University of Southampton

Polymer Thick Film Sensors for Embedded Smartcard Biometrics and Identity Verification

by
Neil James Henderson

A thesis submitted for the degree of
Doctor of Philosophy
in the
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science

June 2002

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

POLYMER THICK FILM SENSORS FOR EMBEDDED SMARTCARD BIOMETRICS AND IDENTITY VERIFICATION

by Neil James Henderson

Smartcards offer robust mechanisms for electronic demonstration of identity, and as a result are an increasingly important tool of e-commerce and information security. The identity of an authorised individual and their smartcard are, however, weakly bound and the presence of a legitimate card-holder can not be assured. Hence, there is growing interest in the use of biometrics to strengthen the association between card and card-holder, although the emphasis has predominantly been to use external components such as sensors or processing elements. Such approaches expose the risk of eavesdrop and replay attacks, and because of this it is desirable to incorporate all elements of an identity-verification system on-card.

Potential discriminatory characteristics are limited to the measurable interactions between an individual and smartcard. Hence, the emerging field of biometrics is comprehensively reviewed for plausible mechanisms of demonstrating identity. Spatial finger characteristics such as fingerprints, finger-geometry and finger-crease pattern emerge as the most likely approaches.

Sensors integrated onto smartcards are required to be mechanically flexible, robust and of low-cost, and such conditions are satisfied by a class of sensors known as polymer thick film (PTF). Piezoelectric and piezoresistive PTF pressure sensors are considered, and shown to exhibit sensitivity to loads applied elsewhere on the card. This effect renders the concept of using PTF pressure sensors to capture spatial finger characteristics infeasible.

Nevertheless such sensing mechanisms can be exploited to capture temporal, rather than spatial interactions, and a novel approach to identity verification is proposed and demonstrated. This approach is based upon the pressure response of finger-taps, and an experiment involving 34 participants demonstrates significant discrimination between individuals. An investigation into suitable verification functions reveals that an equal error rate of 2.3% is achievable under controlled laboratory conditions. Necessary enrolment and verification algorithms are implemented on a typical smartcard platform, and found to execute within 3.1 and 0.12 seconds, respectively. Full compliance with the mechanical, computational, and economical constraints of a smartcard is consequently demonstrated.

This thesis is dedicated to my mum, Irene, who died three days after my viva.
God bless you mum, and may you rest in peace.

Irene Marshall Henderson
20^{th} January 1943 – 11^{th} July 2002

Acknowledgements

It's not easy finishing a PhD thesis, but I found the influence of a number of people helped greatly. Of primary note are my mum and dad, whose unshakable love, support and tolerance was fundamental to my success. The love, encouragement and understanding of Sharon helped me through the *Dark Stages*, whilst Mhairi, my fourteen month-old niece, demonstrated the usefulness of this work by employing a draft thesis as a step. Mhairi should also be thanked for giving me chicken pox shortly after I finished this thesis. My sister and brother-in-law, Fiona and John, are thanked for just being there, and of-course for the calamine lotion.

My friends deserve special mention, not least those closest to the ill-tempered outbursts of a thesis-writing Scotsman. These people include: Thomas; Yavus; Paul; Marcus; Theo; Nicola; Sayed; Neil G; Edward; Peter W; Andy R; and Dan M. Occasionally during the process of writing-up I was dragged, kicking-and-screaming, to The Crown. Some of those (ir)responsible include: John M; Richard; Jasmin; Andy P; Jason; Anna; and Mike G. Finally a number of close friends must be thanked for putting up with all the unreturned 'phone calls, emails, and the endless excuses of *...I'm too busy...* These people include: Iain; Donny; Graham; Ed; Jamie; Steve; Mike C; Adam; Dan S; and Tim. Sorry Guys - I'll make it up to you.

Before starting my PhD, I worked for DERA. My resource manager, Dr. Juliet Dunn-Rogers, deserves thanks for encouraging this venture, and for organising financial assistance from DERA.

Finally, there would be no PhD project if not for my supervisors, Dr. Neil White and Dr. Pieter Hartel. Both deserve thanks for their support, encouragement, and for at least humouring my sillier ideas.

Contents

List of Figures

List of Tables

Nomenclature

C	Capacitance (Farads)
V	Voltage (Volts)
Q	Charge (Coulombs)
R	Resistance (Ohms)
GF_L	Longitudinal Gauge Factor
ε	Strain, proportional change in dimension
ε_x	Longitudinal Strain, $\frac{\delta x}{x}$
ε_y	Transverse Strain, $\frac{\delta y}{y}$
ε_z	Normal Strain, $\frac{\delta z}{z}$
σ	Stress, $\frac{F}{A}$ (Pascals)
σ_x	Planar Stress in the x direction
σ_y	Planar Stress in the y direction
σ_z	Normal Stress in the z direction
E	Young's modulus, $\frac{\sigma}{\varepsilon}$ (Pascals)
ν	Poisson's ratio, $-\frac{\varepsilon_y}{\varepsilon_x} = -\frac{\varepsilon_x}{\varepsilon_z}$
P	Volume proportion
ρ	Resistivity (Ohm metre)
G	Piezoresistivity coefficient
d_{33}	Piezoelectric coefficient describing charge response to normal force (Coulomb / Newton)

d_{31}, d_{32} Piezoelectric coefficient describing charge response to planar x or y forces (Coulomb/Newton)

D_3 Electric Displacement (Coulombs/m^2)

Ξ Electric Field Strength (V/m)

n_{max} Maximum number of sensing elements per row of a sensing array

λ_{min} Minimum screen printing linewidth

FAR False Acceptance Rate

FRR False Rejection Rate

EER Equal Error Rate

$\mathbf{R}$ Reference feature vector

$\mathbf{T}$ Test feature vector (verification)

$\mathbf{U}$ Unknown feature vector (identification)

r_i ith component of $\mathbf{R}$

t_i i^{th} component of $\mathbf{T}$

$\overline{D}_{Enrolment}$ Mean distance of enrolment vectors from reference vector, $\mathbf{R}$

θ Acceptance Thresold

τ Acceptance Tolerance

$\|\cdot\|_1$ ℓ_1 norm

$\|\cdot\|_2$ ℓ_2 norm

$\mathbf{C}$ Class Covariance Matrix

$\mathbf{V}$ Class variance Matrix, leading diagonal of $\mathbf{C}$

$(\cdot)^*$ Transpose operator

Chapter 1

Introduction

Modern smartcards are fully programmable, open and trusted computational platforms, and are used for a wide range of functions. At the simplest level, they can be involved in the storage of electronic data and for the prepayment of goods and services (Rankl & Effing 1997). With cryptographic capabilities, the applications of high-end smartcards include electronic authentication, and the provision of mechanisms for non-repudiation of transactions and document creation (Leach 1995, Hansmann, Nicklous, Schäck & Seliger 2000, Borst, Preneel & Rijmen 2001, M'Raïhi & Yung 2001, Cattell, Carroll & Saby 2002). As a result, smartcards are an important tool of e-commerce and information security.

Whilst a smartcard can robustly demonstrate its involvement during a transaction, or its presence during remote access of resources, the presence of its rightful holder can not be assured. Hence, there is currently much interest in the use of biometrics to securely demonstrate identity to the card (Adams 2000). However, the emphasis has predominantly been to use components which are external to the card, such as biometric sensors or processing elements to execute biometric algorithms (Lapère & Johnson 1997, Sanchez-Reillo 2001, Aufreiter 2001, Praca & Barral 2001). Not only does such an approach expose the system to the possibility of eavesdrop and playback attacks (Hachez, Koeunne & Quisquater 2000), it may also inconvenience the holder by restricting where the card can be used. For these reasons it is desirable to embed biometric sensors onto the smartcard itself, thereby creating an entirely self-contained identity verification system. This is a challenge in terms of the manner of demonstrating identity, and in the method of capturing biometric

characteristics. Sensors required to capture discriminatory characteristics must necessarily be sufficiently flexible and robust to withstand normal smartcard usage, and do so in an economically compliant manner.

This thesis considers the problem of strengthening the association between a smartcard and its legitimate holder. Appropriate biometric characteristics and sensor mechanisms are investigated with the ultimate goal of a self-contained, on-card identity verification system. A novel method of demonstrating identity to the card is proposed, demonstrated, and shown to be compliant with the constrained computational resources of a typical modern smartcard platform.

This chapter begins with an overview of smartcards, their applications and the motivation for restricting access to their contents. Current approaches to the protection of smartcards are considered, and in conclusion an outline of the work presented in this thesis is given.

1.1 A Brief History of Smartcards

The development of smartcards can be traced to the first plastic banking and payment cards, introduced during the 1950s.[1] These served only as durable, portable records of the card-holders details such as their name and card number. A signature field on the back of cards provided all that was required to authenticate the user and permit transactions to occur. Security at this time relied solely on the basis of visual inspection by retail staff, and as a consequence fraud was rife (Rankl & Effing 1997).

The introduction of magnetic stripe cards, in the early 1970's, offered some improvement in that more data could be stored in a machine readable form (Zoreda & Otón 1994). However, since magnetic stripes are easily read and rewritten with only moderate equipment (a process known as *skimming*), these cards are not well-suited to storing confidential information.

Several patents appeared during the late 1960's and early 1970's offering possible solutions to the insecurity of magnetic data storage (Dethloff & Grötrupp 1968,

[1] Payment cards were introduced firstly by Diner's Club (1950), followed by American Express (1958) and Bank of America (1959). For more detail, see Rankl & Effing (1997).

Ellinboe 1972, Castrucci 1972, Moreno 1974, Dethloff 1978). These patents all proposed variants on the idea of embedding integrated circuits into plastic cards.[2] Secret or confidential data could be securely stored within the integrated circuits, making unauthorized access or copying of the data significantly more difficult than had previously been the case.

Little progress in adopting this new technology occurred, until 1984 when the French Postal and Telecommunications agency (PTT) trialed the use of integrated circuit cards as telephone payment cards. The new cards offered significant advantages over coin-based payments, in that monies did not require to be collected from public telephones. Containing less (or no) money, public telephones are less prone to vandalism and theft (Rankl & Effing 1997).

The success of the French telephone cards is entirely evident in the circulation numbers; By 1990, six years after the first trials, an estimated 60 million IC telephone cards were circulating throughout France. Prepaid telephone cards are now used in over 60 countries world-wide (Hansmann et al. 2000).

The type of IC card used in these telephone cards is known as a *memory card.* These cards are prepaid, and their purchase value is stored electronically on the card. Their requirement is simply to securely decrement an internal value. Because of their convenience and low-cost, memory cards have displaced cash in numerous small value transactions, including public transportation, vending machines, photo-copying machines and canteens (Chen 2000).

Memory cards are only suitable for closed systems where a single authority controls the issue of the cards and accepts them for payment. There must be little incentive for fraud in these systems, since transactions can be recorded, then simulated using an elementary microcontroller (Rankl & Effing 1997).

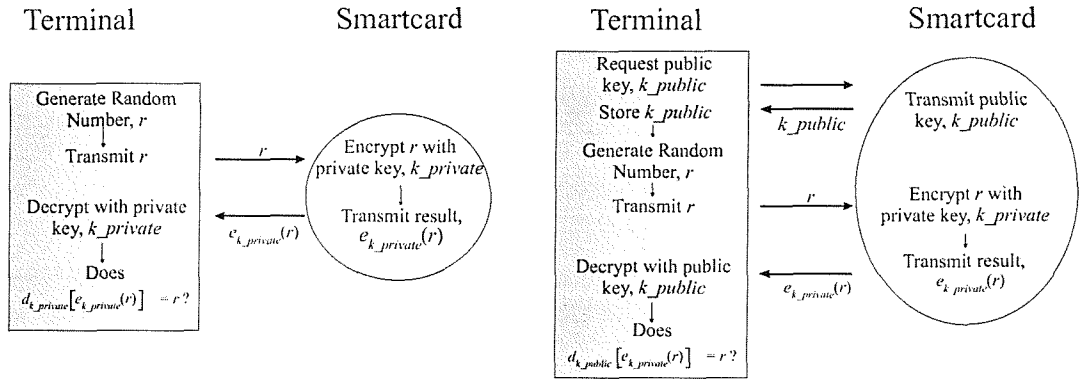
With continued improvements in semiconductor processing techniques, increasingly complex circuitry could be manufactured onto the small areas of silicon of the smartcard.[3] This enabled microprocessor functionality to be exhibited by the smartcard, and data could be processed in addition to being stored. Such cards are termed *microprocessor cards.*

[2]Generically termed *integrated circuit cards* or *IC cards*

[3]The ISO 7816 standard specifies a maximum silicon area of 25mm^2 in order to prevent fracture when the card is flexed (Zoreda & Otón 1994)

The improved capabilities of microprocessor cards led to increased functionality and offered for the first time cryptographic capabilities within the smartcard. In the above example of the prepaid memory card, fraud is possible because the payment terminal is unable to verify the authenticity of the card used for payment. With cryptographic functions this problem can be solved.

