MARF library can be used in two ways. The first way is to define all algorithms that will be used, and then to simply call a function telling the MARF library to start processing. In this mode MARF handles the loading and saving of files, as well as the passing of data between various stages of the recognition process. The other way that MARF may be used is to implement each stage separately and perform the storing and passing of data manually. The second approach requires much more overhead and the first method was therefore chosen. A Java program was written to take arguments specifying which algorithms to use. A Bash script was then written to train the system for all possible combinations of algorithms on the PC implementation. A script was also written to test all possible combinations on the PC in identification mode, and an Android application was developed to test all combinations on the phone, after porting the MARF library to Android.

When running the program on the PC, it was split into four processes, one for each processor. On the phone, only one process (and one thread) was run at a time. Unnecessary processes were killed on the phone before running the program to free up resources. Both training and testing of the system was performed using a speaker database provided by the MARF library.

3.2.2.1 Speaker database

The speaker database contains a total of 293 WAV files for training the system. These files correspond to 27 different speakers, which are specified in a text file that is readable by the MARF system. For the testing of the system, there are 29 WAV files. These files contain audio of the speakers pronouncing different phrases than in the training situation, including different environmental noise. The WAV files used in the database are encoded in a single channel (mono), signed 16-bit PCM format with a sample rate of 8000 Hz, and each sample ranges from about 7 to 20 seconds [48]. According to the Nyquist theorem, the maximum recognisable frequency in the audio data is 4000 Hz, which is high enough for human speech under regular conditions. The database contains both male and female speakers and the samples are of relatively low quality to ensure that the system is properly tested for various environmental conditions. This is pivotal to ensuring that the system is tested with varying data, which would be the norm when using a mobile biometrics system.

3.2.2.2 Training

The system was trained using all possible combinations of algorithms in MARF, and each combination was trained on all 293 training samples (27 speakers) included in the library. Complete training of all algorithm combinations was performed on the PC only. On the smartphone, training was performed only in a few single runs. The processing time was found to be quite large and the phone implementation was therefore not trained using all combinations (The training sets generated on the PC was copied to the phone to be used for identification).

In total, for training on the PC, the entire MARF pipeline was executed 82040 times, or in other words the system was trained 280 times on all 293 samples. There were however errors in some of the algorithm combinations, mainly due to an “out of memory” error in Java. These errors were especially prevalent in neural network implementations. The Java virtual machine (JVM) memory size was increased up to about 1GB, but this amount of memory was still not enough to facilitate the training of a neural network for some combinations of preprocessing and feature extraction techniques.

In total 262 of the algorithm combinations were trained successfully, which is a 93.59% success rate. These 262 successfully trained algorithm combinations generated about 1.5 GB of data that needed to be stored for the system to remain “trained”. Most of these data were generated by the neural network combinations, which use XML files as output in version 0.3.0.5 instead of the GZIP format that is used by other classifiers. The Nexus S provides 16 GB of storage, which was more than enough for the training sets to be stored.

3.2.2.3 Testing

To test the system, the Java program written for the training of the system was modified to perform testing, again on all possible combinations of algorithms, on 29 testing samples. Testing was performed exhaustively on both the PC and the smartphone. The processing time for each algorithm was logged for each sample the system identified correctly and incorrectly. By counting the correct and incorrect matches, the FNMR could be estimated for these specific sets of training and testing samples.

For the testing of the system (identification mode) on the PC, the entire MARF pipeline was run 8120 times or in other words 29 samples were classified using 280 different algorithm combinations. Again, as with training of the system, there were some errors in some combinations of the algorithms. Of the 280 combinations, 257 of them classified all 29 samples without errors. That is a 91.79% success rate. Several of the combinations also performed worse than the random baseline implementation, suggesting that internal errors in the program may have caused faulty classifications.

On the smartphone implementation there were some additional problems. Three algorithms would not process successfully at all: they were the High-Freq-Boost-FFT preprocessing module, the Aggregator feature extraction module, and the Neural Network classifier. On the smartphone, the MARF pipeline was therefore executed 4872 times in total, or 168 algorithm combinations identified 29 samples. An additional single combination gave an error, which gave a processing success rate of 99.4% after the three aforementioned erroneous algorithms had been left out.

3.3 SECOND EXPERIMENT: C++ FACE RECOGNITION BENCHMARK

The OpenCV open-source computer vision library contains several hundred algorithms for computer vision, which include algorithms for image processing, video processing, feature detection, object detection and many more [52]. For this experiment, OpenCV was utilized for the development of a face recognition system on a PC and on Android. Manual porting of OpenCV to Android for native development was not required, because the developers of

OpenCV provide an up-to-date ported Android version. A comparison was made with the PC implementation to analyse the increase in processing time on a smartphone. As with the speaker recognition system, the exact similarity of the implementations on both the PC and the phone ensured that there were no differences in the identification accuracy of the implementations and the processing time could therefore be isolated and studied.

A very simple and well-known face recognition algorithm, known as Eigenface or principal component analysis (PCA), was implemented. PCA was utilized for feature extraction and distance-based classification algorithms were then applied. The classification algorithms that were implemented were Euclidean Distance, Manhattan Distance, Minkowski Distance and Diff Distance.

For data loading, OpenCV provides functions that are similar to those of Matlab. For example, “imread” is used to load an image of any supported format, “imshow” is used to display the image in a new window and “imwrite” is used to write an image to file. Images are contained in matrix (Mat) objects for processing in an OpenCV application.

3.3.1 Eigenface/PCA

OpenCV provides a complete implementation of the PCA algorithm for feature extraction. Figure 3.4 shows, from the left, the average face and the first 4 eigenfaces (eigenvectors) of a database of faces.

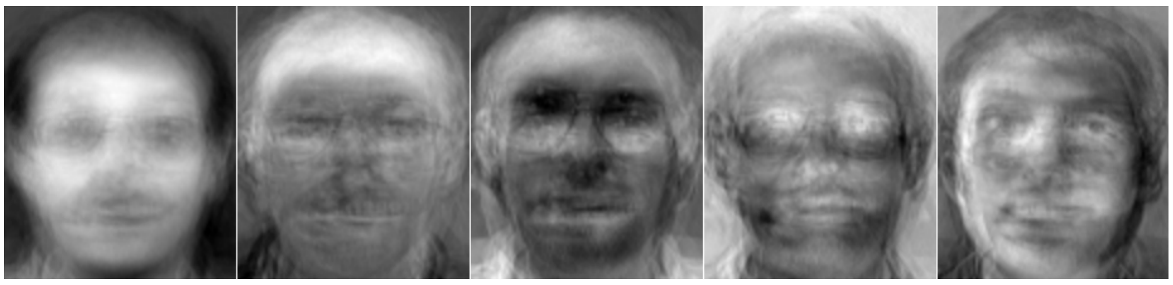


Figure 3.4. From the left, the average face and the first 4 eigenfaces of a set of faces from the AT&T Faces database

Although PCA is not a feature extraction algorithm that delivers the most accurate results, it provides advantages such as storage capacity and speed. When implementing PCA, all images are approximated as a weighted sum of the calculated average face and the eigenface components. The more eigenface components calculated, the more accurate the approximation will be, but this increases the resource usage of the algorithm. For the Eigenface method, each face in a training set does not have to be stored, but instead only the weights of the weighted sum of the eigenfaces can be stored for all faces. This saves storage space for systems that work with large numbers of faces. Some disadvantages of PCA include reduced accuracy under different lighting conditions, poses, and image scaling.

3.3.2 Euclidean distance

After feature extraction using PCA, distance-based classifiers are applied to measure the distance between faces in the training database and the faces that need to be identified. The Euclidean distance classifier determines the distance between feature vectors using (1).

$$d(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \qquad (1)$$

where d is the scalar distance between the feature vectors $\boldsymbol{x}$ and $\boldsymbol{y}$,
n is the number of dimensions in both $\boldsymbol{x}$ and $\boldsymbol{y}$, and
x_k and y_k represent the k^{th} scalar elements in $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively.

The equation computes the actual, straight-line distance between two points in a multi-dimensional hyperspace. The distance is compared between the vector that will be identified (***x***) and all training vectors (***y***), and the ***y*** vector that corresponds to the shortest measured distance *d*, is chosen as the vector that belongs to the same subject as ***x***.

3.3.3 Manhattan distance

Instead of taking the square root of the sum of squares, as in Euclidean distance computations, the Manhattan distance algorithm simply takes the absolute values of the difference between the vector elements, as in (2).

$$d(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{n} |x_k - y_k| \qquad (2)$$

It is quite surprising to note the increase in accuracy of the Manhattan distance over the Euclidean distance algorithm. This increase in accuracy was notable in both the speaker recognition system and the face recognition system.

3.3.4 Minkowski distance

The Minkowski distance, given in (3), represents a general case of the two distance algorithms mentioned above. For $r = 2$, (3) becomes (1) and the equation is equivalent to the Euclidean distance; and for $r = 1$, (3) becomes (2) and the equation is equivalent to the Manhattan distance. In this experiment, the Minkowski distance algorithm gave the most accurate results when using $r = 0.2$.

$$d(\mathbf{x},\mathbf{y}) = \left[\sqrt{\sum_{k=1}^{n} \left(|x_k - y_k| \right)^r} \right]^{\frac{1}{r}} \qquad (3)$$

3.3.5 Diff distance

The Diff Distance was developed by one of the creators of the MARF library, Serguei Mokhov in 2005 [48]. The Diff distance algorithm “rewards” or “penalizes” the distance between vectors depending on the measured values. If the distance between two corresponding points in the feature vectors is large, it is penalized and becomes even more distant, and if two points are very close, they are rewarded with a smaller distance value (usually zero). In this way the algorithm can provide better inter-class variation. The values of the penalty and the bonus, and the criteria for when the penalty or bonus should be applied, should be chosen carefully to ensure that the intra-class variation is not increased to the point where near-random classification results are obtained.

3.3.6 Experimental set-up

The face recognition application executed much faster than the speaker recognition application (partly due to less algorithms being implemented), and it was therefore not necessary to automate the entire process with a bash script. The OpenCV version that was used on both the PC and phone is version 2.3.1. Java wrappers for many of the classes in the library are available for the Android platform, but they were not implemented. Instead, C++ functions were used directly and these functions were called from the main Java class in Android, using the native interface (JNI). In this experiment, the program was not split into different processes or threads. Training and testing of the system was performed simultaneously and no training data is stored. The database with training and testing images can be obtained freely from the Olivetti Research Laboratory.

3.3.6.1 Face database

A face database was obtained from AT&T Laboratories Cambridge which contains a total of 400 greyscale image files for training and testing of the system. There are 40 subjects with 10 files for each subject. Better accuracy results are obtained if more files are used for training, but then less files are left for fair testing of the system (testing the system with the same files that were used for training will always produce good results and will therefore not produce accurate results). It was decided to use 4 files for training (per subject) and 6 for testing.

The image files in the face database are in the portable graymap (PGM) format and have a resolution of 92x112 pixels. The grey level of each pixel is represented by 8 bits (256 grey levels per pixel). The images are quite variable in terms of poses, hair, and facial accessories (mostly glasses). They are however not very variable in terms of background scenery and lighting.

3.3.6.2 Training and testing

Training and testing were performed simultaneously in a single program and the timing values measured are therefore given for the total number of images (400) for both training and testing. In this experiment, there were no processing errors such as with the MARF library, and the processing time can therefore be compared easily between the phone and the PC. The system was tested using only the PCA algorithm for feature extraction, and the 4 distance-based classification methods that were discussed earlier. No preprocessing was necessary, although images from the phone's camera in realistic mobile situations may require lighting normalisation, face detection and scaling for preprocessing.

In total the Eigenface algorithm was performed for all 400 images, while classification was performed 240 times. Each of these 240 classifications involves a one-to-one match with each of the 160 training images. This is much less than for the MARF implementation and the processing time was much less in total. The processing time per image sample could however be compared to the processing time per audio sample for the speaker recognition system. However, the audio data contained about 100 kB of data, whereas each image only contains 10.3 kB of data. When comparing the average processing time per sample, this needs to be taken into account, as well as the fact that preprocessing was performed in most cases for the speaker recognition system.

3.4 THIRD EXPERIMENT: END-TO-END NFC-ENABLED BIOMETRIC SYSTEM ANALYSIS

For this experiment a proof-of-concept biometric system was implemented with the ability to verify the identity of people using NFC-enabled smartphones and RFID tags. A second speaker recognition application was developed for this experiment and the effects of various system configurations were analysed. The system consists of two applications, the first representing the application that an issuing authority would use to generate biometric templates on a smartphone, and the second for the consumer to pass his or her biometric template over NFC, write the template to a tag, or to verify other users. Figure 3.5 shows a block diagram of the system.

In Figure 3.5 the issuing application is used to enrol a user by recording his or her voice several times and extracting LPC coefficients from the recording. Training then consists of simply computing the average LPC coefficients of all the recordings and this average feature vector is stored as the user’s biometric template. The biometric template can either be stored on a smartphone or an RFID tag if the user does not possess an NFC-enabled smartphone. The user application can read the biometric template from phone storage and pass it to another phone by tapping the phones together. The user application running on the other phone will start running automatically upon connection and allow this user to record the voice of the first person for subsequent identification. Optionally, the first user may also write his or her template to an RFID tag, which may be presented to the second user equipped with an NFC-enabled phone.

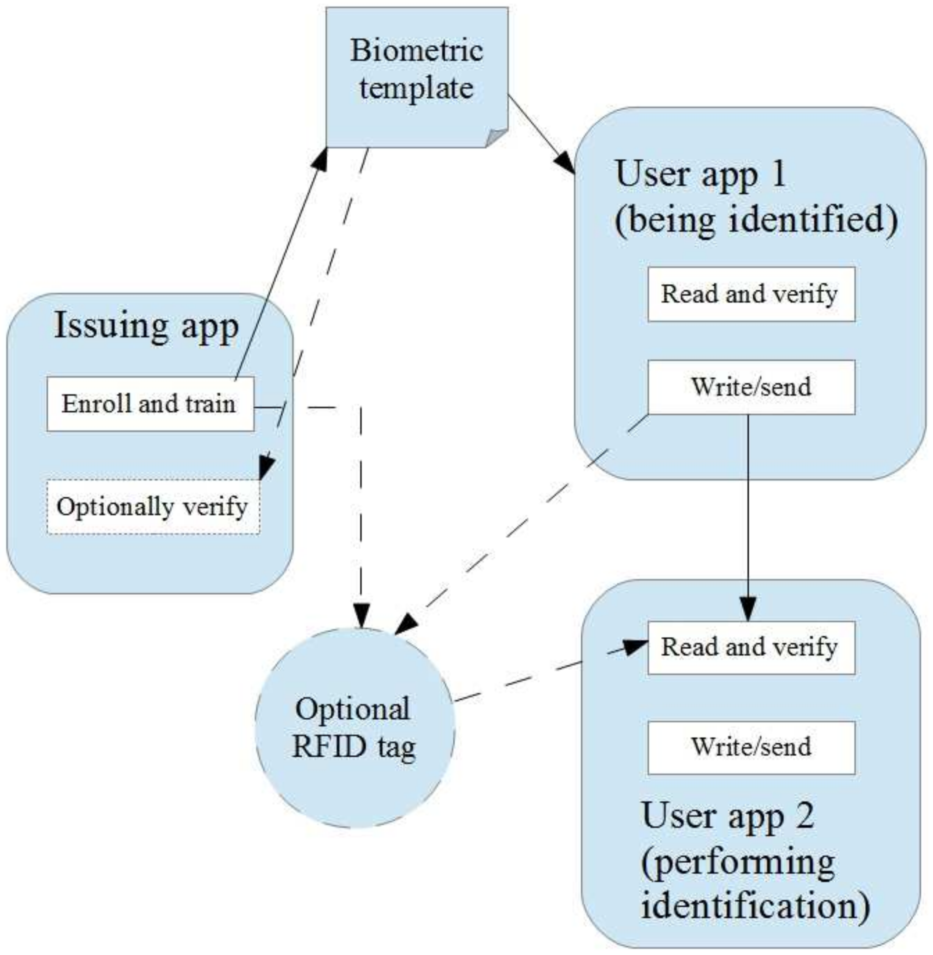


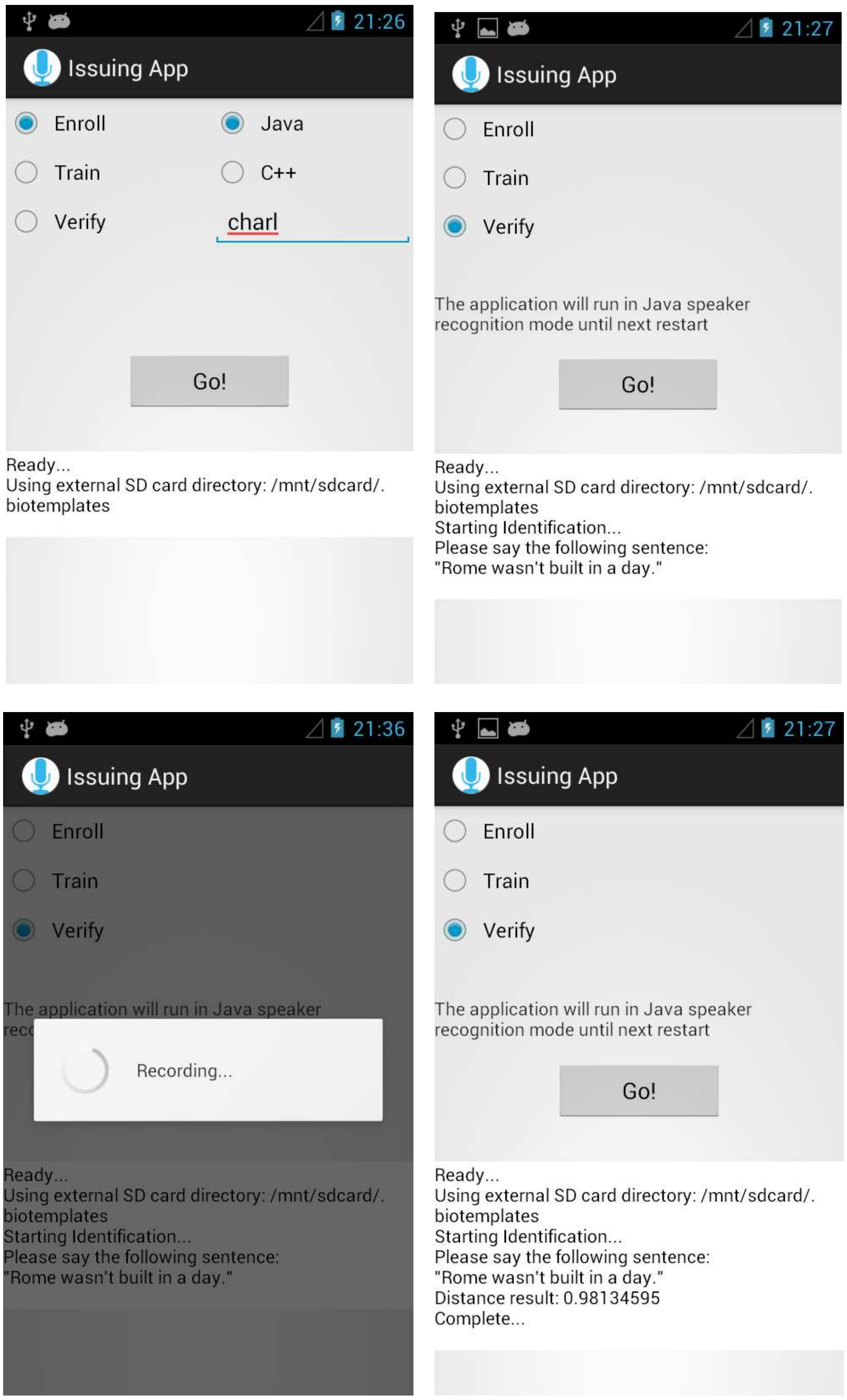
Figure 3.5. A block diagram of the NFC-enabled biometric system

3.4.1 Issuing application

The issuing application was tested under various configurations to study the effects of different variables on the processing time and accuracy of the speaker recognition implementation. Firstly, a comparison was made between the processing time of native code and non-native code. The application provides a GUI with an optional choice between Java and C++, as shown in the screenshots of the application in Figure 3.6. PCM audio samples are directly obtained from the phone's microphone and the sample rate was set as both 8000 and 44100 samples per second for comparison. It should be noted that the sample rate does not affect the number of LPC coefficients that is extracted from the audio for subsequent classification, but merely allows a larger range of frequencies to be present in the original recording (it also adversely affects the time it takes to extract the LPC coefficients from the audio sample). The number of extracted LPC coefficients represents the size of the feature vector, which is finally stored on the phone or on an RFID tag. The feature vector size was varied to fit the storage capacity of different RFID tags and the effect on accuracy, processing time, and also NFC transmission time was studied. Lastly, the effect of text-independence on accuracy was also studied for intruders and the legitimate user by performing verification of users when they record a different sentence than the sentence which was used for enrolment. The interface of the application instructs the user on the sentence to speak as can be seen in Figure 3.6.

3.4.1.1 Functionality

A flow diagram of the issuing application is given in Figure 3.7. The user selects either native or non-native mode and then initiates enrolment, training or verification for the given username. In enrolment mode the user's voice is recorded and LPC features are extracted and stored on the SD card of the phone. Any number of feature vectors may be generated before performing system training, which averages the vectors and stores the result as the biometric template of the person. In verification mode the user's voice is recorded once again and the LPC coefficients are extracted. The Euclidean distance is then computed between the extracted features and the stored template. If the distance exceeds a certain threshold the person is classified as an imposter. Verification in the issuing application is provided only for the purpose of testing the enrolment and training process. Actual identification of persons occurs by means of the user application.



[✓ Enroll] Train Verify
[✓ Java] C++
charl

Ready...
Using external SD card directory: /mnt/sdcard/.
biotemplates

Enroll Train [✓ Verify]

The application will run in Java speaker recognition mode until next restart

Ready...
Using external SD card directory: /mnt/sdcard/.
biotemplates
Starting Identification...
Please say the following sentence:
"Rome wasn't built in a day."

Enroll Train [✓ Verify]

The application will run in Java speaker recognition mode until next restart

Recording...

Ready...
Using external SD card directory: /mnt/sdcard/.
biotemplates
Starting Identification...
Please say the following sentence:
"Rome wasn't built in a day."

Enroll Train [✓ Verify]

The application will run in Java speaker recognition mode until next restart

Ready...
Using external SD card directory: /mnt/sdcard/.
biotemplates
Starting Identification...
Please say the following sentence:
"Rome wasn't built in a day."
Distance result: 0.98134595
Complete...

Figure 3.6. Screenshots of the issuing application's GUI

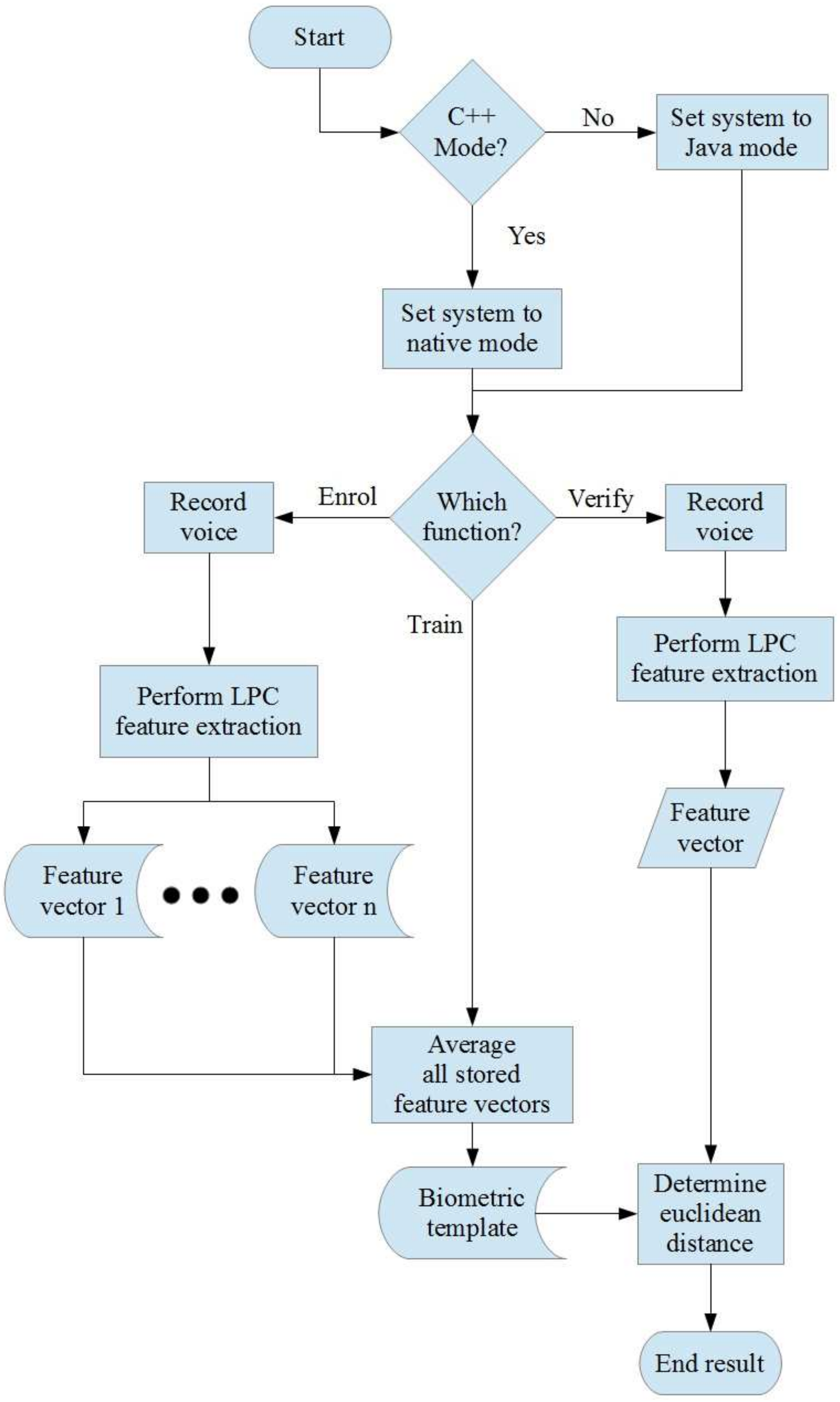


Figure 3.7. A flow diagram of the issuing application

LPC was used as the feature extraction algorithm because it provided a good trade-off between processing time and identification accuracy in the first experiment using MARF. LPC is a method for audio encoding that represents the current audio sample as a linear function of past values [53]. Figures 3.8 and 3.9 show the original speech sample and the extracted LPC coefficients, respectively, for the sentence “Rome wasn’t built in a day”. After feature extraction, the LPC data are stored to a comma-separated values (CSV) file on the SD card of the phone, for later classifier training. Training is performed by simply taking an average of all the LPC coefficients obtained from previous enrolments, on a per user basis. To classify a user, LPC coefficients are obtained for the spoken sentence, and then compared to the averaged LPC coefficients of the claimed identity, by determining the Euclidean distance between the vectors.

3.4.1.2 Library porting

Various open-source libraries were considered to be ported to Android for feature extraction and classification. Among them were LibXtract, Shogun [54], Marsyas, SPro in C++ and Weka [55], jAudio in Java. Although all libraries that were ported to Android were not implemented in the benchmarking applications, the porting process itself is of importance because it shows that state-of-the-art libraries for desktop operating systems can be successfully ported to mobile platforms for data processing. All of these libraries were not rigorously tested for bugs and further research into the consistency of the porting process is essential.

Weka is a popular machine learning and data mining library in Java. Porting of Weka required the removal of several classes that depend on GUI and standard Java libraries similar to the process of porting the MARF library. Similar porting of Weka to Android has also been attempted by other developers. For audio feature extraction in Java the jAudio library was used. Porting of jAudio did not induce much overhead.