A tamper resistant module is installed in each payment terminal which is used to generate and transmit a random number (called a *challenge*) to the smartcard. The smartcard encrypts this random number according to a symmetric encryption algorithm[4], using a private key shared with the payment terminal. The result is sent back to the terminal, decrypted and compared to the original challenge. If the two are the same then the smartcard has demonstrated its authenticity and the transaction may proceed. This is shown in Figure 1.1(a).



(a) Private Key Approach (b) Public Key Approach

Figure 1.1: Symmetric and Asymmetric Authentication Mechanisms: e_k & d_k denote encryption and decryption operations, using a key k, respectively.

Security in such a scheme relies inherently upon the secrecy of private keys. If keys are recovered from a secure module in any of the payment terminals or smartcards, then the entire system can be compromised.

A way around this is to use an asymmetric public–private key approach.[5] Two keys, one of which is private (and hence secret) and the other public, are stored on a smartcard. These keys are related, but in such a way that the private key

[4]Such as DES (Data Encryption Standard), or triple DES. See Schneier (1996) for details.
[5]Such as RSA. See Rivest, Shamir & Adleman (1978) for details.

cannot easily be obtained from the public key. In the above prepaid payment application, the payment terminal sends a random challenge to the smartcard, this is encrypted with the smartcard's private key, and the result transmitted back to the terminal. The terminal requests the card's public key which is then used to decrypt the result. This is compared to the original challenge, as is shown generally in Figure 1.1(b). This procedure is called *authentication* and is designed to prove authenticity, rather than concealment of data. For a detailed account of the cryptographic functionality found on current smartcards see (Borst et al. 2001, Guillou, Ugon & Quisquator 2001).

The use of public keys to demonstrate authenticity removes the need for secure tamper proof modules in payment terminals. Furthermore, each card contains its own unique private key, which means that the integrity of the whole system is not jeopardized if any of the private keys are discovered.

French banks were the first to trial a bank card scheme based upon the use of public keys. The trial took place between 1982 and 1984, and began to be introduced nationwide in 1987 (Townend 1995). The scheme's success is demonstrated in the reduction of fraud: Cartes Bancaires[6], the French banking group, report that as a direct result of the replacement of magnetic stripe cards with smartcards, fraud rates fell from 0.27% of total transaction value in 1987 down to 0.02% in 1997 (Flohr 1998).

Modern smartcards cover an extremely diverse range of applications and functionality. At the high-end, smartcard processors are considered to exhibit approximately the processing power of the first IBM PC (Hansmann et al. 2000), employing 32–bit RISC processors at clock speeds of up to 7.5MHz. Storage space on these processors however, is constrained to around 32–64kBytes of non-volatile EEPROM for application storage, and 1–4kByte of volatile RAM (Castellà-Roca, Domingo-Ferrer & Herrera-Joancomart 2000, Praca & Barral 2001). Cryptographic co-processors can be included to speed up the execution of encryption and decryption functions.

Most modern smartcard processors include a number of hardware features designed to prevent fraudulent access to data and internal processes (Rankl & Effing 1997). These include encapsulating the microprocessor to prevent micro-probing of data

[6]Cartes Bancaires Group, Paris, France. Web: `http://www.cartes-bancaires.com`

lines, metallization layers preventing electromagnetic monitoring of memory states, on-chip sensors monitoring against intrusive attacks, and constant-power microprocessor designs making it difficult for an attacker to infer a processor's state from its power consumption. Such features are generally considered to provide good tamper-resistance rather than absolute tamper-proofing, and the effort required to extract private data from a smartcard is considered to be high (Quisquater 1997).

Many high-end cards are programmed using a subset of the Java language, whose details are prescribed by the Java Card specifications.[7] Java Card offers developers the benefits of a structured, type-safe, high-level development language, whilst retaining the security features required on a smartcard. For example, Java Card provides robust firewall mechanisms allowing disparate applications to co-exist on the same card. Other advantages of programming for a Java Card platform are that applications are independent of the underlying hardware, and new or updated applications can be dynamically loaded onto the card.

In short, the modern smartcard is an open, trusted and fully programmable mobile microcomputer.

1.2 The Applications of Smartcards

Modern smartcards are employed in a wide and varied range of applications. At the simplest level of memory cards, these include the pre-payment of goods and services such as public telephone, transportation ticketing, photo-copying, and vending applications. With more complex multi-application cards, a number of such applications can be stored on the same card. For example,Verschuren (1998) describes in detail the functionality of a Dutch nationwide university smartcard project, the smartcard or *Studentenchipkaart*, combines both a conventional visual pass with a number of electronic applications including access control and a selection of pre-payment applications.

At the high-end of smartcard functionality, cryptographic capabilities are involved in digital signature creation providing a mechanism against the repudiation of

[7]Full details and specifications of Java Card are available from `http://java.sun.com/products/javacard/`

document creation and electronic transaction (Rankl & Effing 1997, Hansmann et al. 2000). Further, cryptographic functionality enables strong authentication at a distance, allowing restricted access to web services or information (Verschuren 1998, Bonetti, Ravaioli & Piergallini 2000). As already indicated, smartcards can be used to securely store information, this is illustrated by Engelbrecht, Hildebrand & Jung (1995) and Naszlady & Naszlady (1998), both describing the use of smartcards to hold patient medical records.

1.3 The Requirement for Strong User Authentication

Without strong user authentication, the use of a smartcard for electronic authentication, or for digital signature creation, proves only that the smartcard was involved in the transaction. But smartcards can be lost, stolen or loaned. As a result, mere possession of a smartcard is insufficient to demonstrate the participation of its rightful holder.

The weakness of this approach is apparent, given the possibility of an imposter *borrowing* a smartcard and using it to perform a transaction. In such a case, the legitimate card-holder may find difficulty in disproving involvement. Indeed, the converse is equally troublesome. If a mendacious card-holder were to repudiate a transaction involving his smartcard, then he merely has to claim that the card was stolen.

In the case of pre-payment applications, particularly when a number of applications are loaded onto a card, restricting access by robustly demonstrating identity is desirable. In doing so, the card with its stored values becomes less attractive to thieves, and the holder's investment is offered some measure of protection. With similar reasoning, a smartcard storing medical or financial data should restrict access to only its rightful holder. In essence, data of this nature is personal and access should exclusively be with the owner's consent.

For these reasons, it is desirable to find mechanisms which test the identity of a person before they are able to use a smartcard. It is clear that binding the electronic data of a smartcard and its rightful holder must be strengthened beyond

mere possession. The next section reviews the mechanisms which are currently used to bind a smartcard with its holder.

1.4 Current Approaches to User Authentication

Currently most smartcards are either entirely unprotected or are protected by conventional security mechanisms such as passwords, or personal identification numbers (PINs). This is commonly known as *two factor* authentication, where someone must firstly have possession of the smartcard and knowledge of its associated secret information. This can provide greater security than either passwords or tokens alone. Unfortunately, these *knowledge*-based approaches are weak, because the secret knowledge may be forgotten, or discovered by third parties. Hence, such an approach is no longer considered sufficiently reliable to demonstrate identity (Jain, Hong, Pankanti & Bolle 1997). These approaches all suffer from an inability to distinguish between a legitimate user and an impostor having acquired knowledge or possession of the access method.

As a result there is an increasing interest in making use of biometrics to demonstrate identity to the smartcard. Biometrics refers to the automated identification of an individual using physiological or behavioural characteristics, such as fingerprints or voice patterns. These characteristics are more difficult to lose or steal than passwords, PINs or tokens, and are hence considered to offer greater security (Jain & Pankanti 2001). Biometrics can be used to *identify* an individual from a population of users, or to *verify* the identity of an individual. This thesis is concerned with verification rather than identification, since it is assumed that a smartcard will have one legitimate holder, only. The field of biometrics is comprehensively reviewed in Chapter 2, but for the moment, it is assumed that a biometric system can be divided into three principal components, that is:

- Sensor to capture discriminatory characteristics;
- Comparison module;
- Means of storing representative features.

Almost exclusively, the work on biometric protection of smartcards has split these components between the card and some external resources. For example, at the

simplest level, smartcards are used to securely store the representative features whilst relying upon external means to capture the biometric characteristics and compare live samples with those stored on-card (Phipps & King 1997). This approach is attractive in its simplicity: A moderately featured smartcard is capable of storing biometric data, and a modern PC is quite capable of performing comparison algorithms. Furthermore, the physical requirements of some biometric sensors are not particularly arduous, and may make use of the space available on a desktop. Indeed, a number of manufacturers offer smartcard readers with integrated biometric sensors.[8] Unfortunately, the host PC must be trusted, and live characteristics may be recorded and *played-back* at a later time. This type of attack is known as an *eavesdrop* or *replay* attack. Typically, such an architecture would be used in a situation where one authority has complete confidence and control over all components of the system. Physical access control and time & attendance monitoring are examples of this structure (Adams 2000).

An improvement is to store both the biometric features and perform the matching process on-card. In this instance, the smartcard is still reliant upon external devices for the capture of biometric characteristics, and is hence susceptible to replay type attacks, but has the advantage of executing the sensitive comparison process on its trusted, tamper-resistant processor. The disadvantage here is that some effort must be made in optimising the matching algorithm for the constrained computational resources of a smartcard. Indeed the matching algorithm may be too demanding to execute within a *reasonable* amount of time. However, this approach is an attractive compromise and a number of smartcard companies are beginning to demonstrate on-card fingerprint and voice matching[9,10] (Höjerback 2000, George 2000).

A further disadvantage to the approaches discussed, is that the card-holder may require the flexibility to use her card at a number of disparate locations. If user-verification relies upon a device which is external to the smartcard, then such devices must be present at all points-of-use of the card. As shall be seen in chap-

[8] See for example the Touch430 smartcard reader with integrated fingerprint sensor from Gemplus. `http://www.gemplus.com`

[9] On-Card Fingerprint matching from *Precise Biometrics*, *Miotec*, *iD2 Technologies* and *Fingerprint Cards*.

[10] On-Card Voice recognition from *Domain Dynamics Limited*, Heaviside Laboratories, Cranfield University, Shrivenham, Swindon, England SN6 8LA. Web: `http://www.ddl.co.uk`

ter 2, not all biometrics are suitable for all people, nor indeed all tasks, thereby confounding the problem. By integrating all relevant components on-card, this problem is removed and existing card-reading infrastructure can continue to be used. Moreover, since the sensor is on-card, the potential for eavesdrop and replay attacks is greatly reduced.

The open literature contains very little evidence of embedding biometric sensors onto smartcards, although an early patent (Löfberg 1986) describes the concept of integrating a fingerprint sensor with a smartcard. However, whilst the most compact of modern fingerprint sensors are silicon-based (Tartagni & Guerrieri 1998, Shigematsu, Morimura, Tanabe, Adachi & Machida 1999, Mainguet, Pegulu & Harris 2000), silicon is brittle in its native form. This is contrary to the necessary requirement that a sensor be sufficiently robust and flexible to withstand the typical handling strains to which a smartcard is subjected. By thinning silicon to around 30μm it becomes mechanically flexible (Klink 2000), however, there still exists the possibility of damage from chemical contamination and electrostatic discharge. Protecting a silicon sensor from these hazards will incur increased manufacturing and packaging costs.

Nevertheless, it is believed that Infineon[11] are in the process of developing flexible silicon fingerprint sensors (Grassmann & Karl 2001). If successful, this venture may enable fingerprint sensors to be embedded onto ISO compliant smartcards, although at the time of writing, no such cards have been openly reported. On the other hand, one company[12] claims to have integrated a *self-authenticating* fingerprint module onto a smartcard. Apparently, the module's packaging provides mechanical rigidity for a conventional fingerprint sensor, removing the requirement of flexibility. This device is 1.5mm thick which is significantly in excess of the ISO standard thickness of 0.76mm, and is hence incompatible with standard card readers. To the author's knowledge, there have been no successful attempts to fully integrate biometric sensors onto ISO standard smartcards.

This thesis investigates a class of inherently thin and flexible polymer sensors, called *polymer thick film* (PTF) sensors[13], for the purposes of capturing discrim-

[11]Infineon Technologies AG. München, Germany. Web: `http://www.infineon.com`

[12]Biometric Associates, Inc. Baltimore, Maryland, USA. Web: `http://www.biometricassociates.com`

[13]Although termed *polymer thick films*, these devices are thin in comparison to the thickness

inatory biometric characteristics. PTF technology is an established circuit fabrication method used in the production of flexible conductors and interconnects (Fu, Laing, Shiramatsu & Wu 1981). In addition, the intrinsic properties of thick film materials can be exploited for sensor purposes, for example *piezoresistive* and *piezoelectric* pressure sensors have been demonstrated (Papakostas, Harris, Beeby & White 1998). Chapter 3 provides a review of the current state of the art, and develops the mathematical description necessary to characterise the behaviour of polymer thick film pressure sensors, bonded onto smartcards.

1.5 Scope of Thesis

The focus of this thesis is to investigate mechanisms of demonstrating the identity of a smartcard holder to their smartcard. From the review presented above, it is clear that significant security and convenience benefits are offered by integrating the sensor mechanism and performing algorithmic comparison on-card. This presents challenges in terms of:

- Identifying human characteristics which are sufficiently distinguishing and are appropriate to a smartcard system;
- Devising sensor mechanisms to capture these characteristics, and do so in a manner which is both mechanically and economically compliant with smartcards;
- Finding efficient comparison algorithms which offer good differentiation properties, and can execute within an acceptable period of time on a typical smartcard platform.

The manner in which identity can be demonstrated to the smartcard must exploit some aspect of the interactions between a person and the card, and is hence restricted by the physical dimensions of the card. Chapter 2 presents a detailed review of the field of biometrics and identifies a number of possible methods by which identity could be demonstrated to the card. These include, but are not limited to: fingerprints; finger geometry; the pattern of creases found on the inner surface of the finger; and voice characteristics. Sensor requirements, in terms of spatial

of a smartcard, and can be expected to be $\lesssim 100$ μm.

resolution are assessed, and additionally, the measurands of pressure, temperature and capacitance are found to be useful in the capture of such characteristics.

As indicated, PTF sensors are inherently thin, robust, flexible and low-cost, and as a result offer attractive properties for an integrated smartcard identity verification system. Hence, the field of polymer thick films is reviewed and assessed in Chapter 3. During the course of this chapter, the sensing mechanisms of piezoresistive and piezoelectric polymer thick films are described, and mathematical models of sensors bonded to smartcards are developed. It is found that both sensor types exhibit theoretical sensitivity to normal and planar strains. This is an undesirable property, since an applied load, such as the presentation of a finger, causes flexure of the card and the propagation of planar strain. Chapter 4 assesses this effect, both theoretically and experimentally, and finds that the planar sensitivity of both sensor types is too high to allow more than one independent sensor on card. This work is documented in (Henderson, Papakostas, White & Hartel 2002). On the other hand, flex of a smartcard causes a sensor to experience higher strains than would be the case for a sensor bonded to a rigid surface. This improved sensitivity to singular loads can be exploited, and Henderson, Papakostas, White & Hartel (2001) documents the differences between piezoresistive and piezoelectric sensors in this respect.

Chapter 5 proposes and demonstrates a novel approach to identity verification. Exploiting the force sensitivity of a piezoelectric sensor, bonded to a smartcard, this new approach is founded upon the principles of *keystroke dynamics*, and is first reported in (Henderson & Hartel 2000) and with more detail in (Henderson, White, Veldhuis, Slump & Hartel 2002a). Differentiation based upon keystroke dynamics utilises differences between typing styles of individuals, and much of the work uses only the information of inter-key and key-hold times (Joyce & Gupta 1990, Brown & Rogers 1993, Obaidat & Sadoun 1997, Robinson, Liang, Chambers & MacKenzie 1998). In this proposed approach to identity verification, an individual taps out a self-selected rhythm on a singular pressure sensor, bonded to a smartcard. In common with keystroke dynamics, inter-tap and tap-duration times are used as features. However, the use of a pressure sensor allows the additional feature of pressure amplitude, providing further scope for discrimination.

The approach is demonstrated in an experiment involving a population of 34 in-

dividuals, each tapping their rhythm a number of times. Straightforward vector distance functions are investigated for their discrimination potential, and the best-performing of which is shown to offer an equal error rate of around 2.3%. This is comparable to much of the work on keystroke dynamics, in terms of both performance, method and population size.

This new approach is a combination of knowledge based and physiologically based verification. However recent work at the University of Twente (Henderson, White, Veldhuis, Slump & Hartel 2002b), investigates the use of only the pressure characteristics of each user's tapping style, thereby moving away from a reliance of knowledge content and towards a *sequence-independent* method of demonstrating identity. This work is based upon the same experimental data as (Henderson & Hartel 2000), and an equal error rate of around 7% is reported.

Chapter 6, considers the constrained computational resources of a typical Java Card platform. The algorithms necessary for demonstrating identity are written and optimised for compliance with the Java Card platform. Java card, for example, does not support floating-point arithmetic nor elementary maths functions such as square roots. In conventional use of this platform, for data storage, cryptography and pre-payment applications, such functions are not required. They are, however, for biometric verification. So this chapter develops the necessary integer-based equivalents and algorithmic techniques to allow the vector distance functions of Chapter 5 to be computed.

This final stage in demonstrating suitability for smartcards ensures that the appropriate verification functions can execute within an acceptable time. It is found that the most discriminating distance approach will perform the one-off enrolment of an individual within 3.1 seconds, and more importantly, will verify their identity on a per-transaction basis within 0.12 seconds. These times are well within reasonable bounds, and this work is documented in (Henderson, White & Hartel 2001).

Chapter 7, sums up the work of this thesis, and considers fruitful areas of further research.

1.6 Original Contribution

Biometrics and the use of smartcards for authentication and non-repudiation mechanisms, are both emerging fields, hence, there is very little published work in the overlapping region between the two. As indicated above, tentative advances have been made in the use of smartcards to securely store biometric information, and to perform on-card biometric matching. However, with the exception of a number of conceptual patents (see for example Löfberg (1986) & Willmore (1994)), and marketing literature (Grassmann & Karl 2001), there is little evidence of efforts to embed biometric sensors onto smartcards.

The work contained herein considers this problem, firstly from the viewpoint of finding discriminatory characteristics which are suitable for smartcards, and secondly, from the perspective of finding appropriate sensing mechanisms with which to capture these characteristics. Polymer thick film sensors are identified as being mechanically and economically compliant with smartcards, and their potential to capture human interactions is assessed. It is believed that this is for the first time that PTF technology has been assessed in this respect, and indeed, the work of Henderson, Papakostas, White & Hartel (2001) has been cited as "...[an] innovative use of thick film sensors..." (Bogue 2002).

During the course of this work, two novel PTF sensing mechanisms have arisen. The first is a polymer thick film realisation of a simple capacitance sensor, and is analogous the the sensing elements of capacitive silicon fingerprint sensors. Although this is proposed in Section 3.4, a discussion on its further development and testing is necessarily postponed until the Further Work section of Chapter 7. The second mechanism is described in Section 4.9, and involves the formation of a dome structure within the substrate underneath a conventional PTF pressure sensor. This has the advantage of allowing increased strain to be transferred to the sensing film, even when the substrate is constrained in its out-of-plane direction. This mechanism was serendipitously discovered, and further development and applications are proposed in Chapter 7.

The most important original contribution of this thesis, is the novel approach to identity verification, proposed and demonstrated in Chapter 5. This is important because all aspects of this approach, including the sensor, the manner of interac-

tion, and the computational resources required for verification, fully comply with the constraints imposed by a smartcard. It is believed that Henderson & Hartel (2000) is the first report of such a system in the open literature.

Chapter 2

Biometrics

2.1 Introduction

Biometrics refers to the automated identification (or identity verification) of a person based upon their distinctive physiological (*something you are*) or behavioural (*something you can do*) characteristics (Miller 1994, Clarke 1994, Jain, Bolle & Pankanti 1998, Hollingum 1999). For example, physiological traits which have successfully been used to identify individuals include fingerprints, hand geometry, iris pattern, retinal pattern and facial features (Jain et al. 1997, Miller 1994, Jain, Ross & Pankanti 1999, Wildes 1997, Davies 1994, Phillips, Moon, Rizvi & Rauss 2000). Whilst behavioral traits include voice characteristics, handwritten signature features, and the manner with which people type on a keyboard (Furui 1997, Newham 2000, Joyce & Gupta 1990).

Conventionally, automated personal identification is based around either a *knowledge-based* or a *token-based* approach (Miller 1994, Davies 1994, Ashbourn 2000, Jain & Pankanti 2001). Knowledge-based identification relies upon *something you know* – an individual identifies himself by demonstrating his knowledge of some predetermined secret, such as a password or a Personal Identity Number (PIN). Whilst on the other hand, token-based identification requires presentation of *something you have*, which could be for example, a key, smartcard or identity card.

Such approaches to identification are attractive because of the simplicity and ease

with which they can be integrated into existing systems – the widespread use of passwords for computer access is a good case in point. Unfortunately they represent weak demonstrations of identity, in that passwords may be forgotten or discovered by third parties, and tokens may be lost, stolen or copied. Combining both approaches by protecting a token with some associated secret knowledge, such as a banking card and PIN, offers some improvement in that an impostor must gain possession of the token and discover its associated knowledge. But this approach still does not demonstrate the presence of the authorized party, merely that the possessor of the token is also in possession of its associated knowledge. Furthermore, secret knowledge is often written down in obvious places, such as diaries or even upon the card itself (Parkes 1991). Such actions further weaken an already weak approach to identification.

Biometrics differs from conventional knowledge-based or token-based identification, in that the biological characteristics of a person cannot be easily lost, shared or misplaced (like passwords or keys). Hence, biometrics are generally considered to offer a more reliable security mechanism than mere demonstration of *something you know* or *something you have* (Jain & Pankanti 2001).

This thesis is concerned with finding a mechanism of demonstrating a smartcard-holder's identity *to the smartcard*, such that the smartcard will not function unless the user's identity has been satisfactorily demonstrated. From the outside-world's perspective such a system can represent an authenticated electronic key which can be used anywhere, and only by the legitimate holder of the smartcard. This situation is depicted in Figure 2.1.

Positive identification of a person in this manner satisfies a combination of two approaches to identification, namely that the card must be possessed (*something you have*) and that the biometric test must be satisfied (*something you are*). In Chapter 5 a novel identity verification method which is wholly appropriate to smartcard integration is proposed and demonstrated.

In seeking an appropriate identity verification mechanism, we look to biometrics-based techniques for inspiration. This chapter presents an overview of the fundamental principles of biometric discrimination, introduces terminology and considers factors affecting the performance of a biometric system. A detailed review of the current biometric state-of-the-art is presented, during which it will become en-

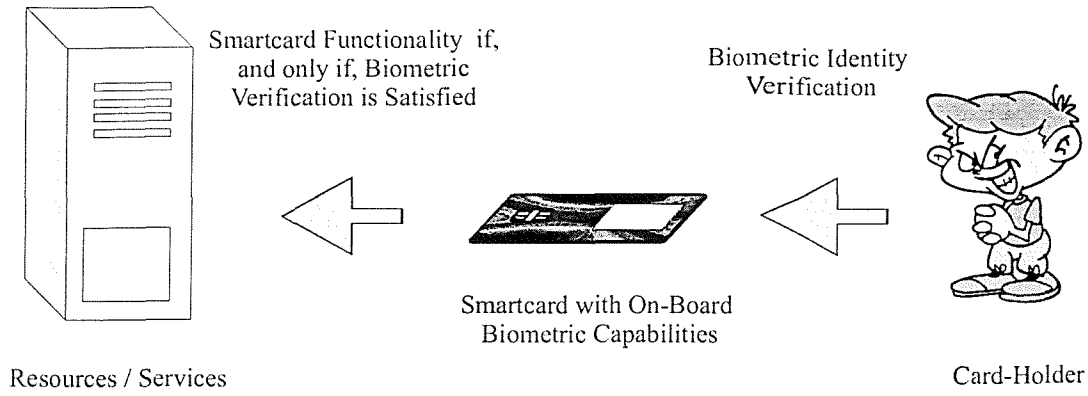


Figure 2.1: Smartcard with Embedded Biometric Capabilities: The smartcard functions only if the holder's identity has been robustly demonstrated. This is in contrast to the operation of current smartcards.

tirely evident that the physical properties of a smartcard restrict both the human characteristics which can be presented, and the manner with which these can be captured. A full discussion on the relevance of biometric techniques to the problem of identity verification on a smartcard concludes this chapter.