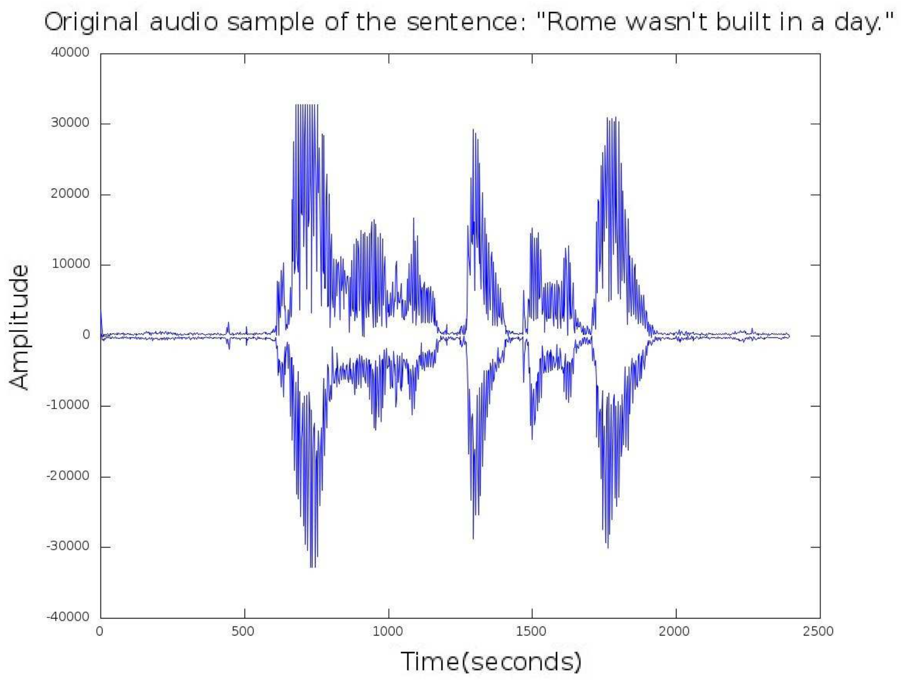


Figure 3.8. An audio sample of the sentence “Rome wasn’t built in a day” being spoken

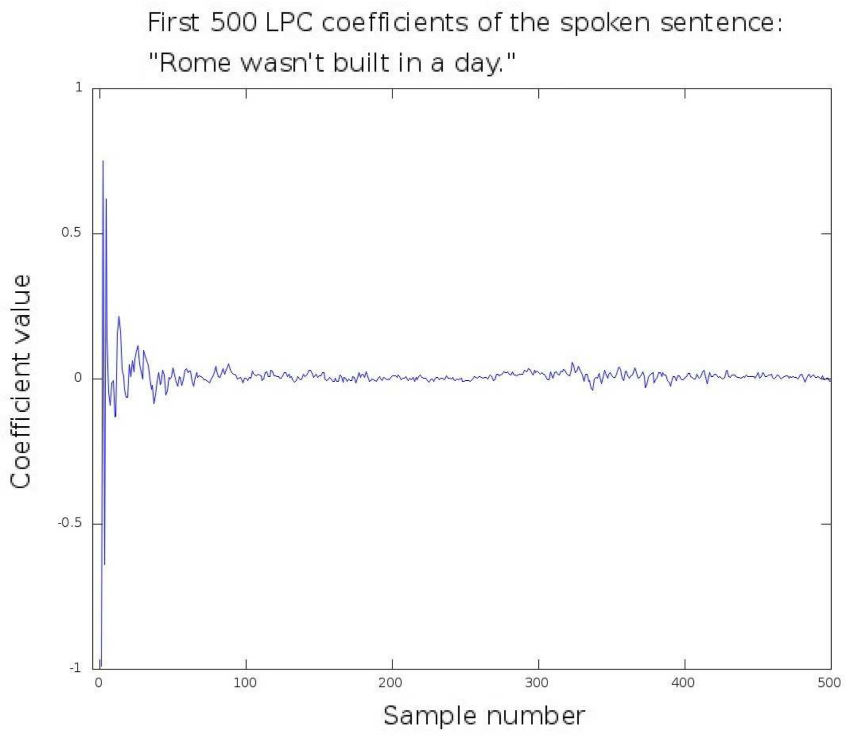


Figure 3.9. The first 500 LPC coefficients of the audio sample in Figure 3.8

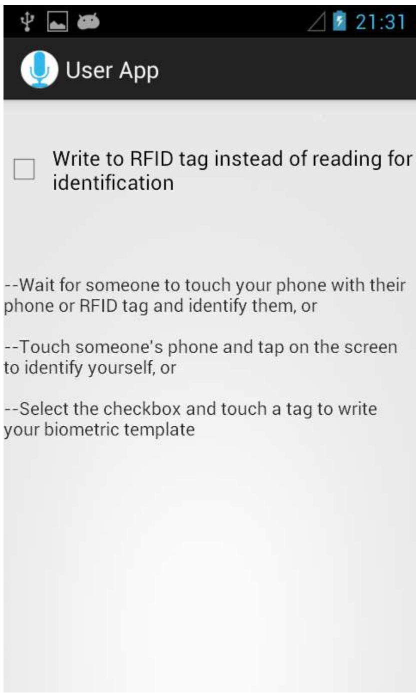
The porting of C and C++ libraries to the Android NDK is much more involved than the simple removal of non-functional code in Java. Libraries that are aimed at the Linux operating system were utilized (Android is based on Linux). These libraries often use build systems such as CMake or the GNU build system. To compile the source code of these libraries without the build system poses many problems, such as missing “include files” (the build systems add “include files” for configuration tasks during the build process). This means that in general the source files cannot be simply added directly to a native Android project. Instead a standalone GNU compiler collection (GCC) toolchain can be built for the Android NDK. When running the “configure” script of the GNU build system, this toolchain is manually linked to the build process and cross-compilation is performed to create ARM-compatible libraries, which can then be linked into the Android native project.

In C++, a popular library for large-scale machine learning is Shogun. Shogun was ported to Android with various issues of incompatibility with standard C header files such as “stdlib.h”, as well as differences in the standard “pthread.h” library for multithreading. These issues were solved by removing and replacing code in Shogun. Shogun also has a dependence on the fast automatically tuned linear algebra software (ATLAS) math library for many of the shogun features to function properly. Unfortunately ATLAS could not be ported because the library “tunes” itself on the platform on which it is compiled and cross-compilation is therefore not possible. The possibility might arise in the future to perform source code compilation on an Android phone itself, which could spawn new areas of research. Other dependencies of Shogun include LibSVM and SVMLight which are easily ported as part of the Shogun library. These libraries are used for support vector machines in Shogun.

LibXtract, Marsyas and SPro are all C or C++ feature extraction libraries that were considered. Porting of these libraries required no more effort than the porting of Shogun. LibXtract did however require a dependency library, FFTW, to be ported as well. The FFTW library is used for the computation of the fast fourier transform. The jAudio and SPro libraries were finally implemented for audio feature extraction in Java and C, respectively. For classification, the Euclidean distance algorithm was simply programmed using the standard math libraries in C and Java, which are available in Android.

3.4.2 User application

The user application allows the user to pass his or her biometric template to another phone for identification or write the template to an RFID tag, or to receive a template and then record a person’s voice to verify the person’s identity. Biometric templates are not generated by the user application. Figure 3.10 gives a screenshot of the application. As seen in the figure, there are very few interactive widgets on the screen due to the fact that mainly the touch interaction of NFC is required. Android’s tag dispatch system was used to allow the application to run automatically whenever an NDEF message is received with the appropriate application record. Having the application run automatically provides an extra sense of user-friendliness as well as increasing the overall promptness of the process.



☐ Write to RFID tag instead of reading for identification

--Wait for someone to touch your phone with their phone or RFID tag and identify them, or

--Touch someone's phone and tap on the screen to identify yourself, or

--Select the checkbox and touch a tag to write your biometric template

Figure 3.10. A screenshot of the user application’s GUI

RFID tags can also be formatted as NDEF tags with the appropriate application record, which is intercepted by Android applications' intent filters to determine whether to run automatically. When non-NDEF formatted tags are detected by the application, they are automatically formatted and the user's biometric template is written to the tag.

3.4.2.1 Functionality

A flow diagram of the user application is given in Figure 3.11. The application can be started either manually or automatically using Android's tag dispatch feature. The dispatch system allows applications to be started when certain intents are raised on the phone. Examples include when a text editor application runs automatically when a text file is opened or a media player application opens when a video file is clicked by the user. These intents can be raised in different ways and Android allows the specification of intent-filters in the manifest of each application where certain criteria are specified to allow the application to launch automatically under specified conditions. Android specifies various intents for NFC events, such as when a certain NDEF record is present in a message or when certain types of RFID tags are touched with the phone. For this project, the application was allowed to run automatically when any tag is detected, in which case the tag would be formatted for NDEF messages and two MIME type records written to the tag. The first message specifies an application record to register the tag for dispatching the correct application and the second record contains the biometric data. When an NDEF tag is read or a peer-to-peer NDEF message is received containing these two records, the application starts automatically and reads the biometric data for verification. As seen in the flow diagram, when the application is launched by the tag dispatch system, the accompanying biometric data in the NDEF message is immediately fed into the application without any user interaction. Another option to allow automatic launch of the application is to use a custom NDEF record provided by Android called an Android application record (AAR). This allows normal intent filters to be overridden to ensure that the correct application is launched when an NDEF message is received with the AAR.

When the application is user-activated, the application waits until a phone or RFID tag is touched. An RFID tag will either be written with the biometric data on the phone or data

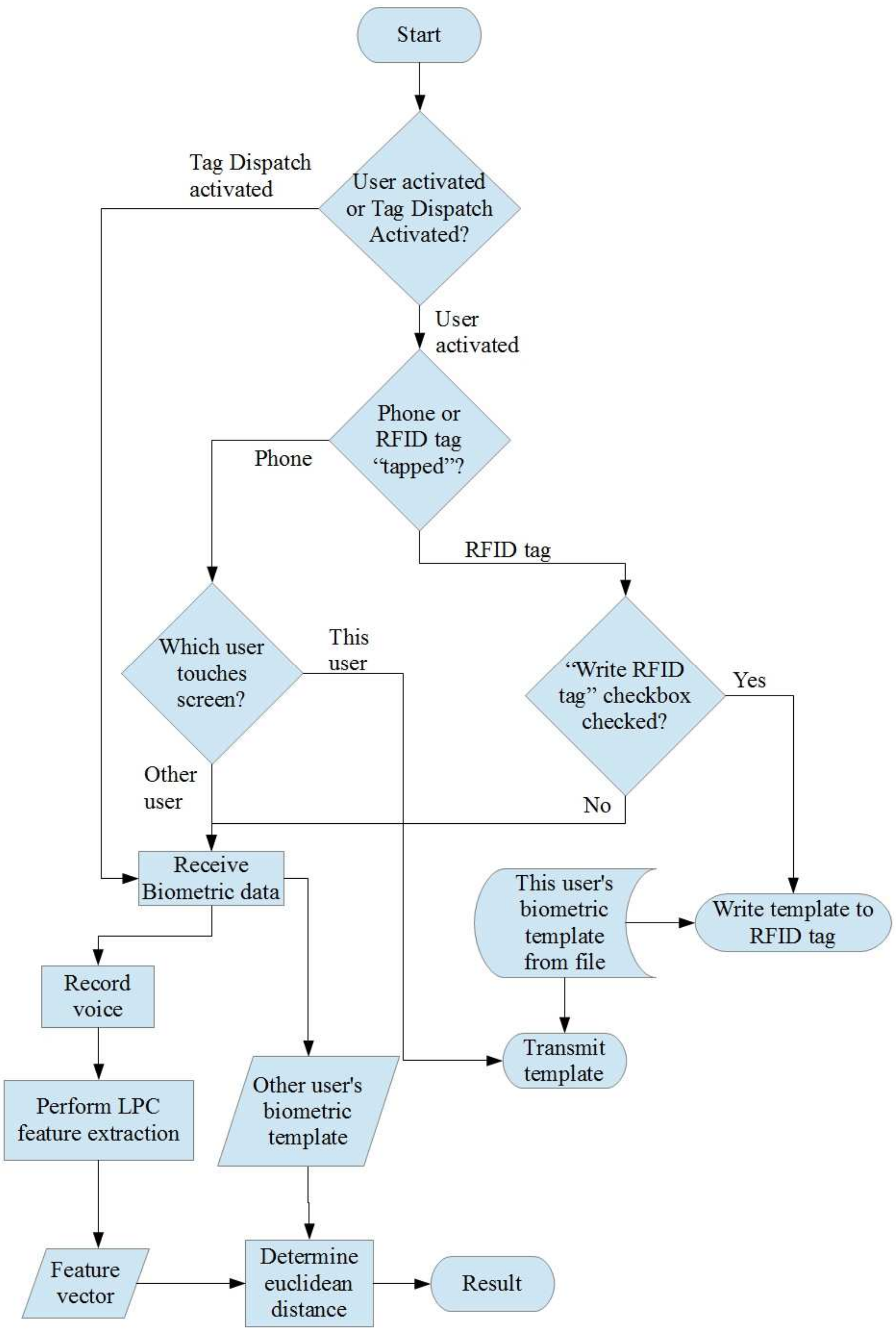

Start
User activated or Tag Dispatch Activated?
Tag Dispatch activated
User activated
Phone or RFID tag “tapped”?
Phone
RFID tag
Which user touches screen?
This user
Other user
“Write RFID tag” checkbox checked?
Yes
No
Receive Biometric data
This user's biometric template from file
Write template to RFID tag
Record voice
Transmit template
Perform LPC feature extraction
Other user's biometric template
Feature vector
Determine euclidean distance
Result

Figure 3.11. A flow diagram of the user application

will be read from the tag and verification performed, depending on whether the checkbox on the user interface is checked. If another phone is touched, Android notifies that another device is in range and the user taps on the screen to send the NDEF message containing his or her biometric data and the application record that allows the other user's app to run automatically. The other user then records the instigator's voice to verify the biometric template that was received in the same manner as was done with the issuing application.