2.2 Biometric Discrimination

Any human physiological or behavioural characteristic can be used to identify an individual, provided that the requirements of *universality*, *uniqueness*, *persistence* and *collectability* are met (Jain et al. 1997). Universality requires that every person should exhibit the characteristic. Uniqueness implies that there must be sufficient variability within a group of people, such that no two people are exactly the same in terms of the characteristic. Persistence requires that the characteristic be time invariant, whilst collectability means that the characteristic must be accessible to quantitative measurement. Additionally, a practical biometric system should exhibit satisfactory performance in terms of accuracy and speed, should be perceived to be acceptable to its users, and be resistant to circumvention by fraudulent means.

There are key differences between the use of physiological and behavioural characteristics. Physiological traits are, barring injury, unlikely to change over time, whilst behavioral traits may exhibit frequent variations depending upon the psy-

chological state of an individual.

2.2.1 Biometric Sensors

A principal aspect of any biometric system is a means of capturing live biometric samples. The sensor mechanism which achieves this, may be a dedicated sensor explicitly designed to capture biometric samples, such as fingerprints (Tartagni & Guerrieri 1998), or it may exploit the availability of existing devices, as is the case for speaker recognition over telephone networks (Lamel & Gauvain 2000).

The desire to embed a biometric system on a smartcard places particular constraints on the sensor. As indicated in Chapter 1, the sensor must be mechanically compliant with the smartcard – a requirement clearly illustrated when one considers that usual handling causes flexure of the card. In addition, the sensor should be sufficiently robust to withstand sheer stresses associated with insertion and removal of the card from wallets and readers. For these reasons, and that of low cost, polymer thick film sensors are identified as appropriate for integration with smartcards.

It is clear that properties, such as sensitivity, cross-sensitivity, and spatial resolution, impose a limit on the amount of information which can be captured. Hence, Chapter 3 reviews the current state of polymer thick film sensors, whilst Chapter 4 investigates the properties of such sensors when bonded to smartcards.

The choice of an appropriate identity verification method may be strongly influenced by the sensing resources available.

2.2.2 Modes of Operation

A biometric system can operate in one of two modes – either *identification* or *identity verification.* Identification is the process of answering *"who am I?"* and requires a subject's biometric characteristics be compared to an entire database of enrolled users. Identification is hence, a *one-to-many* comparison. Verification, on the other hand, represents a *one-to-one* matching process, and attempts to answer the question *"am I who I say I am?".* Verification requires a person firstly, to make

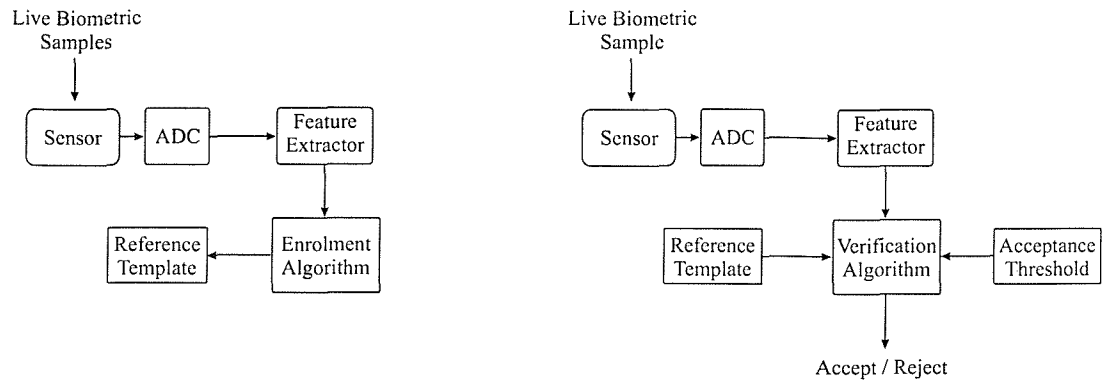
a claim of their identity, then provide a biometric sample which is captured and compared to a previously stored sample associated with the identity being claimed. If the two are *sufficiently similar*, then the identity being claimed is accepted as valid.

It is the mode of identity verification which is of concern throughout this thesis. The assumption is made that a smartcard has only one legitimate holder, and it is this identity which is associated with the data stored on card. Hence, an embedded verification system serving to test the identity of someone in possession of a smartcard, should reject all with the exception of the legitimate holder.

The functionality of a biometric verification system can be split into two principle algorithmic cores, namely the *enrolment* and *verification* modules. The enrolment module is responsible for the one-off process of generating and storing a representative template of an individual's biometric characteristics. Whilst verification occurs on a *per-transaction* basis, and compares live characteristics to the stored template. If the two are sufficiently similar, according to a predetermined threshold, then the live sample is accepted.

Upon presentation and capture of an individual's biometric sample, a *feature extraction* process performs the task of extracting pertinent, time-invariant characteristics. Using the example of hand geometry, such features may include finger lengths, finger widths, and the width of the palm (Zunkel 1998). The feature extraction process is common to both enrolment and verification modules, as shown in Figure 2.2.

The computational requirements of the matching process depend largely on the complexity of the template and the matching algorithm. Hence, the computational resources of the matching processor determine the time required to execute the verification algorithm. This is of particular concern when dealing with the constrained resources of the smartcard. As a result, processing considerations will be considered in significantly greater detail in Chapter 6.



(a) Enrolment Process (Performed Once on Set-up)

(b) Verification Process (Performed on a Per-Transaction Basis)

Figure 2.2: Generic Biometric System

2.2.3 Performance Considerations

Most biometric characteristics will exhibit some variance, either as an inherent aspect of the characteristic itself (for instance the variations found in handwritten signatures), as a physical consequence of the measurement process (fingerprints may be captured when the fingertip makes incomplete contact with the sensor, the sensor's surface may be dirty, or the sensor may contribute random noise to the measurement), or as a result of feature extraction poorly representing the invariant characteristics of the biometric. All of these sources of variance degrade the matching accuracy and hence, the performance of the system. See (Jain & Pankanti 2001) for further details on this matter.

Due to these variations, the process of biometric verification does not give an absolute accept or reject decision. Rather, a genuine claimant may be accepted, and an impostor rejected, with an empirically determined confidence. Conversely, the system may commit errors by falsely *accepting* an impostor or falsely *rejecting* a genuine user, again with an empirically determined probability. Obviously, these errors should be minimised in the interests of overall accuracy of the system.

The pattern-matching component of a biometric system compares the biometric features which are expected to remain invariant in different *presentations* of the biometric quantity. However, due to the reasons stated above, some variance is to be expected. This is taken into account by setting an appropriate acceptance

threshold. For example, if representative features are used to create a *pattern vector*, $\mathbf{X}$, and $\mathbf{R}$ is the mean feature vector from one person, then $\mathbf{X}$ will be accepted as being *sufficiently close* to, $\mathbf{R}$ if $\|\mathbf{R}-\mathbf{X}\| < \theta$ where θ is the acceptance distance and the magnitude operator, $\|\cdot\|$, is an appropriate vector distance function (for example Euclidean norm).

Figure 2.3 shows two possible feature distributions: One from a population of impostor, the other from a genuine user. As the acceptance threshold is increased, the probability of false rejections decreases but the chance of false acceptance increases. It is the goal, then, of a biometric system to use characteristics which are sufficiently different for each individual within a population, and to find representations which allow these differences to be maximized.

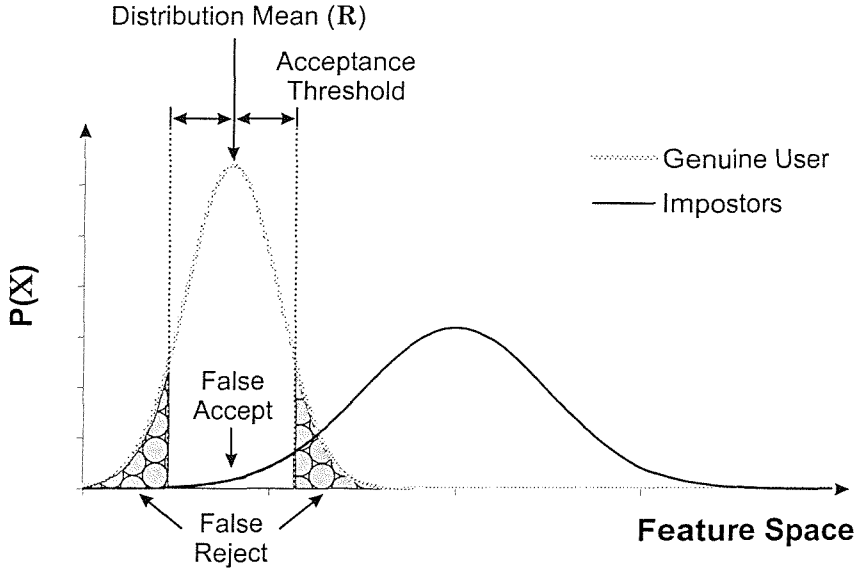


Figure 2.3: Possible Genuine and Impostors Feature Distributions

The performance characteristics of a biometric verification system are typically given in terms of *False Acceptance Rate (FAR)* and *False Rejection Rate (FRR)*. The FAR is defined as the probability that a biometric system will fail to reject an impostor, and is stated as

$$FAR = \frac{Number\ of\ false\ acceptances}{Number\ of\ impostor\ attempts}. \tag{2.1}$$

The FRR, on the other hand, can be expressed as the probability that a biometric system will fail to verify the legitimate claimed identity of an enrolee. It is stated

as

$$FRR = \frac{Number\ of\ false\ rejections}{Number\ of\ genuine\ attempts}. \tag{2.2}$$

FAR and FRR clearly vary inversely with each other. By way of illustration, consider the following, a system exhibiting a false acceptance rate of zero is required. This may be achieved simply by blocking access to all users – a system with a false rejection rate of one will have been created. Conversely, designing a system to falsely reject no legitimate users is plainly a matter of allowing access to all users, legitimate or otherwise. A system with a false acceptance rate of one is the result.

Figure 2.4 shows the generalised error characteristics of a biometric system. The crossover point of both error curves is known as the *equal error rate*, although this rate is not always quoted by manufacturers of biometric systems, it is the author's belief that the equal error rate gives a more balanced view of the system than mere false acceptance rate alone.

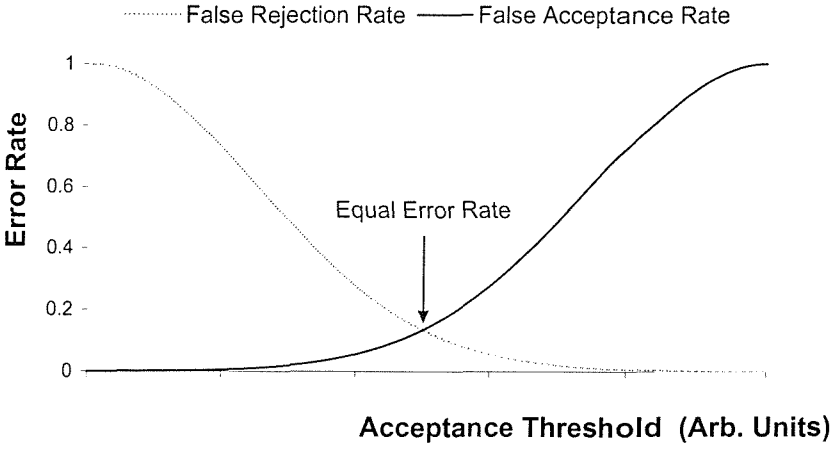


Figure 2.4: Generalised Biometric Error Rates

One can dispense with the detail of acceptance threshold by plotting FAR against FRR, which is known as the *Receiver Operating Characteristics* or *ROC* curve of the system. Figure 2.5 shows a generalised ROC curve.

It should be emphasised that performance characteristics of a biometric system rely inherently upon the matching algorithm and the acceptance threshold. Access to either of these allows the performance level to be altered. This point represents a

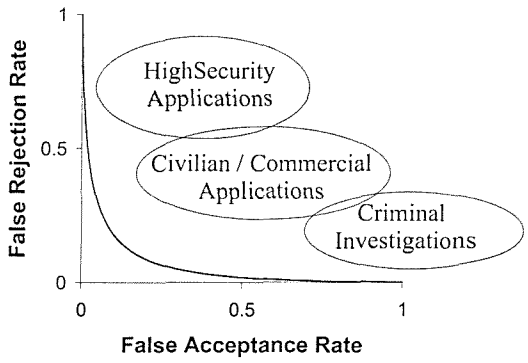


Figure 2.5: Generalised ROC Curve: Typical error ranges are given for different applications.

principal benefit of performing biometric matching on-card, in a secure and trusted manner.

2.3 Existing Biometric Methods

This section provides a review of the biometrics state-of-the-art, and highlights the approaches which make use of characteristics suitable for an on-card verification system. Where appropriate, specific aspects of sensing mechanisms are given, thereby allowing comparison with the properties of polymer thick film sensors. System performance and algorithmic details are provided where available and relevant.

Unfortunately much of the work on biometrics remains proprietary or is commercially sensitive, and as a result one must look to marketing literature or patents for information. Whilst this may adequately serve to provide an outline of each biometric method, such reference sources can rarely be relied upon for peer-reviewed, balanced analysis. However, two recent independent investigations into the performance of biometric systems have been made (Funk, Finke & Daum 2000, Mansfield, Kelly, Chandler & Kane 2001), and will be quoted where appropriate.

The study by Mansfield et al. (2001), for the UK National Physical Laboratory (NPL)[1], assessed biometric systems for the verification of unhabituated users. Two

[1]National Physical Laboratory, Teddington, Middlesex, UK. Web: `http://www.npl.co.uk`

hundred self-selected individuals were involved – each participating in one enrolment and two verification sessions, separated by an interval of one month. Face, fingerprints, hand geometry, hand vein, iris and voice verification systems were assessed in this study.

The work by Funk et al. (2000), for the German Information Security Agency[2], investigated the suitability of biometric systems for everyday identification and verification operations. Under investigation were optical and silicon based fingerprint sensors, face recognition systems, signature verification systems, a hand geometry recognition system and an iris based verification system. This study involved forty volunteer participants, each using all ten systems on a daily basis.

Funk et al. (2000) report only false rejection rates without corresponding false acceptance rates for the verification systems under review. This is unfortunate since judicious selection of acceptance threshold can achieve any desired false rejection rate, as can be seen from Figure 2.4. However, both quantities are reported for the identification systems under investigation, and will be quoted appropriately.

It seems reasonable that an integrated smartcard biometric makes use of the physical interaction between card and user. For this reason, particular attention will be given to hand or finger based biometrics, whilst only cursory commentary will be given to other methods for completeness. At the end of this chapter a full discussion on the relevance of current biometrics to smartcard integration ensues.

2.3.1 Fingerprints

Fingerprint characteristics is perhaps the most mature area of personal identification, being used since the beginning of the twentieth century for criminal and forensic applications, and more recently in the automated identification of individuals (Jain et al. 1997).

Fingerprints are the visible patterns of ridges covering the fingertip regions of the finger. The outer region of the fingertip is composed of two layers, the outermost being a layer of dead skin cells, called the *epidermis*, whilst the inner layer is a living conductive layer called the *dermis* (see Figure 2.6). It is the visible pattern

[2]Bundesamt für Sicherheit in der Informationstechnik, Bonn, Germany. Web: `http://www.bsi.de`

of ridges and valleys which makes up the complex pattern of the fingerprint, and the ridge separation distance is reported to be around 330-400 μm (Cummins 1964)

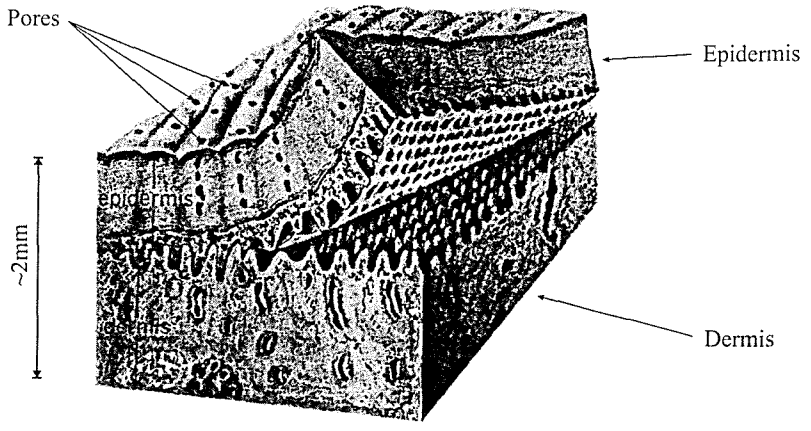


Figure 2.6: Finger Ridged Skin (After Cummins (1964))

The fingerprint pattern begins to form at about twelve weeks of gestation. Undulations begin to appear on the inner surface of the epidermis, later developing into a structure of ridges, furrows and sweat glands. As a child grows, the structure grows, but the pattern geometry is preserved unchanged (Holt 1968). Further evidence to this effect was provided by (Galton 1892) who recorded fifteen sets of fingerprints then re-recorded and compared each set, *a number of years later.* He observed that in all cases the form of the pattern and the detailed structure of the ridges remained unchanged. One striking example is given in the case of a woman who, as a young child was fingerprinted by Galton (in 1890), and then again by Penrose (1969) in 1961. Over seventy years later, the ridge pattern and detail remained preserved. Numerous subsequent studies have strengthened these conclusions (Holt 1968).

Galton (1892) described fingerprints in terms of their gross pattern of *loops*, *arches* and *whorls*, also making note of small local features called *minutiae*, which include *ridge endings*, *ridge bifurcations*, *delta-points* and *islands*. Although fingerprints may be generally classified by pattern type and ridge counting methods, identification of an individual is subject to matching of the minutiae geometry of their prints. Figure 2.7 shows an inked print of one of the author's fingers[3], upon which

[3]The fingertip was coated with LaserJet toner, then dabbed onto paper. The resulting print

a number of minutuae features are highlighted.

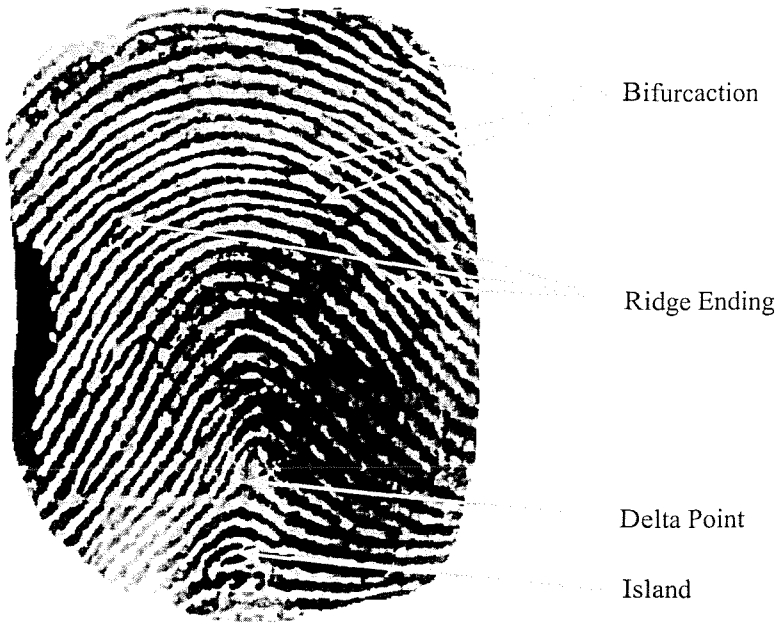


Figure 2.7: Example of an Inked Fingerprint: Finger ridges appear darker than valleys.

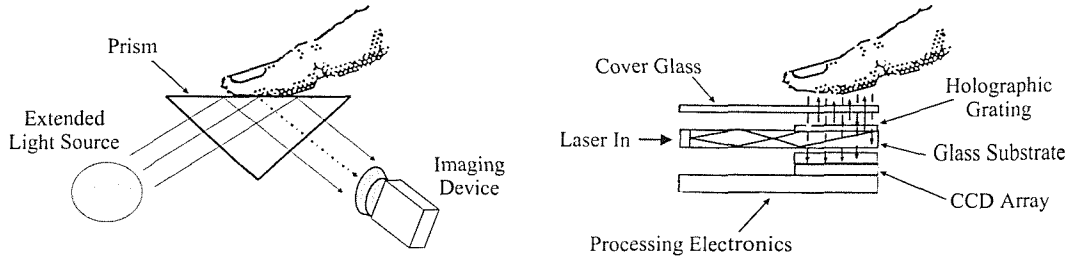
There are a number of ways of capturing fingerprints including optical, capacitance, pressure and thermal based approaches. Of these methods optical capture is perhaps the most intuitive, with a number of variants reported in the literature (Coetzee & Botha 1996, Drake 1997). The most straightforward approach is simply to use a direct optical scanner.[4] However there are a number of problems with this approach since the pressure of a finger-tip placed on a scanner reduces the contrast between ridges and valleys, and further, circumvention using a photograph is likely to succeed.

An improved optical capture method is that of *frustrated total internal reflection* (Coetzee & Botha 1996). This scheme relies upon light internally reflecting from an inside face of a glass prism, and onto an optical imager. A fingertip is placed upon the outside of a reflecting face, and at positions of ridge contact the refractive index of the glass-air boundary are changed thereby frustrating total internal reflectance. Hence, ridges appear dark and valleys appear as bright regions. Such a scheme is presented schematically in Figure 2.8(a).

was captured using a flatbed scanner with 8-bit greyscale at 600dpi.

[4]For example, see the BioMouse from ActivCard, Inc. Fremont, CA, USA. Web: `http://www.activcard.com`

The state of the art in optical fingerprint imaging is a holographic fingerprint sensor (Drake 1997), a schematic of which is presented in Figure 2.8(b). The bulky prism and extended optics of the previous device are replaced by a holographic grating of an extended light source, sandwiched between a glass cover and an optical CCD sensor. The grating is illuminated by a diode laser, appearing as an extended source to the glass cover. Light is then reflected from the finger and onto the imaging sensor. This is a compact optical device, in the order of 1 to 2 cm high.



(a) Frustrated Total Internal Reflection Approach

(b) Holographic Sensor

Figure 2.8: Optical Fingerprint Sensors (After Drake and Fiddy (1996))

The most common contemporary approach to sensor design is silicon based, offering a significantly more compact alternative to optical methods. An additional advantage is the ability to integrate image processing circuitry to improve image quality on the same die as the sensor. Jung, Thewes, Goser & Weber (1999), describes an approach to integrating feature extraction functionality on the sensor die, whilst Shigematsu et al. (1999), reports on the design of a single-chip sensor with verification functionality.

Silicon based capacitive sensors have been described by Young (1997) and Tartagni & Guerrieri (1998). Figure 2.9 outlines one possible approach in which a reference voltage is applied to the electrode of each pixel, and a charge is developed across the insulating layer. The resulting charge, Q_{Pixel}, is proportional to the capacitance between the pixel electrode and the (conductive) dermal layer of the fingertip, C_{Pixel}. That is

$$Q_{Pixel} = C_{Pixel}\, V_{Ref}. \tag{2.3}$$

Upon completion of the charging cycle, the charging switch is opened, and the discharge switch closed allowing charge to be transferred to a reference capacitor. The resulting voltage ($V_{Sensor} = \frac{Q_{Pixel}}{C_{Ref}}$) may then be extracted and measured. A suitable array infrastructure, such as charge-coupling (Tartagni & Guerrieri 1998) or switched X-Y scanning (Young 1997) is necessary to localise charge and allow extraction for measurement.

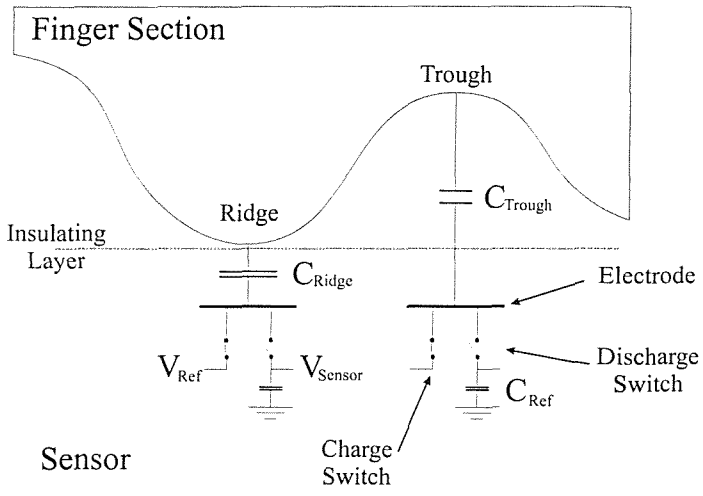


Figure 2.9: Capacitive Measurement of Finger Ridge Pattern

Figure 2.10(a) shows the capacitive sensor from Infineon[5]. This device has 224×288 pixels (of dimensions 50×50μm), offering a spatial resolution of 513dpi with an 8-bit greyscale pixel depth.