Three popular RFID tags were studied in addition to normal peer-to-peer transmission. The MIFARE Ultralight (NFC Type 2 Tag), MIFARE Classic 1K and the MIFARE DESFire (NFC Type 4 Tag) were studied, with storage capacities of 48 bytes, 1 kilobyte, and 4 kilobytes, respectively. NDEF formatting of tags reduce the overall storage sizes to 46 bytes, 716 bytes, and 4094 bytes, respectively. In this space, two NDEF records have to be stored, firstly for the application record allowing tag dispatch and secondly for the custom payload record which contains the required biometric data. Due to the very limited storage capacity of the MIFARE Ultralight tag, it was not formatted for NDEF messages, but instead it was analysed as if biometric data were directly stored on it. The LPC samples that represent the biometric data were formatted as floating point numbers with 4 decimal digits and this was converted to a fixed-point data format using two bytes for each sample. Leaving a few bytes open on each tag for future improvements such as a digital signature, and subtracting the space used for the application record on the MIFARE Classic and the MIFARE DESFire only, the number of LPC samples that can be stored on each tag is 20, 325, 2000, respectively. The same number of LPC samples was also transmitted using peer-to-peer mode to strike a comparison. By studying the effect of different numbers of LPC samples on accuracy and transmission time, a comparison is made between the feasibility of applying different RFID tags for mobile biometrics.

To examine the accuracy of the biometric implementation, the voices of a legitimate user and several intruders were recorded under different conditions. The system was trained with the voice of the legitimate user recorded 5 times when saying the phrase "Rome wasn't built in a day". The voices of 4 intruders were then recorded to compare the relative Euclidean distances between the training set and the recorded template. The voice of the male legitimate user was compared with one female intruder and three male intruders. More male intruders were involved in the study since the likelihood of a false match with a male voice was more likely. Recognition of each intruder was performed several times while changing variables such as feature vector length, recording sample rate, and the sentence spoken. The feature vector length was varied between 20, 325, and 2000 samples for compatibility with three different NFC tags. The NFC transmission time was observed in addition to the change in system accuracy. The effect of sample rate on system accuracy was studied by changing the sample rate from 8000 Hz to 44100 Hz. Lastly the effect of porting Java code to native code on processing time was studied.

CHAPTER 4 RESULTS

4.1 FIRST EXPERIMENT RESULTS

The program was run with the aim of logging the processing time and the accuracy of the system, for comparison between a PC and a smartphone and for later comparison with other biometric implementations. In training mode, the only variable to analyse was the processing time needed to train the system. In testing mode, the processing time was important, as well as the actual accuracy of the system. The accuracy is however highly dependent on the quality of the data classified by the system, and is therefore in reality a function of various quantities such as sample rate, SNR, distance from mouth to microphone, preprocessing algorithm, etc. However, when testing the system on a very large database, with varying conditions and quality, a good estimate can be obtained.

4.1.1 Training results

The training program was developed to log the processing time for each combination of algorithms in milliseconds. The program was executed on the PC and Table 4.1 gives a summary of the results measured. The training times given in Table 4.1 include the loading of audio samples, preprocessing, feature extraction, training data (and user templates) storage, and actual training of the classifiers (if required). For the most part, Table 4.1 gives these time values for the entire training set (all 293 samples in total), unless otherwise specified. It should be noted that the distance-based classifiers do not require training as in the sense of neural network training, where specific weights are stored for branches between neurons. For distance-based classifiers, the training data is preprocessed and feature vectors are generated and stored for use when the classifiers are used in testing or identification mode.

The average time values given in Table 4.1 for each preprocessing method, is the average time for the entire MARF pipeline, when different feature extraction and classification methods are combined with the specific preprocessing module. In this way a decision can be made on which methods train the longest and quickest on average. The same goes for the feature extraction and classification time values given in Table 4.1. It should be noted however, that when the statement is made that a certain feature extraction module is “slow”, the real reason for the large time value might be that the module does not reduce the dimensionality of the vector enough, and this causes the classifier to train for a long time, so it may not be the feature extraction module itself that is slow, but it results in slow classifier training or something similar.

Following this convention, the first conclusion that can be drawn is that Raw preprocessing (or no preprocessing) is the quickest preprocessing method, which is somewhat intuitive, since there is no preprocessing performed. The downside to Raw and Dummy preprocessing is that noisy feature vectors are passed to the feature extractors, which causes the classifiers to be trained with more random data. The slowest preprocessing method (or the method that results in the slowest training process) is the High-Pass-Boost FFT filter, which is a combination of two preprocessing techniques. The High-Boost filter by itself is quite slow compared to the High-Pass filter. For the feature extractors, the Random feature extractor causes a very slow average training time. From the actual data, it was clear that the Neural Network classifier trained very slowly when Random feature extraction was used, and this fact was mostly responsible for the large average training time of the Random feature extractor. The FFT feature extraction module resulted in very low training time values, but all attempts to train a neural network failed when using the FFT module and this has a considerable effect on the average training time.

When looking at the classifiers, it can be seen that the training time for Random classification is very low, which is very intuitive. The processing time for some of the distance-based methods is very low as well, because they do not require any training. They do however require the storage of training data. The Neural Network training time is very high in relation to the other classifiers. The Neural Network also caused the most errors in training. The error that occurred most regularly when training the network was an “out of memory” error. In an attempt to solve the problem, more memory was allocated to the JVM, but the problem remained. To avoid these errors the size of the network may be decreased by decreasing the number of neurons and possibly removing the single hidden layer which was used.

Table 4.1. Summary of the measured processing times for the training of various algorithms on the PC implementation. Time values are given per combination of algorithms unless specified as "per sample" and the values are for the processing of the entire MARF pipeline

General processing time measurements	**Training time**
Total for all algorithm combinations and all training samples	16 hours and 7 minutes
Average per sample	753 ms
Average per algorithm combination	221.39 seconds
Measurements based on preprocessing methods	
Average with no preprocessing (Raw)	62.72 seconds
Average with only normalisation (Dummy)	186.79 seconds
Average with endpoint preprocessing	97.43 seconds
Average with High-Boost FFT filter	482.88 seconds
Average with Band-Pass FFT filter	137.19 seconds
Average with Low-Pass FFT filter	148.63 seconds
Average with High-Pass FFT filter	143.96 seconds
Average with High-Pass-Boost FFT filter	504.47 seconds
Measurements based on Feature Extraction Methods	
Average with Random feature extraction	739.87 seconds
Average with LPC algorithm	57.57 seconds
Average with FFT algorithm	44.59 seconds
Average with Min/Max algorithm	160.86 seconds
Average with aggregation of LPC and FFT algorithms	62.51 seconds
Measurements based on Classification Methods	
Average with Random classification	26.67 seconds
Average with Neural Network	1994.88 seconds
Average with Euclidean Distance Classifier	96.35 seconds
Average with Manhattan Distance Classifier	96.37 seconds

Average with Minkowski Distance Classifier	28.84 seconds
Average with Mahalanobis Distance Classifier	27.82 seconds
Average with Diff-Distance Classifier	27.64 seconds

The following list gives some additional observations made from the training time data.

- The combination of algorithms that resulted in the quickest training was the Endpoint preprocessor, the Min/Max feature extractor and the Random classifier (about 2.31 seconds in total or 7.89 ms per audio sample). On the smartphone implementation, this combination resulted in a training time of 77.64 seconds in total or 265 ms per audio sample. That is about 33.6 times longer on the phone than on the PC. The Random classifier is however not useful in an actual system.
- The second quickest combination and the quickest useful combination was the combination of the Endpoint preprocessor, the Min/Max feature extractor, and the Diff-Distance classifier at about 2.59 seconds in total or 8.83 ms per audio sample. On the smartphone implementation, this combination resulted in a training time of 98.94 seconds in total or 337.7 ms per audio sample. That is about 38.2 times longer on the phone than on the PC.
- The slowest combination was the High-Pass-Boost FFT preprocessor, with Random feature extraction and the Neural Network classifier. The total training time was about 227 minutes (3 hours and 47 minutes) in total or 46.48 seconds per audio sample. Based on an estimate that the phone trains about 36 times longer than the PC, it can be estimated that the phone would train on this combination for about 8172 minutes or 136 hours and 12 minutes in total or 27.89 minutes per audio sample.

The reason for comparing the training time of the PC implementation of the speaker recognition system with the implementation on a smartphone, is to determine whether the system will allow for the addition of new users to the system on a smartphone, and whether the training can be performed on the phone itself, or whether it would need to be delegated to a more powerful computer. It was found that the training time of some algorithms was too slow to be feasible even on the PC, much less on the phone. Most combinations are however quite feasible on the phone.

Holding to the convention that the phone trains about 36 times longer than the PC it can then be estimated from Table 4.1, that the total training time for the phone (all combinations on all samples) would be more than 580 hours (about 24 days), which is why the phone was not trained exhaustively using all algorithms. It can also be estimated (from Table 4.1) that the average training time per audio sample would be about 27.1 seconds, which is very reasonable, because a user would only train the system with his or her own audio samples in most cases. Many of the algorithms provide training times (per audio clip) of much less than 27.1 seconds in any case, and only one or a few of them, which perform accurately in identification mode would probably be considered for a final biometric implementation, in which case both training and identification would be possible (and comfortable) on the phone.

4.1.2 Testing results

The results that were measured and logged in identification mode were both timing values and accuracy values. Table 4.2 gives a summary of the processing time values measured for different combinations of algorithms on both the PC and the Android implementations. As was mentioned in Section 3.2, there were some algorithms that did not execute successfully on the smartphone (High-Freq-Boost-FFT preprocessor, Aggregator feature extractor and Neural Network classifier), and consequently they are not included in the results given in Table 4.2, even though they executed successfully (for the most part) on the PC. These values are not directly comparable to the timing results measured for training that were given in Table 4.1, because training was performed on more than 10 times the number of samples that were used for testing. The results are however useful for comparing the time needed for identification by the various algorithms and also to compare these times between the PC implementation and the Android smartphone implementation. On average, the PC implementation performed about 30 times faster than the smartphone implementation.

Under the preprocessing section in Table 4.2, the Dummy implementation was quite slow in training and is again very slow in testing, whereas the Lowpass FFT filter performs relatively quickly. Under the feature extraction section, the LPC and Min/Max algorithms are very quick whereas the FFT algorithm is now very slow. As for the classifiers, the Neural Network classifier increased heavily in speed in testing mode as opposed to training mode (although it is not shown in Table 4.2 because it did not execute successfully on the smartphone). It was however still not quite as quick as the distance-based classifiers, but this may be partly due to the fact that the Neural Network's structure is loaded from file storage upon identification. The Mahalanobis Distance classifier is very slow in identification mode and this is probably due to the specific distance formula used, which contains matrix manipulations.

To measure the accuracy of each algorithm combination, the following approach was followed, the FRR could be determined from the identification rate (IR) as in (4).

$$IR = \frac{n_c}{n_t} = \frac{n_t - n_{ic}}{n_t} = 1 - \frac{n_{ic}}{n_t} = 1 - FRR \qquad (4)$$

where n_c is the number of correct classifications,

n_{ic} is the number of incorrect classifications, and

n_t is the total number of classifications (29 in this case).

In this case the FRR is taken as any incorrect classification, even if the resulting classification is still a legitimate user in the database. This approach was taken for the calculation of the FRR because the final implementation of such a system would be a verification system, in which the user “logging in” supplies a user ID and classification as another legitimate user would therefore not be possible. The resulting classification will either be positive (the user is who he or she claims to be) or negative (the user is not who he or she claims to be).

The resulting FRR values for several of the best algorithm combinations are given in Table 4.3, arranged from best to worst. The recognition accuracy of these combinations has also been tested by the developers of the MARF library. From the values given in the MARF manual [48], the FRR values were determined and compared with the values determined in this experiment, as given in Table 4.3. They were found to be equivalent for the most part. Table 4.3 does not include the algorithms that failed to execute on the smartphone, even when they performed well on the PC. The fact that the Aggregator feature extraction module could not execute on the smartphone was quite unfortunate, because there were many combinations in the top performing algorithms that used an aggregation of the FFT and LPC feature extraction modules.

Table 4.2. Summary of the measured processing times for the identification of 29 speech samples using all possible combinations of algorithms on both the PC and the smartphone implementations, excluding the algorithms which did not execute successfully on the smartphone

General processing time measurements	**Testing Time (PC implementation)**	**Testing Time (Smartphone implementation)**
Total for all algorithm combinations and all testing samples	2 hours and 17 minutes	68 hours and 37 minutes
Average per sample	1.73 seconds	51.94 seconds
Average per algorithm combination	50.24 seconds	1506.15 seconds
Measurements based on preprocessing methods		
Average with no preprocessing (Raw)	95.15 seconds	1862.92 seconds
Average with only normalisation (Dummy)	59.41seconds	1633.19 seconds
Average with endpoint preprocessing	43.25 seconds	1617.98 seconds
Average with Band-Pass FFT filter	48.86 seconds	1687.40 seconds
Average with Low-Pass FFT filter	39.64 seconds	1451.48 seconds
Average with High-Pass FFT filter	48.76 seconds	1684.69 seconds
Average with High-Pass-Boost FFT filter	48.00 seconds	1739.53 seconds
Measurements based on Feature Extraction		

Methods		
Average with Random feature extraction	17.49 seconds	772.69 seconds
Average with LPC algorithm	10.96 seconds	233.39 seconds
Average with FFT algorithm	142.68 seconds	5044.47 seconds
Average with Min/Max algorithm	33.04 seconds	91.38 seconds
Measurements based on Classification Methods		
Average with Random classification	10.06 seconds	100.42 seconds
Average with Euclidean Distance Classifier	10.08 seconds	103.47 seconds
Average with Manhattan Distance Classifier	9.97 seconds	102.89 seconds
Average with Minkowski Distance Classifier	10.78 seconds	113.14 seconds
Average with Mahalanobis Distance Classifier	253.26 seconds	8618.66 seconds
Average with Diff-Distance Classifier	10.31 seconds	103.06 seconds

As a baseline for these values, the FRR was also measured for the implementation of the Dummy preprocessing module, the Random feature extractor, and the Random classifier. The result was an FRR of 89.66%. According to the MARF manual, the FRR of this baseline combination of algorithms is about 96.55%, but differences in these values are expected because the feature extraction and classification modules are both random.
It is quite interesting to note that the Dummy and the Raw preprocessing methods are quite prevalent in the top performing algorithms in Table 4.3. According to the authors of the MARF manual, the reason for the good performance of the Raw preprocessing module is that the module has an “unfair advantage” over the other preprocessing modules because it does not perform normalisation [48].

Another statement of interest in the MARF manual was that in the current version of MARF, the Mahalanobis Distance classifier is equivalent to the Euclidean Distance classifier, because the covariance matrix used by the Mahalanobis Distance classifier has

Table 4.3. FRR values of some of the most accurate algorithm combinations

Combination of algorithms	**FRR (%)**
Endpoint, LPC, Manhattan-Distance	17.24
Dummy, FFT, Diff-Distance	24.14
Raw, FFT, Euclidean-Distance	24.14
Dummy, FFT, Euclidean-Distance	27.59
Dummy, FFT, Manhattan-Distance	27.59
Endpoint, LPC, Euclidean-Distance	27.59
Raw, FFT, Manhattan-Distance	31.03
Dummy, FFT, Minkowski-Distance	34.48
Dummy, LPC, Euclidean-Distance	34.48
Dummy, LPC, Manhattan-Distance	34.48
Dummy, LPC, Minkowski-Distance	34.48
Dummy, LPC, Diff-Distance	34.48
Low-Pass-FFT, FFT, Euclidean-Distance	34.48
Low-Pass-FFT, FFT, Manhattan-Distance	34.48
Low-Pass-FFT, FFT, Diff-Distance	34.48
Raw, LPC, Euclidean-Distance	34.48
Raw, LPC, Manhattan-Distance	34.48
Raw, LPC, Minkowski-Distance	34.48
Raw, LPC, Diff-Distance	34.48
Endpoint, LPC, Minkowski-Distance	37.93
Raw, FFT, Diff-Distance	37.93
Low-Pass-FFT, FFT, Minkowski-Distance	41.38
High-Pass-FFT, FFT, Euclidean-Distance	44.83
High-Pass-FFT, FFT, Manhattan-Distance	44.83
High-Pass-FFT, FFT, Minkowski-Distance	44.83

Low-Pass-FFT, LPC, Manhattan-Distance	44.83
Low-Pass-FFT, LPC, Euclidean-Distance	48.28
High-Pass-FFT, LPC, Manhattan-Distance	51.72
Low-Pass-FFT, LPC, Minkowski-Distance	51.72
Low-Pass-FFT, LPC, Diff-Distance	51.72

been set to an identity matrix until further improvements have been made to the module [48]. It is quite clear from the testing results that these two distance-based classifiers are equivalent, except for the fact that the Mahalanobis Distance classifier executes much slower than the Euclidean Distance classifier in both training and testing. The Mahalanobis Distance classifier was therefore omitted from Table 4.3, and the “Euclidean-Distance” entry in the table represents both these classifiers.

4.2 SECOND EXPERIMENT RESULTS

Table 4.4 gives a summary of the processing time measured on the Nexus S phone as compared with the PC implementation. The values given in Table 4.4 include the loading of images, PCA feature extraction for all samples, and classification for 240 samples. For simplicity, the total processing time is simply divided by 1600 (4 classification algorithms times 400 images) to determine the processing time “per sample”. Training data were not stored at this stage and this will further increase the processing time when storage of these data is performed. The “absolute value” of the Manhattan distance classifier performs a bit quicker than the “square root of the sum of squares” that is used by the Euclidean distance classifier. The Minkowski distance classifier is the slowest but the difference is quite small between all of them.

It can be seen from Table 4.4 that the processing time on the phone was about 21 times as much as on the PC for this experiment. This is quite reasonable and it is clear from these values that a user-friendly, comfortable face recognition system should be viable on smartphones.

To measure the accuracy of each classification algorithm (combined with the Eigenface algorithm), (4) was again used to determine the FRR. Table 4.5 gives the resulting FRR values that were measured for each classification algorithm. These results are equivalent for the PC and phone implementation because the software was identical. The FRR is again taken as any incorrect classification, even if the incorrect classification still results in a legitimate user in the database being identified, as was discussed for the speaker recognition system.

On average these FRR values are much lower than those that were obtained with the MARF program, but considering that the MARF library was used for a text-independent system, with noisier data, it seems that the speaker recognition would probably be more promising in terms of accuracy. The Minkowski distance algorithm performed very well for r = *0.2*, as seen in Table 4.5.

Table 4.4. Summary of the measured processing times for the training of 160 images and classification of 240 images. For simplicity, there is no discrimination between training and testing times, and the average time per sample is therefore the average time for both training and testing

General processing time measurements	**Processing time (PC implementation)**	**Processing time (smartphone implementation)**
Total for all algorithms and all samples	11.174 seconds	3 minutes 54.48 seconds
Average per sample	6.98 ms	146.55 ms
Average per algorithm	2.79 seconds	58.62 seconds
Measurements based on classification methods		
Average with Euclidean Distance Classifier	2.784 seconds	57.605 seconds
Average with Manhattan Distance Classifier	2.745 seconds	57.404 seconds
Average with Minkowski Distance Classifier	2.892 seconds	61.843 seconds
Average with Diff-Distance Classifier	2.753 seconds	57.633 seconds

Table 4.5. FRR values for each classification method (combined with the Eigenface algorithm)

Classification Algorithm	**FRR (%)**
Euclidean Distance	19.67
Manhattan Distance	18.75
Minkowski Distance	10.83
Diff Distance	18.33

4.3 THIRD EXPERIMENT RESULTS

Although a small user database was used in the third experiment, rough visual inspections could still be made to observe the effects of various factors on identification accuracy and thereby judge the feasibility of the biometric application. In addition to determining whether the experiment delivered reasonable results in terms of accuracy, the NFC transmission time for each number of LPC samples to different tags and over peer-to-peer mode was also studied as well as the processing time required by the smartphone in each case. Table 4.6 gives the results of the measured transmission times for various feature vector lengths.

Figure 4.1 gives a visual representation of the distance values that were measured when verifying the legitimate user and various intruders for each feature vector length. The average legitimate and intruder scores also indicated on the figure and a wider distance between these two averages indicates a more accurate system since the legitimate and intruder scores are then further separated and overlap is less likely. As expected the larger feature vector size delivers more separation and therefore better accuracy. A trade-off needs to be made however since the larger feature vectors result in longer transmission times as seen in Table 4.6, especially since the feature vector length of 2000 results in a much longer transmission time than for a length of 325 but not in a much more accurate system. The results in figure 4.1 were measured with an audio sample rate of 8 kHz.

Table 4.6. NFC transmission for various feature vector sizes over peer-to-peer mode and to different RFID tags

NFC Mode	**Number of LPC samples**	**Transmission Time (ms)**
Peer-to-peer (Android Beam)	2000	3736
	500	2035
	325	1595
	20	1146
Mifare DESFire	2000	2277
Mifare Classic 1k	325	742
Mifare Ultralight	Not feasible. Foreground dispatch alone requires 43 bytes. Total NDEF size after format is 46 bytes. About 20 samples can be stored when no foreground dispatch is used however.	

4.3.1 Effect of sample rate

To study the effect of sample rate on system accuracy, the sample rate was changed from 8 kHz to 44.1 kHz and the results are displayed in figures 4.2, 4.3 and 4.4 for feature vector lengths of 325, 2000 and 20, respectively. It is clear from the figures that the sample rate did not have much of an effect on the system accuracy since the human voice does not generally extend to frequencies higher than 4 kHz. Higher sample rate does however result in longer processing time at the feature extraction stage since more audio samples are processed.

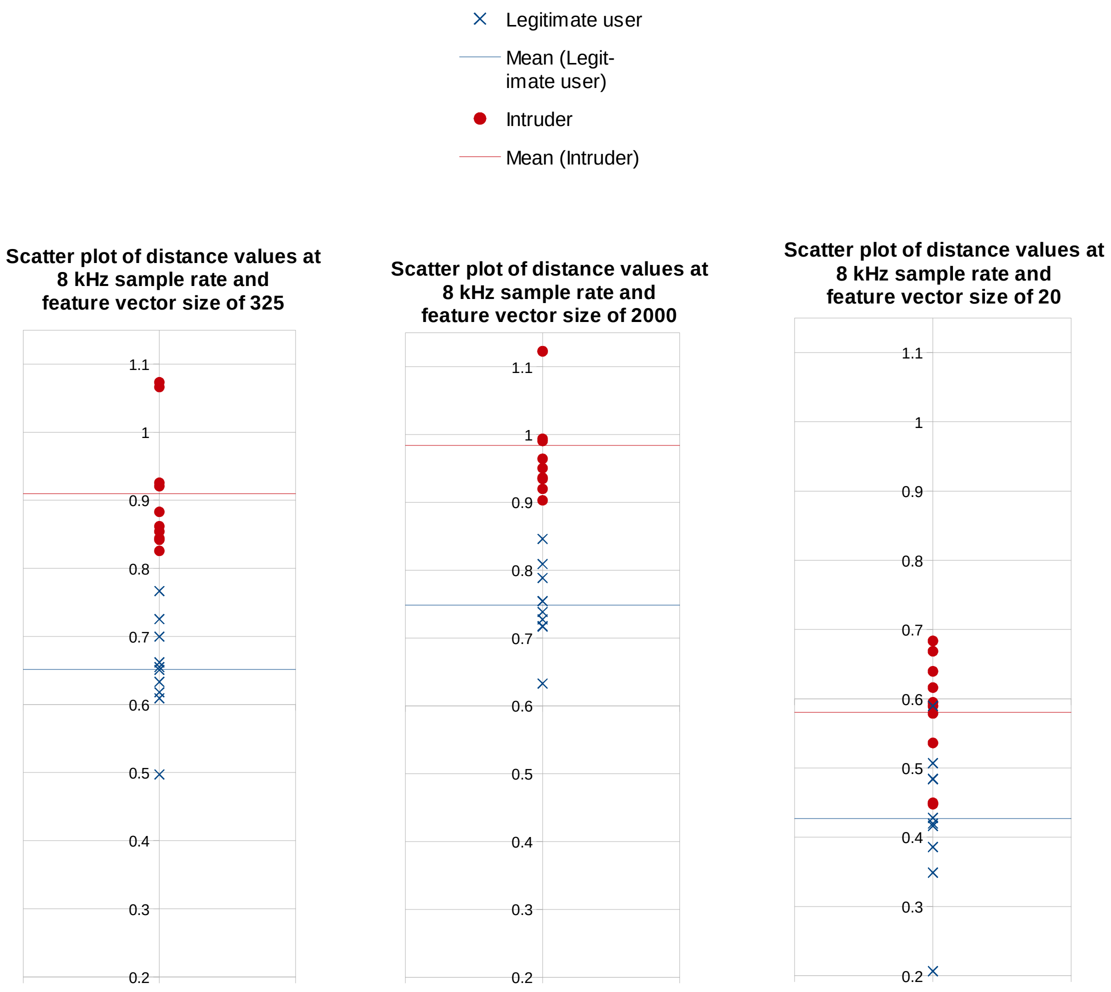


Figure 4.1. Comparison of the legitimate vs. intruder scatter plots for Mifare Classic 1K, Mifare DESFire, and Mifare Ultralight RFID tags when using an audio sample rate of 8 kHz

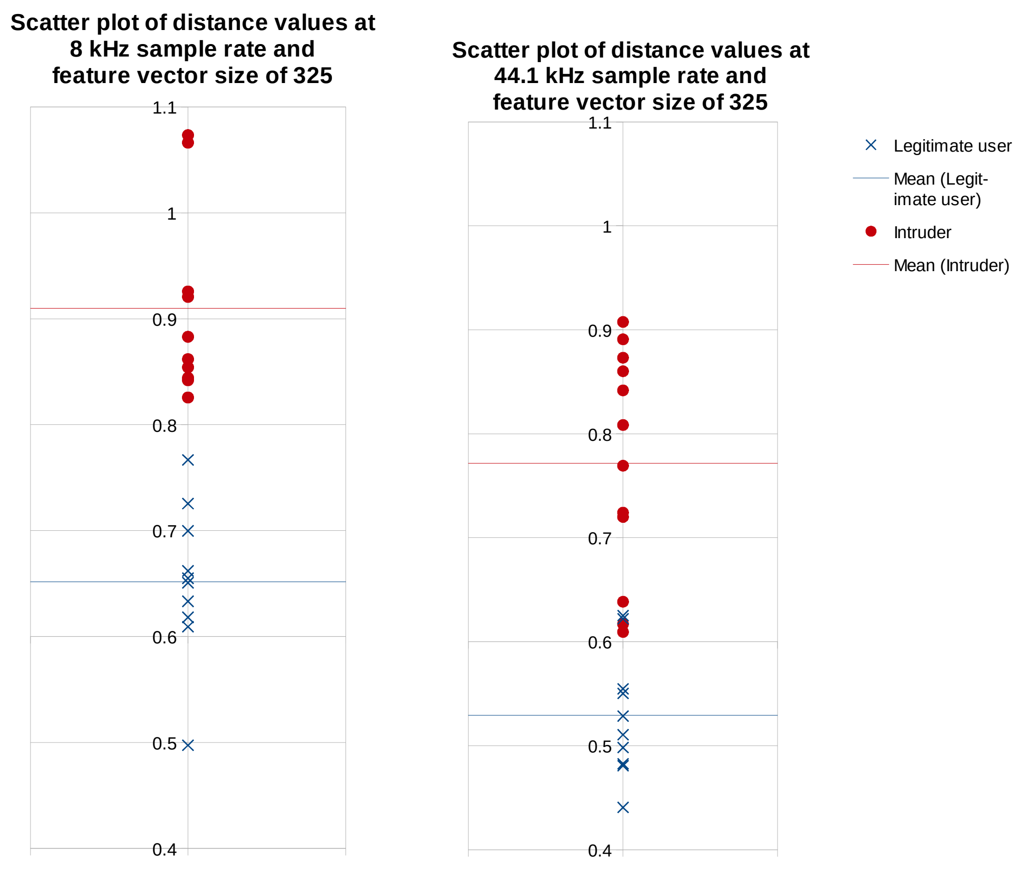


Figure 4.2. Comparison of the legitimate vs. intruder scatter plots for Mifare Classic 1K tags when using an audio sample rate of 8 kHz vs. a sample rate of 44.1 kHz