Pyroelectric materials have been employed in the fabrication of thermal fingerprint sensors (Mainguet et al. 2000), where temperature differences between the finger and sensor are captured and used to create a fingerprint image. In this scheme, heat is drawn away from the sensor in places of finger contact. The sensor detects this change in temperature and is hence able to record the position of ridges. This scheme is novel in another respect: Rather than employing a square sensing matrix, the sensor described by Mainguet et al. (2000) (marketed as the 'FingerChip'[6]), is a rectangular array of 8×280 pixels, each being 50×50μm and offering 8-bit greyscale image depth. A fingertip is swept across the sensing array and a series of rapid image samples are captured and reconstructed to create a complete image

[5]Infineon Technologies AG St.-Martin-Strasse, 81669 München, Germany. Web: `http://www.infineon.com`

[6]FingerChip sensor from Atmel, Grenoble, France. Web: `http://www.atmel.com`

of the fingerprint. This scheme has the obvious advantage of requiring a smaller sensor, but requires sufficient computational resources to reconstruct the image. The thermal sensor offered by Atmel is shown in Figure 2.10(b).

(a) Capacitive Sensor (Infineon) (b) Thermal Sensor (Atmel)

Figure 2.10: Silicon Fingerprint Sensors

It is possible to capture a finger ridge pattern using an array of pressure sensors (Edwards 1984). Here, a CCD array is coated with a layer of piezoelectric material, which generates charge in response to an applied pressure. Fingertip ridges exert greater pressure than valleys, and thus a fingerprint image can be captured. Another method of capturing finger ridge pressures is given in the micromachined pressure sensor approach of Rey, Charvet, Delay & Hassan (1997). Fingerprint pressure sensors do not seem to have gained the popularity of the capacitive and thermal approaches, as outlined above.

A scanning resolution of 500dpi is widely considered to be a minimum requirement for the capture of fingerprints, and this necessitates a pixel pitch of $\sim 50\mu$m (Lee & Gaensslen 1991).

The most common approach to identification and verification, using fingerprints, relies upon the position and orientation of minutiae features (Jain et al. 1997). Ridges must be detected and *binarized* such that the captured grey-scale pixels are reduced to either black or white values. Due to the effect of noise, dirt, or non-uniform finger placement, the binarized image may contain *holes* and *speckles*. These should be removed before ridges are *thinned* to facilitate minutiae extraction. Straightforward algorithms exist for minutiae detection (for example see Jain et al.

(1998)), which should be applied at this stage. The matching of minutiae between enrolment and live fingerprints results in an acceptance decision. Details of these processes are beyond the scope of this thesis, but the reader is directed to Jain et al. (1997) for greater depth.

A Note on Fingerprints

Since the beginnings of forensic use of fingerprints it has been widely assumed (based upon empirical experience) that no two fingerprints were identical (Lee & Gaensslen 1991). However, Pankanti, Prabhakar & Jain (2001), investigate minutiae distribution and establish a correspondence between two arbitrary fingerprints. It is concluded that false associations may occur, particularly in the case of automated matching, and the probability of a false correspondence between two fingerprints is estimated to be around 5.86×10^{-7}, when 12 minutiae points are used. This analysis does not consider any genetic or environmental correlation within a population, which is expected to significantly reduce the degree of uniqueness within a population.

Furthermore, van der Putte & Keuning (2000), prescribe a straightforward method of copying fingerprints either overtly with the aid of the person belonging to the fingerprint, or covertly making use of latent prints. In both cases, and with a minimum of effort, Van der Putte, has shown that silicone-rubber fingerprint copies can be made and demonstrated that a number of commercially available fingerprint sensors are readily circumvented.

A number of sensor manufacturers claim to improve security by including *liveness* tests in their sensors. These tests may include temperature, conductivity, and dielectric properties, but as Van der Putte points out, fingerprint sensors are required (and designed) to function under a very wide range of operating conditions, and in doing so circumvention is made easier. For example, the resistance of a fingertip may vary from several MΩ during cold, dry weather, and may drop to a few kΩ during summer. It is claimed that a drop of saliva decreases the resistance of silicone sufficiently to circumvent a conductivity test.

Van der Putte's work involved state-of-the art (in the year 2000) fingerprint sensors and included optical, capacitive silicon and thermal silicon sensors. With some

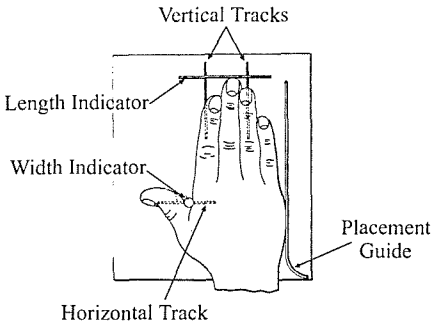
basic skills, van der Putte estimates that covert creation of latex prints requires about 8 hours, whilst overt copying would take just *a few hours*. Whilst the effort involved is non-trivial, it is still well within the reach of a motivated impostor.

Nevertheless, fingerprints still exhibit a large degree of measurable uniqueness. The performance study by NPL found that the fingerprint sensor and matching algorithm from Infineon offered an equal error rate of ~3%, whilst the BSI study reports a false identification rate of 6.5% at a false rejection of 5.7%. Whilst these performances are (in common with all biometrics) not absolute, it is the author's opinion that fingerprints will continue to be a useful tool in both forensic and security applications. It is likely that other less-mature biometrics (particularly those based on external physiology, such as hand-geometry, finger-crease pattern and palmprints) will suffer similar characteristics. Acknowledging the limitations of a particular method will enable informed decisions to be made, allowing their useful exploitation in appropriate circumstances.

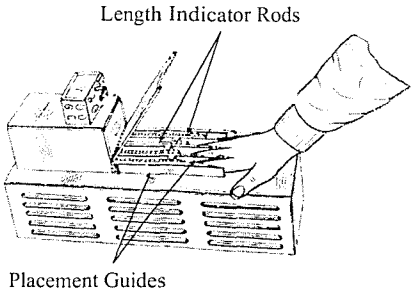
2.3.2 Hand Geometry

Discrimination on the basis of hand geometry relies upon individual differences in the length, width and height of fingers, and palm width (Zunkel 1998, Jain et al. 1999). The use of hand geometry as an automated means of identification was first proposed in the patents of Ernst (1971), Miller (1971) and Jacoby, Giordano & Fioretti (1972). The early systems of Ernst (1971), and Miller (1971), were electro-mechanical in nature, whilst the proposition of Jacoby et al. (1972), introduced the approach of electro-optic sensing of hand characteristics. Figure 2.11 provides schematics of these early systems.

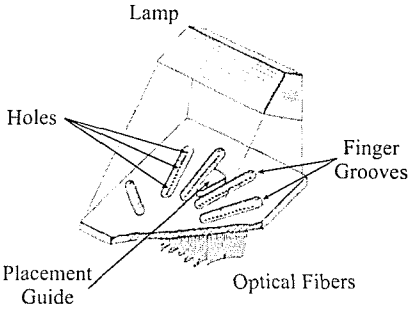
These three approaches measured 2-dimensional planar characteristics of the hand: The system of Ernst measures only the length and breadth of the hand as a whole; Miller's approach was to measure individually the length of each of a person's four fingers, whilst Jacoby's approach was to record the length of all fingers and thumb of one hand. The sensing method employed by both Ernst and Miller was for the hand to physically move resistive contacts, hence, finger lengths were deduced by resistance measurements. With the alternative approach of Jacoby, fingers rest on perforated guide slots, and are illuminated from above. Presence of a finger



(a) After Ernst (1971)



(b) After Miller (1971)



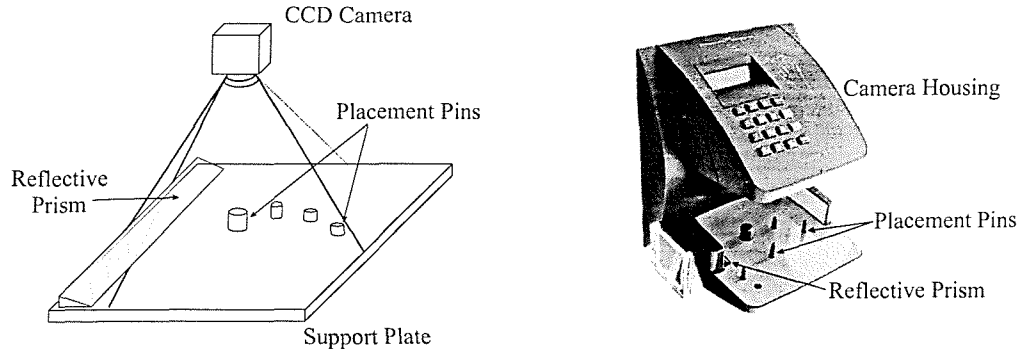
(c) After Jacoby et al. (1972)

Figure 2.11: Early Hand Geometry Recognition Systems

blocks light, and hence finger length can be measured. All three patents provided electro-mechanical means of storing and retrieving hand characteristics on a card, enabling the systems to be used in identity verification mode.

Modern hand geometry systems use CCD cameras to capture both planar and side views of a person's hand, and image processing enables the extraction of features upon which discrimination is based. These typically include finger and thumb lengths, the width and thickness of each finger at one or more positions, and the thickness of the hand at one or more positions. Figure 2.12 shows both a schematic of this approach and a commercially available system.

Whilst the details of most hand geometry systems remain proprietary, some open



(a) Hand Geometry Schematic, After Sidlauskas (1988).

(b) Commercially Available System (Recognition Systems Inc.)

Figure 2.12: Modern Hand Geometry System

research has been recently published. For example, Jain et al. (1999), describes an attempt to design a verification system, broadly of the form described in Figure 2.12(a), using image processing techniques to measure finger length, finger width, hand width and hand thickness. Visual inspection of Jain's results reveals an equal error rate of around 7%, this however, does not include approximately 28% of captured samples which were discarded due to inconsistent hand placement. Sanchez-Reillo, Sanchez-Avila & Gonzales-Marcos (2000), reports upon a similar investigation with 20 users each providing 10 samples, false acceptance and rejections rates of 6.6% and 9%, respectively were calculated.

In an attempt to improve performance, Jain & Duta (1999), describes an approach based upon a measure of hand *shape*, rather than the extraction of explicit features, as reported above. With a false acceptance rate of 2%, Jain and Duta report a 3.5% false rejection rate. By visual inspection of their results, an equal error rate of just over 2.5% is estimated, representing significant improvement. Lay (2000), describes an approach to hand shape recognition, in which the hand is illuminated by a sinusoidally varying intensity pattern, which is claimed to facilitate shape extraction. In a verification experiment involving 100 participants, Lay reports an equal error rate of around 1%.

According to Miller (1994), commercially available systems offer cross-over error rates of around 0.2%, however no details of the testing regime are given. Mansfield

et al. (2001) reports an equal error rate of around 1.5% for the HandKey system (Recognition Systems Inc.[7]), whilst Funk et al. (2000) found a false acceptance rate of 2.2% at a false rejection rate of 4.4%, for the same system.

Hand Geometry is widely considered to provide sufficient discrimination for identity verification rather than identification (Zunkel 1998, Jain et al. 1999, Ashbourn 2000), and the results presented above are aligned with this view. Indeed, as a method of identity verification, hand geometry is widely used, and its most common application is in time and attendance monitoring. Modern hand geometry systems may interface directly with payroll systems, reducing processing costs and virtually eliminating fraud due to clocking-in on the behalf of others (a phenomenon commonly termed *buddy-punching*).

Additionally, hand geometry systems have been used successfully in access control applications. For example, access to the athlete's village during the 1996 Olympic Games was controlled by means of hand geometry systems, and throughout the duration of the games, over 65000 people were enrolled for verification (Ashbourn 2000). The American Immigration and Naturalization Service's Passenger Accelerated Service System (INSPASS), uses hand geometry to automatically verify the identity of frequent, low-risk travellers (Dell & Bunney 2001). The system is installed in a number of airports across the States, and in the year 2000 processed over two-hundred and eighty thousand inspections.

Due to the physical bulk of current hand geometry systems, they tend not to be well suited to protection of PCs or network access. Additionally, it has been suggested that hand geometry systems may be circumvented using a simple cardboard silhouette (Sidlauskas 1988), although the International Biometric Group[8] suggest that an entire cast of an enrolled hand would be required. Furthermore, hand geometry may not be suitable to those with limited hand movement, for example severe arthritis sufferers. Nonetheless, in terms of the applications described above, hand geometry may be considered a mature and stable biometric. Recognition Systems Inc., for example, claim to have deployed over 55000 units worldwide.

In an attempt to reduce the bulk of the hand geometry systems described above,

[7]Recognition Systems Inc. Campbell, CA, USA. Web: `http://www.handreader.com`
[8]International Biometric Group, New York, USA. Web: `http://www.biometricgroup.com`

the patent of Harkin (1998) describes a planar sensor capable of capturing the outline of a hand. This sensor is effectively a large scale, low resolution version of the fingerprint sensor proposed by (Young 1997), and detailed above. Indeed, Harkin proposes the provision of an area of high resolution sensing elements corresponding to the position of the fingertips, such that fingerprints may additionally be captured. To the author's knowledge, this approach has not been developed further.