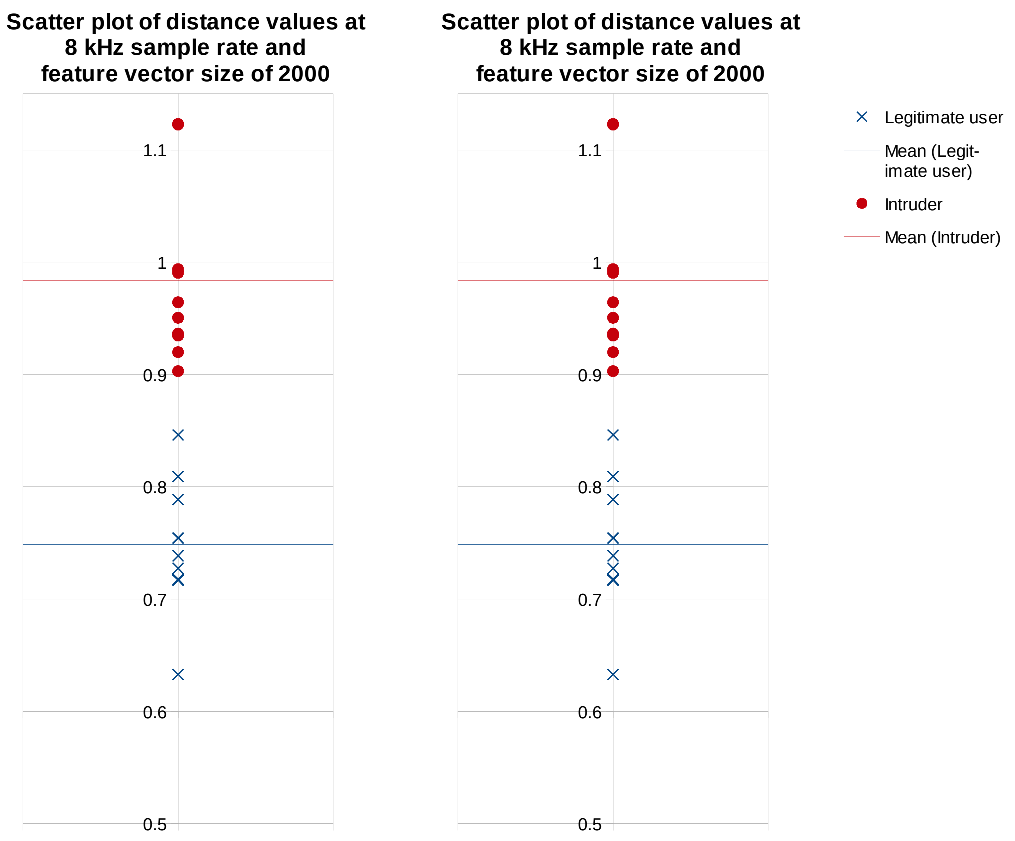


Figure 4.3. Comparison of the legitimate vs. intruder scatter plots for Mifare DESFire tags when using an audio sample rate of 8 kHz vs. a sample rate of 44.1 kHz

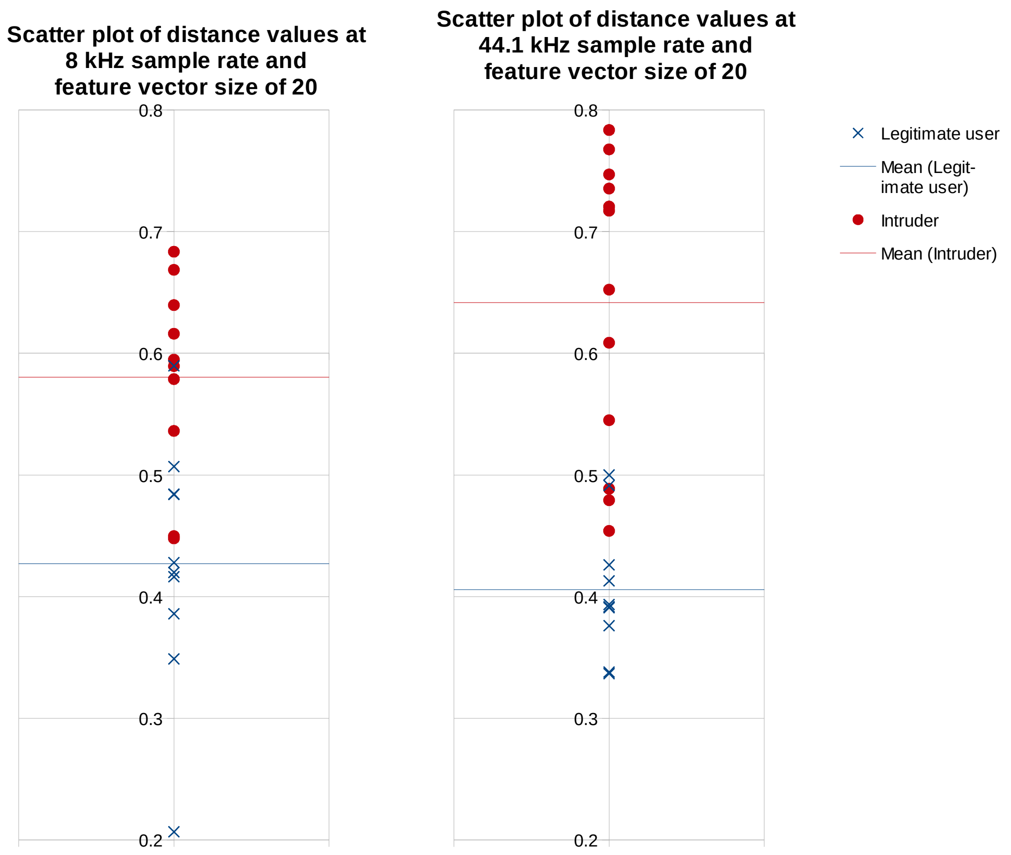


Figure 4.4. Comparison of the legitimate vs. intruder scatter plots for Mifare Ultralight tags when using an audio sample rate of 8 kHz vs. a sample rate of 44.1 kHz

4.3.2 Effect of text-independence

To study the effect of text independence on system accuracy, the legitimate user and the intruders were instructed to speak a different sentence to the one that was used by the legitimate user to enrol. The results are displayed in figures 4.5, 4.6 and 4.7 for feature vector sizes of 325, 2000 and 20, respectively. Each figure shows the original text-dependent results next to the measured text-independent results. It is clear that text dependence has a large influence on the results since more measurements for intruders and legitimate user start to overlap and the average distances become closer together. Text-independent systems are therefore not recommended for highly secure applications.

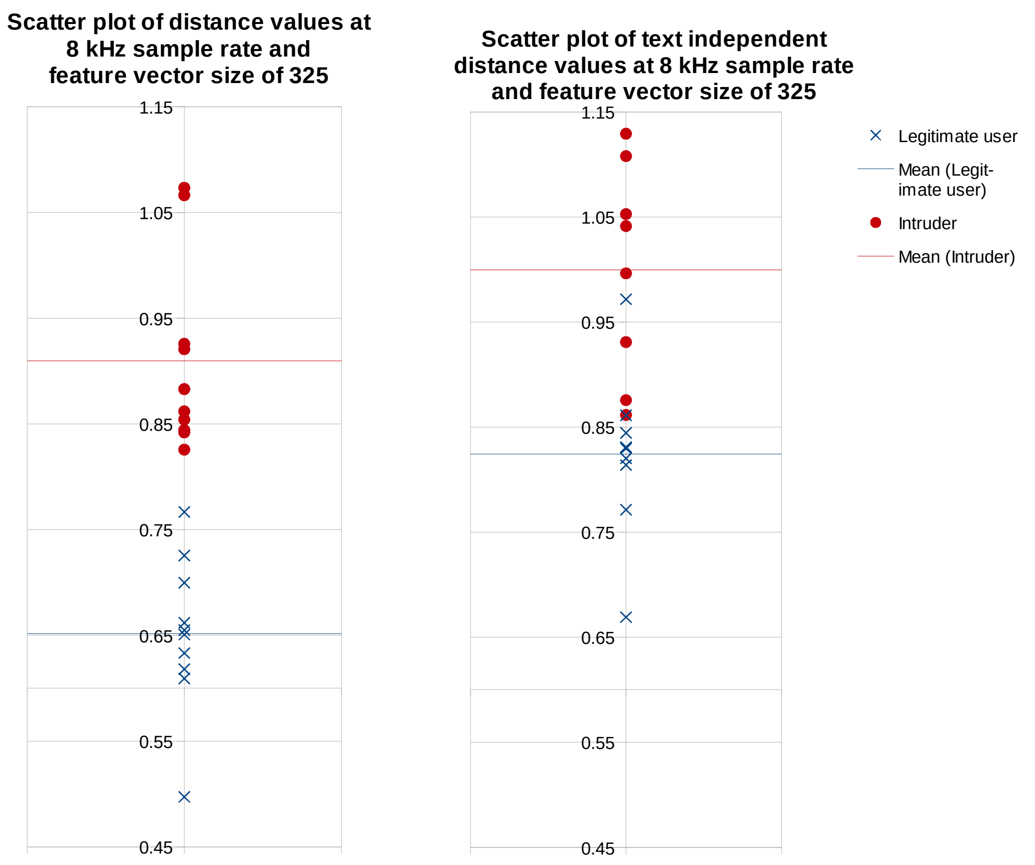


Figure 4.5. Comparison of the legitimate vs. intruder scatter plots for Mifare Classic 1k tags when using a text-dependent vs. text-independent system

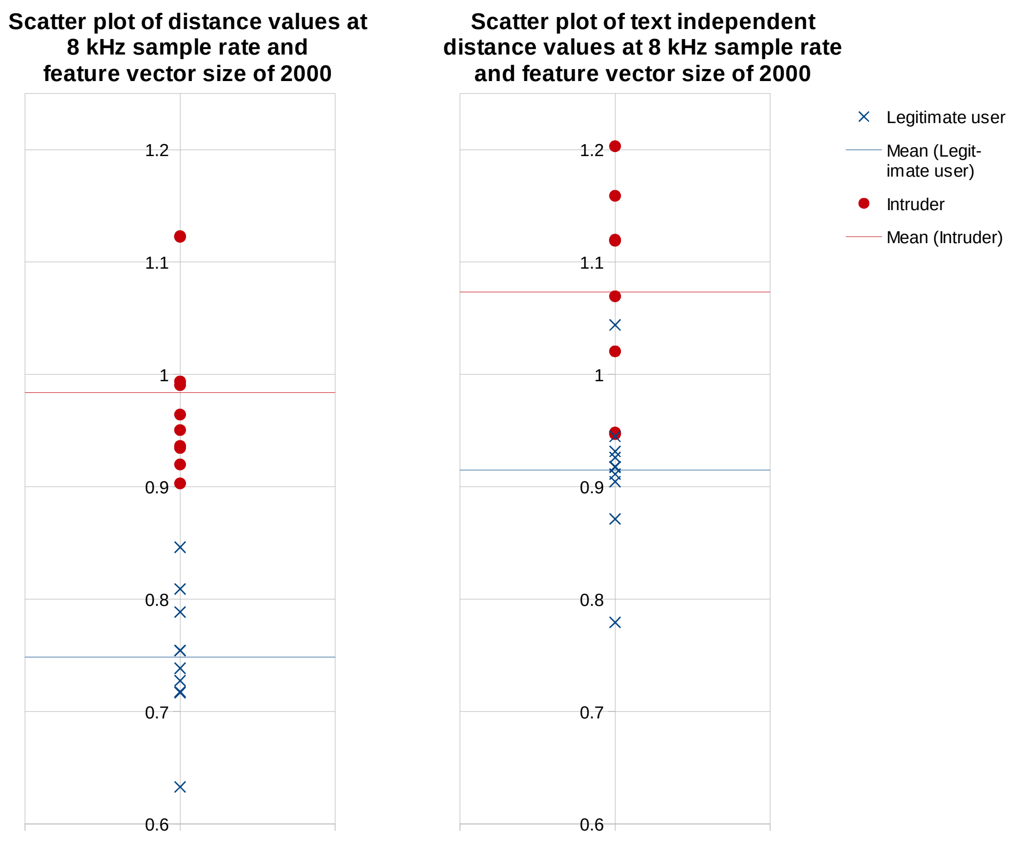


Figure 4.6. Comparison of the legitimate vs. intruder scatter plots for Mifare DESFire tags when using a text-dependent vs. text-independent system

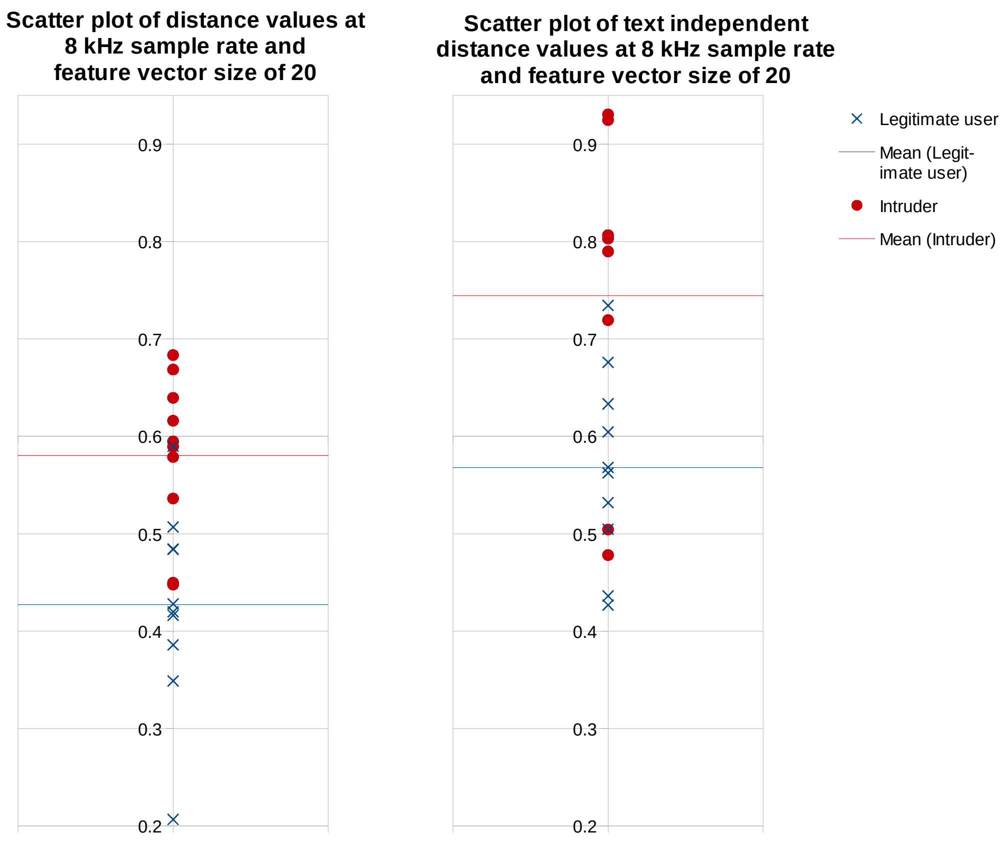


Figure 4.7. Comparison of the legitimate vs. intruder scatter plots for Mifare Ultralight tags when using a text-dependent vs. text-independent system

4.3.3 Effect of native code on performance

It was found that on average native code performed about 9 times faster than non-native Java code. Figure 4.8 shows the relative processing times for the enrolment, training and classification stages of the application, respectively. In the enrolment stage, a sentence is recorded, 500 LPC features are extracted, and the feature vector is stored to a CSV file on the phone's SD card. The time given in Figure 4.8 for enrolment does not include the recording time, which is constant in both implementations. Training may be performed on any number of enrolled samples, but the results in Figure 4.8 are given for the training of 3 enrolled samples. In the training stage the enrolled feature vectors are loaded from file storage and the average feature vector is computed and stored to a separate CSV file. For the comparison of the classification stage, a single sample was classified. The application interface instructs the user on a sentence to pronounce. The sentence is recorded and the LPC features are extracted. The Euclidean distance is then computed between the current feature vector and the average feature vector computed in the training stage. The timing results in Figure 4.8 for classification does not include the recording time.

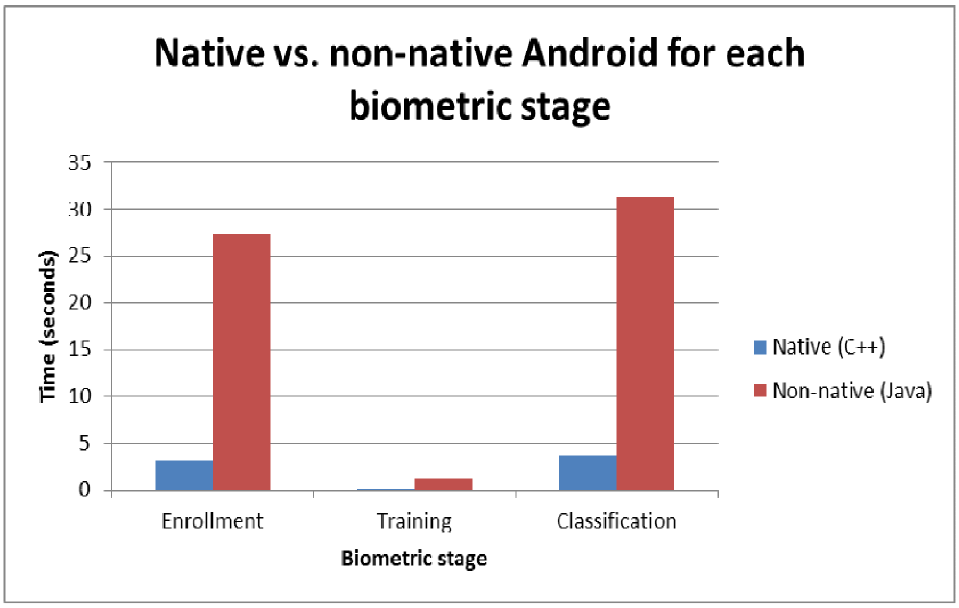


Figure 4.8. Comparison of the processing time in Java and C++ on Android for each of the biometric stages

As was seen in Figure 3.6 the recording time of the sentence “Rome wasn’t built in a day” takes about 2.5 seconds to record, regardless of whether Java or C++ mode is being used. This time is always added to the total system time in the enrolment and verification modes, which means the total enrolment time is about 30 seconds per sample in Java and about 6 seconds in C++, and a slightly larger timeframe is required for verification.

CHAPTER 5 DISCUSSION

5.1 GENERAL DISCUSSION OF RESULTS

In the first experiment the MARF audio library was used to exhaustively test all available algorithms in terms of training time, identification time, and identification accuracy on a PC and on a Google Nexus S smartphone. It was found that the smartphones performs about 30 times slower than the PC on average when running a Java application. A comparison of the identification accuracy results was made with the results that were obtained by the developers of the MARF library to confirm correct operation. Several candidate algorithms performed reasonably well in terms of FRR, and training and testing time on the phone. The combination of Endpoint preprocessing, LPC feature extraction, and Manhattan-Distance classification scored the best results in terms of accuracy. For this combination, the training time on the smartphone was measured as 4.66 seconds per audio sample and the testing time as 5.03 seconds per audio sample. These times were deemed to be reasonable, and LPC feature extraction was therefore used for the third experiment.

Although the testing times given in Table 4.2 cannot be directly compared with the training times in Table 4.1, it is worth noting that the time taken to identify a single sample takes more than double the time taken to train a system on one sample. In fact, when the average time per sample is measured for all algorithm combinations, including those that do not execute on the phone, the time to identify a single sample is more than three times the time it takes to train the system on one sample. This is not true for the neural network implementation, which trains much longer than it classifies, but it is quite intuitive that the distance-based implementations would take longer to identify a sample than to “train” itself on a sample, because it is only in identification mode that the distance-based classifiers execute the actual distance algorithms. Figure 4.8 also shows that the identification time was about 30 times larger than the training time for the third experiment in Java and about 100 times slower in C++.

For the second experiment OpenCV was used to determine the relative processing time of the PC and the smartphone when running C++. For this experiment it was found that the phone was about 21 slower than the PC. In terms of speed, the Manhattan distance implementation was the quickest, but not by much. Both the Euclidean distance and Diff distance classifiers were almost as fast as the Manhattan distance classifier, but the Minkowski was relatively slow, probably because of the “pow” mathematical function that was used with powers smaller than 1. In terms of accuracy, the Minkowski distance was however by far the most accurate. Diff distance was a bit more accurate than Manhattan distance, and the Euclidean distance classifier was the least accurate. Considering the small difference in processing time between Minkowski distance and the other classifier, the increase in accuracy would justify the use of the Minkowski distance rather than the other distance-based methods.

In the third experiment SPro and jAudio libraries were used for the implementation of the LPC feature extraction algorithm and this was combined with a simple Euclidean distance classifier to analyse a complete end-to-end biometric prototype under various configurations and to determine the relative processing time between C++ and Java. It was found that audio sample rate did not have much effect on the system as long as the sample rate is at least 8 kHz to cover human voice frequencies. Feature vector length and text-independence showed significant influence on the results. Longer feature vector length is generally better although it results in much longer transmission time. Text-independence allows a level of usability when the user is able to speak any words to be verified, but it results in a significant decrease in system accuracy. Consequently, trade-offs need to be made to “tune” such a biometric system for each application. After porting the application to C++, it was found that C++ application runs about 9 times faster than Java applications, and this project therefore recommends the use of C++ for processing intensive applications on smartphones. Although the Java results are significantly slower than the native code, the use of Java can also provide advantages over a native implementation in many cases. Table 5.1 compares the use of non-native Java code and native C++ code on a mobile platform. In addition to the consideration in Table 5.1, Java is also the recommended programming language for Android, due to a higher level of support by the Android API, especially for specialised tasks such as interfacing with sensors of the phone. In certain applications, Java is therefore still the preferred language.

Table 5.1. Comparison of the overall usage benefits of Java and C++ for mobile measurement and processing applications

Property	**Java**	**C/C++**
Ease of porting	Relatively simple	Complex
Ease of application implementation	Very simple	More involved
Availability of open-source software	More limited than C++	Very wide availability
Processing speed	Slower	Faster
Compatibility with standard desktop libraries	Mostly compatible (GUI libraries and some system libraries are unavailable)	Mostly compatible, but libraries depend on each other and require the porting of dependencies

When considering some of the results given in Tables 4.1, 4.2, 4.4 and Figure 4.8, it appears at first that the training or identification time may be too large for some algorithms, to implement a comfortable, user-friendly biometric system on a phone. However, for the final system, only a single combination or a few combinations is chosen and implemented, and when considering a single combination of algorithms being used to train and classify a single audio sample, the processing time is usually very small for C++ but not always as small for Java.

The accuracy results that were obtained were always equivalent on the phone and the PC and are library- and algorithm- specific. For this reason, the accuracy of the implemented algorithms was not an important focus of this project. Some of the accuracy results given in Tables 4.3 and 4.5 also suggest that many of the implemented algorithms do not provide for a feasible implementation due to high error rates. However, the accuracy of many of the algorithms was found to be quite high, and these results can be further improved when considering the following.

- For the second experiment, greyscale images were used with a 92x112 resolution, whereas the Nexus S provides a colour camera with a 2560x1920 resolution. This means that the phone can provide up to 11 times more digital samples per audio clip than the clips used in the first experiment, and 1431 times more samples from the images in the second experiment. Using more samples will increase the processing time, but will provide better discrimination between samples from different users (higher inter-class variation).
- When using the phone to record speech, the microphone can be held close to the speaker's mouth, providing a high SNR in an environment with reasonable ambient noise.
- By requiring speakers to always say the same phrase, the speaker recognition system is converted to a text-dependent speaker recognition system. This increases the accuracy of the system, because there is much less intra-class variation (variation within the samples from one speaker).
- The speaker recognition and the face recognition systems in the first two experiments both operated in identification mode instead of verification mode. In other words, the user being recognised does not supply any identification information with the biometric measurement. As was stated in Section 2.6, identification mode increases the FAR, because the current sample is compared with all users in the biometric database. In contrast, peer-to-peer biometric verification on the smartphone can effectively be interpreted as an identification system with only one biometric entry in the database. The effect on the FRR is however negligible when converting an identification system to a verification system [31], but the FRR can be decreased by varying the threshold under which a user is classified as an intruder (there is more room for varying the threshold when the FAR is decreased).
- The accuracy may be increased with the implementation of multimodal biometrics. Speaker and face recognition can be combined to increase the accuracy. This may cause an increase in processing time, but when using C++, the larger processing time would still be reasonable. Fusion of data from the voice and face modalities

can be performed at various levels of the biometric pipeline. The fusion level directly influences the resulting accuracy and is therefore a topic that requires further research.

- Other biometric modalities can be applied as the appropriate sensors become available on smartphones. A good example is fingerprint scanners which are currently entering the mobile space.

5.2 SIGNIFANCE OF EXPERIMENTAL OUTPUTS

At present, two biometric modalities have been studied, speaker recognition and face recognition. Both were found to be viable both in terms of processing time and accuracy. There is however much room for improving the accuracy, which may be accomplished by combining the face, voice and possibly other modalities, as well as implementing better classification algorithms.

In the past, mobile phones could mainly be applied as communications devices in measurement applications to transfer data from dedicated sensors to remote computers for processing [10]. A short-range technology such as NFC or Bluetooth would commonly be implemented between the phone and the sensor, and a long-range telecommunications technology could then be used for data transmission to a remote computer. Such an elaborate system is often not required, and smartphones can now replace either the sensor side, the computer side, or both sides of the system in many applications. Smartphones can provide the sensors, the processing resources and a high level of portability.

The motivation for benchmarking the performance of a current smartphone was to establish whether or not mobile measurement and processing applications are realistically feasible. Smartphones are widely adopted and the latest smartphones are bought off-the-shelf with a wide range of sensors and powerful processing resources. They run powerful operating systems and SDKs are available for rapid application development. This project proposed that general-purpose mobile devices offer a distinct advantage over specialized hardware and software in many applications. One of the main advantages is cost, because a smartphone can be reused for a variety of applications, whereas specialised devices cannot.

Smartphones are very adaptable. Open-source software can be ported to a smartphone platform or software can be designed from scratch, and even the operating system of a phone can be modified if an open-source platform, such as Android, is used. When considering these advantages, smartphones may be a better choice for many sensing and processing application over specialised single-purpose hardware devices.

CHAPTER 6 CONCLUSION

6.1 RESEARCH CONCLUSIONS

The use of biometrics provides many benefits over other security mechanisms such as having to remember one or multiple PIN codes or having to carry around an identification document or a key. Easy- to-remember PIN codes are easily guessed, while hard-to-guess PIN codes are easily forgotten by the legitimate user. PIN code security relies on what a person knows or remembers and key-based security relies on what a person possesses, while biometric security relies on who a person is [31], [32].

The use of biometrics is becoming a very important security consideration as the use of smartphones for banking, credit card payments, electronic ticketing, access control and other confidential activities increase. Many people also store sensitive information on their smartphones which could lead to damage if compromised [35]. Biometrics provides a possible alternative for securing important mobile data. At the moment there are however disadvantages to using biometrics, which include the need for complex hardware and in many cases the tediousness of performing biometric recognition on a regular basis in busy environments. For the mitigation of the need for complex hardware, this dissertation proposed the use of smartphones as biometric devices for general use and specifically for use as a replacement to paper ID documents.

To study the viability of smartphones as biometric devices, the sensing and processing capabilities were studied as well as the effect of various system variables on performance and accuracy. The processing power of a smartphone was compared with that of a PC, and the speed of Java code was also compared with that of native C/C++ code on a mobile platform. The ratio of processing time between a PC and a phone was found to be about 30 in Java and about 21 in C++. It was also found that, on average, a nearly identical implementation of C++ code performs about 9 times faster than in Java on a smartphone. The goal of this native versus non-native performance comparisons was to determine which kind of implementation will provide a better solution, when also taking into consideration the availability and portability of software in both languages, as well as the fact that Java code is better supported in Android, especially when working with peripheral devices. Native support for Android applications is however increasing constantly.