2.3.3 Hand Vein Pattern

The pattern of veins on the back of the hand is claimed to be unique and to offer the possibility of detecting the *liveness* of the hand (Rice 1985). Furthermore, since veins reside below the skin it is believed that vein patterns are more difficult to forge than conventional hand geometry (Lockie 2001b).

Hand vein systems are physically similar to those used for capturing hand geometry, as shown in Figure 2.12, consisting of a hand grip upon which the user's hand is placed, infra-red illumination source to improve contrast between vein and skin, and an imaging device to record the pattern (further details are provided in Clayden (1998)). Image processing techniques are then used to extract the vein pattern for comparison. Figure 2.13 shows the vein pattern from three different hands.

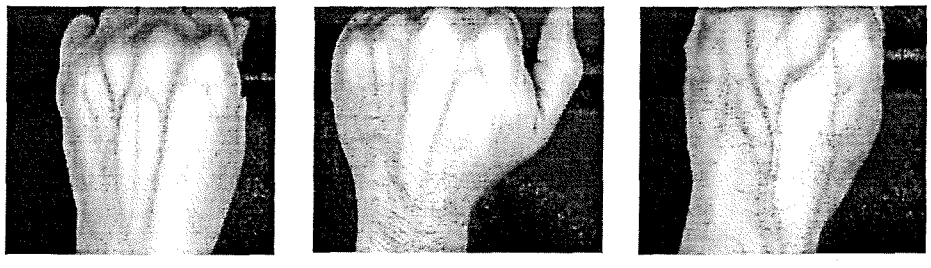


Figure 2.13: Vein Patterns from Three Hands (After NeuSciences Ltd.)

Hand vein systems are currently manufactured by two companies: NeuSciences[9] and BK Systems[10], and are currently deployed in a number of access control and attendance monitoring applications (Choi 2000).

[9]NeuSciences, Southampton, United Kingdom. Web: `http://www.neusciences.com`
[10]BK Systems, Seoul, South Korea. Web: `http://www.bk12.com`

Although BK Systems claim a false acceptance rate of 0.0001% at 0.1% false rejection, independent testing of hand vein recognition by NPL revealed an equal error rate of around 10% for a prototype system from NeuSciences (Mansfield et al. 2001).

Advanced Biometrics Inc.[11] have proposed a similar scheme based upon the vein pattern found on the palm-side of the hand. Their system captures an infrared image of the palm, from which vein patterns are extracted. Performance characteristics are not stated, nor is it believed that any palm vein systems are currently in use.

Details of the scanning mechanism are unknown, however, a patent by one of the company directors outlines the broad functionality of such a system (Stiver & Peterson 1998). In this a linear array of fibre optic cables is housed in a transparent cylinder, upon which a hand grips. As the cylinder rotates an infrared image of the palm vein pattern is captured.

2.3.4 Palmprints

The palm offers a rich set of time-invariant features which can be used for personal identification (Shu & Zhang 1998). These features include the geometrical characteristics of length, width and area; the principal lines, numbered 1-3 in Figure 2.14(a); and minutiae features, common to all ridged skin. Figure 2.14(b) shows an optical scan of the author's palm, captured with 8-bit greyscale resolution at 150dpi using an off-the-shelf flatbed scanner (ScanJet5200C from Hewlett Packard). Due to low contrast it is difficult to see much of the detail in this image. Local histogram equalisation was applied over consecutive 8×8 pixel regions to improve feature contrast, giving the result of Figure 2.14(c), which clearly shows the three principal lines as suggested by 2.14(a). Detail of large well defined ridges can be seen in the lower left-side of the image, although the scanning resolution is too low to see the fine ridges of much of the palm's remainder.

Identity verification using palm characteristics offers some advantage over the use of fingerprints in that the geometrical and principal line features can be extracted from relatively low resolution images, and as a result exhibit greater insensitivity

[11] Advanced Biometrics Inc. Puyallup, Washington, USA. Web: `http://www.adv-bio.net`

(a) Principal Palm Lines, After Shu (1998)

(b) Optically Scanned Palm (8-bit greyscale, 150dpi)

(c) After Histogram Equalisation

Figure 2.14: Palm Prints

to noise. Furthermore it is claimed that their capture is easier amongst groups whose fingerprints are faint, such as the elderly or manual workers.

Shu & Zhang (1998), describe an approach to palmprint identification using a combination of palm geometry and two of the principal lines, indicated above. Their approach firstly determines the origin and orientation of the palm, then the position of a number of equidistant points on two principal lines are extracted and used as features for comparison. Although Shu and Zhang do not give explicitly the number of participants in their experiments (stating only that "...48 pairs of prints from the same hand and 844 pairs of prints from different hands..." were involved), they report an equal error rate of zero. No details about their method of capturing palmprints is given, other than the final images characteristics of 400×400 8-bit greyscale pixels, sampled at 100dpi. Further details of their line extraction algorithm, necessary for automated verification, is given in (Zhang & Shu 1999).

Duta, Jain & Mardia (2002), describe a slightly different approach to palmprint verification, in which isolated points lying along all of the palm lines are used for comparison. This approach is claimed to be faster and more efficient than the line extraction method of (Shu & Zhang 1998, Zhang & Shu 1999). An overlap of 5% in genuine and impostor feature distributions is reported.

Duta et al. (2002), captured their palmprints using a flexible rubber pad, upon which an inked palm is placed. This leaves an inked print of the palm which is

then transferred to paper and scanned using an off-the-shelf flatbed scanner (8-bit greyscale at 200dpi). 30 images in total were collected from both hands of three participants.

To the author's knowledge there are no companies offering palmprint identification systems for civilian applications. However, palmprints are generating some interest in the field of criminal investigation, and a number of companies offer inkless palm scanning and recognition devices.[12]

2.3.5 Finger Characteristics

Both finger geometry and other finger characteristics such as finger crease patterns and finger vein patterns have been proposed for use in automated identification.

Finger geometry is a variation of hand geometry, in which geometrical characteristics, such as finger length, width and height, or the position of the finger joints, are used as the basis for discrimination (Jain et al. 1998). One company, BioMet Partners Inc.[13], has developed such a system. As can be seen in Figure 2.15, its form is similar to that of the hand geometry system presented in Figure 2.12, if more compact.

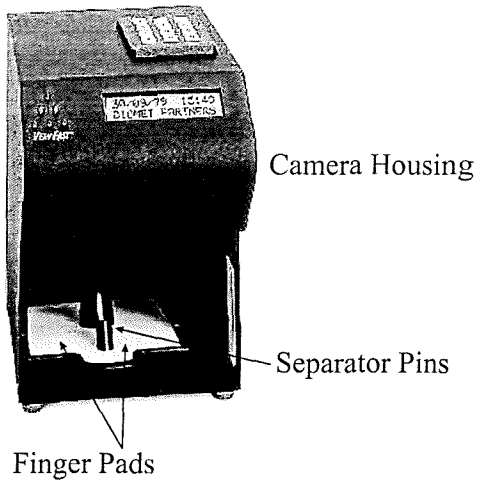


Figure 2.15: Finger Geometry System (BioMet Partners, Inc.)

[12]See for example Printrak's LiveScan3000 device (`http://www.printrakinternational.com`) or the PrintQuest device from SPEX Forensics Group (`http://www.crimescope.com`)

[13]BioMet Partners, Inc. Pestalozzistrasse 12 CH-3280 Morat, Switzerland. Web: `http://www.biomet.ch`

BioMet claim that their system has been extensively tested with over 5 million enrollees using a number of installations at DisneyLand (Florida, USA). Although offering no testing nor evaluation details, an equal error rate of 0.1% is claimed.

Joshi, Rao, Kar, Kumar & Kumar (1998), describes a method of capturing and using the pattern of creases, found on the inner surface of the finger, as the discriminatory features for verification. Their apparatus is again similar to that of figures 2.15 and 2.12, with an imaging device positioned over the hand. In this approach, however, the hand is placed palm-side up, offering the finger crease pattern to the imaging device, as shown in Figure 2.16. As the hand is moved into position a micro-switch closes which triggers the imaging device. This approach is believed to reduce positional variance between images.

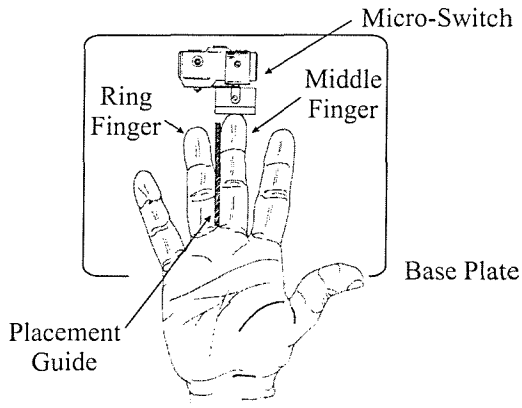


Figure 2.16: Finger Crease Schematic, After Joshi et al. (1998)

In a verification experiment involving 206 participants, Joshi reports an equal error rate of ~0.3% when using crease patterns from middle fingers only. This improved to around 0.02% when both middle and ring fingers were considered. Finger crease patterns were represented as intensity averages, taken across the width of the finger at 472 points along the finger length, giving 512×512 pixel images with an 8-bit greyscale resolution.

Hitachi[14] have developed a verification system based upon the vein pattern of a person's index finger (Lockie 2001a). In common with the approach to hand-vein recognition of NeuSciences, infrared light is used to illuminate the finger

[14]Hitachi Ltd. Tokyo, Japan. Web: `http://www.hitachi.com`

and improve vein-flesh contrast. Hitachi claim a 100% identification rate, from a population of 700 participants, although no further details are known.

Willmore (1994) was granted a patent for the broad concept of embedding an infrared CCD (IR-CCD) onto any type of transaction media (such as credit cards, computer keyboards, security access media). The function of the IR-CCD is to capture IR radiation from vein-dense body parts – Willmore suggests the possibility of using fingers, nose or ear-lobes as being suitable for the task. As such, this approach differs from that of the other IR vein pattern verifiers in that it relies upon thermal contact of the body part rather than an optical imaging device to collect the radiation. This concept may lead to significantly smaller devices.

Willmore proposes that the necessary verification tasks of thermal imaging, image storage, and image comparison, are performed on the same ASIC module. Specific application details are lacking, and it appears that this work is based upon concept rather than implementation.

2.3.6 Face Recognition

Face recognition is perhaps the most common means by which humans recognise each other. Automating the process of recognition, however, is a difficult task requiring that the system account for variations in expression, facial hair, wearable items such as glasses, hats and scarves.

The automated recognition of human faces can be performed either *overtly* or *covertly*. Overt operation requires the implicit knowledge and cooperation of an individual, and is typically performed within a controlled environment with constant illumination and uniform background. Resulting face images are normalised in size and position prior to, and to facilitate, feature extraction and recognition. Such images are termed *canonical* faces.

Covert operation, on the other hand, may involve imaging an individual's face without their knowledge, or indeed consent. This may take place in airports, shopping or town centres (Ashbourn 2000). Covert operation (in particular) must therefore be sufficiently robust to deal with variations in lighting conditions, background, and the imaging angle between face and camera.

Irrespective of operational mode, face recognition techniques must be insensitive to the normal "non-obstructive" variations of clothing, glasses and facial hair, as indicated above. Intuitively, derived features such as the position and angle between the eyes, mouth, nostrils and chin, could be used as the basis for comparison. Unfortunately the automatic detection of such features is unreliable due to the variations as described above. As a result these manually derived features are difficult to measure accurately. An alternative approach is to use automatically derived features for comparison. To this end, the use of self-organizing neural networks has been demonstrated (Kohonen 1988) and other possibilities include *appearance*-based techniques, such as those described in (Turk & Pentland 1991, Etemad & Chellapa 1994).

A number of companies develop and sell face recognition systems, including: Visionics Corporation[15], ZN Vision Technologies[16], Dermalog[17] and FaceKey Corporation[18]. Visionics' recognition system operates with a claimed equal error rate of less than 1% in verification mode (the NPL study suggests an equal error rate of ~2.5%). Applications of face recognition technology include the covert surveillance of air travellers at a number of undisclosed US airports and Iceland's Keflavik airport, covert surveillance of Birmingham city centre (UK) and the town centre of Newham borough (London, UK); and at a number of football grounds in the UK. ZN Vision Technologies and FaceKey both sell face recognition systems for overt access control and covert personnel tracking in buildings.

The applications described above require the use of visible-band imaging devices. These will typically be modern CCD video cameras, although the recognition system of Visionics is able to use digitised images for a number of sources, including digital still cameras and even scanned 'photo-fit' images.