Speaker recognition and face recognition applications were implemented in three different experiments. The first experiment involved the use of the MARF Java library for speaker recognition using the built-in microphone of the smartphone. An initial face recognition application was then implemented in C++ using OpenCV. The first two experiments were performed on both a PC and a smartphone to compare the performance. The third experiment was then conducted to determine the effect of changes in various system parameters on system accuracy and performance, as well as the performance increase when using native code on the smartphone as compared to non-native Java code. For this experiment the jAudio and SPro libraries were used for the Java and C++ implementations respectively. Various feature extraction and classification algorithms were implemented and compared for each experiment as discussed in Chapter 3.

No loss in identification accuracy was observed on the phone when compared to a PC. Considering the measured results of these proof-of-concept biometric applications, this dissertation suggests that a variety of processing and sensing intensive applications should be feasible on current mobile platforms using readily available open-source software, and such implementations provide many benefits over the use of specialized hardware and software.

6.2 POSSIBLE FUTURE WORK

Suggested future research includes the porting of similar benchmarking software to other smartphone platforms for a processing speed comparison between different handset models and operating systems to find which platforms provide the best performance and usability in different circumstances. Such a study may also include tablet computers. Future research may also be conducted into other useful software packages that may be ported to mobile platforms for specific applications, or for sensing and data processing in general.

Other future work on biometric recognition on smartphones may include the combination of biometric modalities for an increase in accuracy, especially considering the variable environment that smartphones are used in. More complex classification algorithms may also be used to serve the same purpose such as support vector machines (SVMs), hidden markov models (HMMs) and neural networks. Biometric modalities other than face and voice may also be studied in further detail. Examples include teeth, hand geometry, gait and hand movement. Since fingerprint scanners are currently being integrated in smartphones, fingerprint biometrics will soon become viable and will provide more accurate biometric recognition on smartphones. Since the introduction of contactless ID smartcards in South Africa a new area of research has also opened to study the possible future evolution from these plastic cards to NFC phones, possibly utilising an embedded secure element or cloud-based secure element to store sensitive ID information and having an app that could display and update information as requested. These smartcards will contain fingerprint features of South African citizens and will have a contactless interface which will in theory allow NFC phones to read the cards and perform fingerprint verification.

REFERENCES

[1] R. Want, “Enabling ubiquitous sensing with RFID,” *Computer*, vol. 37, no. 4, pp. 84-86, Apr. 2004.

[2] R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison, “Bridging physical and virtual worlds with electronic tags,” in *Proc. CHI 99 Conf.: CHI is the Limit - Human Factors in Computing Systems*, pp. 370-377, 1999.

[3] S. Kerr, H. Thinyane, and G. Foster, “Mobile phone performance analysis for camera based visual interactions,” in *Annu. Research Conf. South African Inst. of Comput. Scientists and Inform. Technologists, Proc. ACM Int. Conf. Proc. Series*, Vanderbijlpark, Emfuleni, 2009, pp. 80-86.

[4] NFC Forum. (2004, Mar.). Nokia, Philips And Sony Establish The Near Field Communication (NFC) Forum. [Online]. Available: http://www.nfc-forum.org/news/pr/view?item_key=d8968a33b4812e2509e5b74247d1366dc8ef91d8

[5] *Information technology - Telecommunications and information exchange between systems - Near Field Communication – Interface and Protocol (NFCIP-1)*, ISO/IEC Std. 18092, 2004.

[6] *Near field communication interface and protocol (NFCIP-1)*, ECMA Standard 340, 2004.

[7] NXP Semiconductors. (2009, Apr.). NFC Forum Type Tags, White paper v1.0. [Online]. Available: http://www.nfc-forum.org/resources/white_papers/NXP_BV_Type_TTTT_White_Paper-Apr_09.pdf

[8] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication, 3rd edition. West Sussex, United Kingdom: Wiley, 2010.

[9] NFC Forum. (2010, Jul.). NFC Forum Connection Handover Technical Specification v1.2. [Online]. Available: http://www.nfc-forum.org/specs/spec_list/

[10] C. A. Opperman, and G. P. Hancke, “Using NFC- enabled phones for remote data acquisition and digital control,” in *Proc. IEEE Africon 2011*, Victoria Falls, Livingstone, 2011, pp. 1–6.

[11] E. E. Imhontu, and Y. O. Kumah, “A survey on Near Field Communication in mobile phones and PDAs”, M.S. thesis, School Inform. Sci., Comput., Elect., Halmstad, Univ., Halmstad, Sweden, 2010.

[12] Order Detail ID: 64035839

RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-field Communication by Finkenzeller, Klaus;

Müller, Dörte. Reproduced with permission of Wiley in the format Reuse in a dissertation/thesis via Copyright Clearance Center.

[13] Near Field Communication Forum Inc. (2006, Jul.). NFC data exchange format (NDEF). Wakefield. [Online]. Available: http://www.nfc-forum.org/specs/spec_list

[14] *Identification cards – Contactless integrated circuit(s) cards – Proximity cards*, ISO/IEC Standard 14443, 2001.

[15] *Identification cards – Contactless integrated circuit cards – Vicinity cards*, ISO/IEC Standard 15693, 2010.

[16] E. Strömmer, M. Hillukkala, and A. Ylisaukko-oja, "Ultra-low Power Sensors with Near Field Communication for Mobile Applications", *IFIP International Federation for Information Processing*, vol. 248, pp. 131-142, 2007.

[17] W. Guo, "Design of architecture for mobile payments system", in *Proc. Control and Decision Conf.*, Yantai, 2008, pp. 1732-1735.

[18] R. K. Balan, N. Ramasubbu, K. Prakobphol, N. Christin, and J. Hong, "mFerio: The design and evaluation of a peer-to-peer mobile payment system", in *Proc. 7th ACM Int. Conf. Mobile Systems, Applications, and Services*, Krakow, 2009.

[19] J. Sieck, "Location based services and museum information systems," in *Proc. 3rd International Conference on Intelligent Systems Modelling and Simulation*, Kota Kinabalu, 2012, pp. 663–666.

[20] I. L. Ruiz, and M. A. Gómez-Nieto, "University smart poster: Study of NFC technology applications for university ambient," *Advances in Soft Computing*, vol. 51, pp. 112-116, 2009.

[21] R. Hardy, E. Rukzio, M. Wagner, and M. Paolucci, "Exploring expressive NFC-based mobile phone interaction with large dynamic displays," in *Proc. 2009 1st Int. Workshop Near Field Communication*, Hagenberg, Austria, 2009, pp. 36–41.

[22] C. A. Opperman, and G. P. Hancke, "A generic NFC-enabled measurement system for remote monitoring and control of client-side equipment," in *Proc. 2011 3rd Int. Workshop Near Field Communication*, Hagenberg, Austria, 2011, pp. 44–49.

[23] A. Marcus, G. Davidzon, D. Law, N. Verma, R. Fletcher, A. Khan, and L. Sarmenta, "Using NFC-enabled Mobile Phones for Public Health in Developing Countries", in *Proc. 2009 1st Int. Workshop on Near Field Communication*, 2009, pp. 30-35.

[24] G. Govindan, S. K. Balakrishnan, R. L. Ratheendran, and S. K. Sivadasan, "Real time security management using RFID, biometric and smart messages", in *Proc. 3rd Int. Conf. Anticounterfeiting, Security and Identification in Communication*, Hong Kong, 2009.

[25] K. Hyppönen, M. Hassinen, and E. Trichina, “Combining biometric authentication with privacy-enhancing technologies”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4968, pp. 155-165, Mar. 2008.

[26] A. Juels, D. Molnar, and D. Wagner, “Security and privacy issues in e-passports”, in *1st Int. Conf. Security and Privacy for Emerging Areas in Commun. Networks*, Athens, 2005, pp. 74-85.

[27] News24. (2012, Apr.). ID smartcards not far off. *News24*. [Online]. Available: http://www.news24.com/SouthAfrica/Politics/ID-smartcards-not-far-off-20120425

[28] J. Blum. (2011, Oct.). Is Google Wallet the Next Step in Mobile Payments? *Entrepreneur*. [Online]. Available: http://www.entrepreneur.com/blog/220500

[29] C. A. Opperman, and G. P. Hancke, “Smartphones as a platform for advanced measurement and processing,” in *Proc. 2012 IEEE I2MTC - International Instrumentation and Measurement Technology Conference*, Graz, Austria, 2012, pp. 703–706.

[30] A. Jain, L. Hong, and S. Pankanti, “Biometrics: Promising Frontiers for Emerging Identification Market”, *Communication of the ACM*, vol. 43, no. 2, pp. 91-98, Feb. 2000.

[31] A. K. Jain, A. Ross, and S. Prabhakar, “An introduction to biometric recognition”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4-20, Jan. 2004.

[32] K. Delac, M. Grgic, “A survey of biometric recognition methods”, in *46th Int. Symp. Electronics in Marine*, Zadar, 2004, pp. 184-193.

[33] IEEE journal of solid-state circuits by IEEE SOLID-STATE CIRCUITS COUNCIL. Reproduced with permission of INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, in the format Republish in a thesis/dissertation via Copyright Clearance Center.

[34] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, “FVC2002: Fingerprint verification competition,” in *Proc. Int. Conf. Pattern Recognition (ICPR)*, Quebec City, QC, Canada, 2002, pp. 744-747.

[35] D. J. Kim, K. W. Chung, and K. S. Hong, “Person authentication using face, teeth and voice modalities for mobile device security”, *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2678-2685, Nov. 2010.

[36] A. Ross, A. K. Jain, “Multimodal biometrics: An overview”, in *Proc. 12th European Signal Processing Conference*, Vienna, 2004, pp. 1221-1224.

[37] A. Morris, S. Jassim, H. Sellahewa, L. Allano, J. Ehlers, D. Wu, J. Koreman, S. Garcia-Salicetti, B. Ly-Van, and B. Dorizzi, “Multimodal person authentication on a smartphone under realistic conditions”, in *Proc. SPIE – Int. Soc. for Optical Eng.*, vol. 6250, Kissimmee, 2006.

[38] L. Allano, A. C. Morris, H. Sellahewa, S. Garcia-Salicetti, J. Koreman, S. Jassim, B. Ly-Van, D. Wu, and B. Dorizzi, “Non intrusive multi-biometrics on a mobile device: a comparison of fusion techniques”, in *Proc. SPIE – Int. Soc. for Optical Eng.*, vol. 6202, Kissimmee, 2006.

[39] M. O. Derawi, C. Nickely, P. Bours, and C. Busch, “Unobtrusive user-authentication on mobile phones using biometric gait recognition”, in *6th Int. Conf. Intelligent Inform. Hiding and Multimedia Signal Process.*, Darmstadt, 2010, pp. 306-311.

[40] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Cell Phone-Based Biometric Identification”, in *4th IEEE Int. Conf. Biometrics: Theory, Applicat. and Syst.*, Washington, DC, 2010.

[41] F. Okumura, A. Kubota, Y. Hatori, K. Matsuo, M. Hashimoto, and A. Koike, “A Study on Biometric Authentication based on Arm Sweep Action with Acceleration Sensor”, in *Int. Symp. Intelligent Signal Process. and Commun.*, Yonago, 2006, pp. 219-222.

[42] H. Manabe, Y. Yamakawa, T. Sasamoto, and R. Sasaki, “Security evaluation of biometrics authentications for cellular phones”, in *5th Int. Conf. Intelligent Inform. Hiding and Multimedia Signal Process.*, Kyoto, 2009, pp. 34-39.

[43] Jae Min Kang, T. Yoo, and Hee Chan Kim, "A Wrist-Worn Integrated Health Monitoring Instrument with a Tele-Reporting Device for Telemedicine and Telecare," *IEEE Transactions on Instrumentation and Measurement,* vol. 55, no. 5, pp. 1655-1661, Oct. 2006.

[44] B. Gozick, K. P. Subbu, R. Dantu, and T. Maeshiro, "Magnetic Maps for Indoor Navigation," *IEEE Transactions on Instrumentation and Measurement,* vol. 60, no. 12, pp. 3883-3891, Dec. 2011.

[45] G. Hache, E. D. Lemaire, and N. Baddour, "Wearable Mobility Monitoring Using a Multimedia Smartphone Platform," *IEEE Transactions on Instrumentation and Measurement,* vol. 60, no. 9, pp. 3153-3161, Sept. 2011.

[46] K. M. Bin Saipullah, A. Anuar, N. A Binti Ismail, and Y. Soo, "Real-time video processing using native programming on Android platform," in *Proc 2012 IEEE 8th Int. Colloquium on Signal Processing and Its Applications,* CSPA 2012, pp. 276-281.

[47] Google Inc. (2010, Dec.). Nexus S owner’s guide. [Online]. Available: http://www.samsung.com/us/Nexus_S_Owners_Guide/

[48] Modular Audio Recognition Framework v.0.3.0.5 (0.3.0-devel-20060226) and its Applications, *The MARF Research and Development Group*, Canada, 2006.

[49] X. Wang, and K. K. Paliwal, “Feature extraction and dimensionality reduction algorithms and their applications in vowel recognition,” *Pattern Recognition*, vol. 36, no. 10, pp. 2429-2439, Oct. 2010.

[50] L. Deng, and D. O'Shaughnessy, Speech processing: a dynamic and optimization-oriented approach. *CRC Press*, 2003, pp. 41–48.

[51] A. K. Jain, R. P. W. Duin, and J. Mao, “Statistical pattern recognition: A review,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, Jan. 2000.

[52] The OpenCV Reference Manual: Release 2.3, 2011.

[53] D. O’Shaughnessy, “Linear Predictive Coding,” *IEEE Potentials*, vol. 7, no. 1, pp. 29-32, Feb. 1988.

[54] S. Sonnenburg, G. Raetsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc, “The SHOGUN Machine Learning Toolbox,” *Journal of Machine Learning Research*, vol. 11, pp. 1799-1802, Jun. 2010.

[55] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” *SIGKDD Explorations*, vol. 11, no. 1, 2009.