There have been some developments in the use of infrared imaging of faces (*face thermograms*) (Prokoski & Riedel 1998). In common with the hand vein and infrared finger approaches to identification, infrared face scans make use of vascular patterns for discrimination. This offers some advantages, in that facial hair, and other feature-obscuring agents are less pronounced in the IR band. Further-

[15]Visionics Corporation, Jersey City, USA. Web: `http://www.visionics.com`
[16]ZN Vision Technologies, Bochom, Germany. Web: `http://www.zn-gmbh.com`
[17]Dermalog GmbH, Germany. Web: `http://www.dermalog.com`
[18]FaceKey Corporation, Albany, San Antonio, USA. Web: `http://www.facekey.com`

more, plastic surgery which does not re-route the vasculature is believed to have no circumventive properties. In addition, IR imagers may be used robustly under variable lighting conditions or in the absence of light altogether. However, it is likely that face thermograms will be affected by emotional state and body temperature (Jain, Hong & Pankanti 2000).

The level of vascular detail available to a recognition system depends upon the thermal sensitivity of the imaging device. Low-sensitivity imaging systems[19] are restricted to using contour matching of captured images, whereas sensitive devices[20] are able to capture the *minutae-like* detail of the blood vessels. For more detail see Prokoski & Riedel (1998).

2.3.7 Iris Features

The iris controls the amount of light entering the eye by constricting and dilating in response to ambient light. It is the distinguishing features contained in the tangled mesh of iris musculature and connective tissue upon which iris recognition is based (Daugman 1998).

The iris is an internal component of the eye protected by a transparent sheath called the *cornea*. As such it is accessible to optical image capture, and yet remains inaccessible to modification unless through invasive surgery. Liveness tests are able to exploit the iris' response to light intensity, the reflective properties of the eye, or the small, steady-state oscillatory dilation of the iris (this is termed *hippus* and has a frequency of around 0.5Hz).

The complex features of the iris are widely considered, both theoretically and empirically, to contain an extremely high number of independent degrees of freedom leading to a recognition method of high accuracy (Daugman 1993, Miller 1994, Wildes 1997). Indeed a number of manufacturers[21] of iris recognition systems claim equal error rates in the order of 10^{-6}. The NPL study found 2% false

[19]Devices having a sensitivity of around 0.7 to 0.1°C Noise Equivalent Temperature Difference (NETD).

[20]With sensitivity less than 0.07°C NETD.

[21]See for example: Iridian (`http://www.iridian.com`), who are the licence holders of US patent: US5291560, upon which all current commercial iris recognition systems are based (Daugman 1998)

acceptance with no false matches, involving over 2 million cross matches.

The practical implementation of an iris recognition system will typically involve the use of a monochrome CCD camera to capture an individual's face. Preprocessing will then find the location and outline of the eye and from this extract an image of the iris. The iris image is then split into annular regions which are characterised by texture, according to the algorithms of Daugman (1994).

2.3.8 Ear Characteristics

Biometric ear characteristics relies upon the distinctive folds of cartilage constituting the visible outer portion of the ear (Burge & Burger 1998). Evidence that ear characteristics are likely to be unique is given by Iannarelli (1989), who made an objective study of over 10000 ears. Indeed, Iannarelli proposed a system of 12 anthropometric measurements, upon which his study was based. Identity verification based upon ear characteristics is an attractive proposition because of their accessibility to image capture and are less easily disguised in the manner of face characteristics.

Although ear recognition complies with an image processing approach, latent ear prints are often found at the scenes of crime. This is particularly common in regions with a high prevalence of apartment blocks, and can be explained by would-be intruders listening with their ears pressed against doors (Lockie 2000).

A number of computer-vision approaches to automating identification or identity verification using ears have been reported (Hurley, Nixon & Carter 1999, Burge & Burger 2000), although this work is in the research stage and the author is unaware of any commercial systems making use of ear recognition.

2.3.9 Retina Identification

The uniqueness and invariance of the vein pattern upon a specific region of a person's retina is the basis for retina identification (Hill 1998). This method is widely claimed to be highly secure due to the difficulty involved in changing, or replicating a person's retinal vasculature. However, the intrusive nature of retinal

identification reduces user acceptance as many users fear retinal damage, or other health risks associated with close contact of the imaging system.

In operation, a typical system[22] requires a person to look into an eyepiece then move and tilt their head, until a number of alignment marks coincide. At this stage the user must initiate a near-infrared scan of (a small disk of) their retina. IR light entering the eye actually passes through the retina (which is transparent to near-IR), before reflecting from the vasculature of a structure behind the retina, called the *choroidal* vasculature (Hill 1998). Thus, as Hill comments, the term 'Retinal Identification' is something of a misnomer[23]. Representative features are then extracted from the reflected light.

Acquisition of high quality retinal images requires involved cooperation from the user. For this reason a real operational system is likely to experience a significant number of false rejections. However, retinal identification systems have found applications in a number of high-security government, military, nuclear and financial settings, in which false rejections are of less concern than false acceptances. False acceptance rates of 10^{-6} are claimed by EyeDentify, albeit at an unknown false rejection rate.

2.3.10 Speaker Recognition

The recognition of a person through their voice characteristics can be an efficient and natural method biometric, especially so when voice capture devices are already present (such as telephones or PC microphones) (Campbell 1997, Campbell 1998, Ashbourn 2000).

Speaker recognition[24] may be text dependant, requiring an individual to repeat a predetermined word or phrase during both enrolment and verification, or may be text independent, recognising an individual by the general characteristics of their speech. As a behavioural biometric, speaker recognition must be sufficiently robust to cope with natural variations induced by psychological state, tiredness, and illness. Other sources of variation occur from the use of different microphones

[22] Such as the system from EyeDentify (`http://www.eye-dentify.com`)

[23] Early 'retinal identification' systems used visible light, but the intensity levels required for reasonable signal to noise ratios caused discomfort to users, hence the move towards infrared.

[24] Rather than *voice recognition* which generally refers to decipherment of speech.

for enrolment and verification sessions (*channel mismatch*) from inconsistent room acoustics, or from varying background noise (Furui 1997).

Speaker recognition may be defeated by simple recording of an enrolled person's voice (or specifically the pass phrase in the case of text-dependant recognition). In an attempt to defeat circumvention in this manner, systems have been proposed which request the user to utter words in a randomly prompted manner (Matsui & Furui 1993).

A number of authors have reported different approaches to speaker recognition. Although specific details are beyond the scope of this thesis, some of the investigations include Doddington (1985) who reports text-dependant verification exhibiting an equal error rate of 0.8% with 6 seconds of speech (200 participants); Tishby (1991) reporting an equal error rate of 2.8% for verification using 1.5 seconds of prompted, isolated digits (100 participants); and Higgins & Wohlford (1986) with an equal error rate of 10% for verification with 2.5 seconds of text independent speech (11 participants). These results are taken to represent verification using only short utterances, from a wider survey by Campbell (1998).

A number of commercial voice verification systems exist, including those of VeriVoice[25] offering both stand alone access control and telephone integration systems with claimed equal error rates of 1.7%, Veritel[26] offering access control (with undisclosed error rates) and Keyware Technologies[27] offering both access control and telephonic systems. The NPL study reports an equal error rate of ~1.3% for speaker verification.

2.3.11 Handwritten Signature Verification

The handwritten signature has long been accepted as a legal demonstration of identity (Ashbourn 2000), however, as a static entity there may be little to distinguish between a genuine signature and a good forgery. For this reason, there is considerable interest in capturing the additional invariant *dynamics* of signature creation for the purposes of identity verification (Plamondon & Lorette 1989, Nalwa 1997).

[25] VeriVoice, Inc. Princeton, USA. Web: `http://www.verivoice.com`
[26] Veritel Corporation, Chicago, USA. Web: `http://www.veritel.com`
[27] Keyware Technologies, Woburn, USA. Web: `http://www.keyware.com`

Signature verification may be either *off-line* wherein the static resulting signature characteristics are compared, or verification may be *on-line* in which dynamic characteristics such as pen pressure, pen tilt, and time-position are the basis for comparison. Dynamic verification requires the use of some peripheral device (such as a graphics tablet or instrumented pen-like device) to capture pen stroke information, whereas static off-line verification requires only that final signature images be digitised in some manner.

There is a significant body of published work involving both on- and off-line signature verification, and some recent examples include: (Bajaj & Chaudhury 1997, Huang & Yan 1997, Baltzakis & Papamarkos 2001) reporting off-line approaches and (Yang, Widjaja & Prasad 1995, Wu, Lee & Jou 1998) describing on-line schemes. Plamondon & Srihari (2000) provide a comprehensive survey of handwriting recognition, of which signature verification is a subset.

Bajaj & Chaudhury (1997) reports a false rejection rate of 1% at a false acceptance rate of 3% for an investigation involving 10 persons, each of whom supplied 5 training and 5 testing signatures. Impostor samples were supplied by 100 random forgeries, although no further details are given. (Huang & Yan 1997), on the other hand, made use of 144 detailed forgeries per genuine signature, created either by tracing or freehand copying of the genuine signatures. From an experiment involving 21 participants, each supplying 24 genuine signatures, an equal error rate of 11% is reported. Baltzakis & Papamarkos (2001) collected between 15-25 signatures each from 115 participants (10 of whom provided signatures on 3 to 5 separate occasions), and reports a false rejection rate of 3% with a false acceptance of 9.8%.

Yang et al. (1995) reports a false acceptance rate of 5.2% at a false reject rate of 2.5% in an experiment involving 31 participants, each of whom provided 8 training and 8 testing samples. False acceptances were determined with the signatures of all other participants being used as impostor signatures, rather than explicit forging attempts. Whilst (Wu et al. 1998) used 30 samples each from 27 participants, 10 of which were used for enrolment, the others for testing. Four 'skilled experts' provided 20 forged signatures associated with each of the genuine participants. A false rejection rate of 1.4% at 2.8% false rejection is reported.

A number of commercial products embodying handwritten signature verification

exist, including: On-line verification software from PenOp[28] and CyberSign[29], although neither company quote performance rates; and the instrumented Smartpen from LCI Technology group[30]. Performance characteristics are also not quoted for this device.

In common with other behavioural biometrics, signature verification is subject to changes in psychological state, and stress, pressure, tiredness and alcohol, for example, are all thought to contribute to short term variations in handwritten signatures (Ashbourn 2000).

2.3.12 Keystroke Dynamics

Keystroke dynamics offers a route to identity verification through the characteristics of a person's typing style (Umphress & Williams 1985, Joyce & Gupta 1990, Bleha, Slivinsky & Hussien 1990, Obiadat 1998). Two forms of keystroke dynamics are considered, and these are namely *static* or *dynamic* verification. Static verification is the one-time recognition of a user, generally at the start or login of a computer session and typically makes use of key-press and inter-key times. Dynamic verification, on the other hand, aims to continuously test identity throughout the lifetime of a session, and makes use of the timing characteristics of specific key press sequences (called *di-* or *tri-graphs*).

It has been suggested that the same neurophysiological characteristics responsible for the individuality of handwriting result in the individuality of a person's typing characteristics (Joyce & Gupta 1990). Indeed, static verification may be considered the keystroke equivalent of handwritten signature verification, in that both consider the invariance of short fixed strings. Equally, dynamic verification may be considered analogous to text-independent handwriting recognition using an arbitrary sample of prose.

Joyce & Gupta (1990) reports an equal error rate of around 3% for static verifcation with each participant typing their first and last names and computer usernames

[28]PenOp, Communication Intelligence Corporation, Redwood Shores, CA, USA. Web: `http://www.penop.com`

[29]CyberSign Inc. San Francisco, CA, USA. Web: `http://www.cybersign.com`

[30]LCI Smartpen Inc. International Drive, Portsmouth, NH, USA. Web: `http://www.smartpen.com`

and passwords, as login strings. Bleha et al. (1990) and Brown & Rogers (1993) make use of only the participant's names as login strings. Bleha reports a false acceptance rate of 2.8% with a false rejection rate of 8.1%, whilst the method of Brown resulted in a false rejection rate of 8% at a false acceptance rate of 9%. Meanwhile, Robinson et al. (1998) used participant's usernames only, and reports a false rejection rate of 10% with a false acceptance of 9%.

Keystroke dynamics is an attractive proposition for computer access control simply because it makes use of existing hardware. Indeed, commercially available keystroke verification software is offered and licensed from BioPassword[31], and is used in a number of applications including: authentication of credit card users across the internet; authenticating access to on-line learning resources; and in the control of legitimate music downloads from the internet. Performance characteristics are not given.

In common with signature biometric, this method is behaviorally based, and may be influenced by the psychological state of the subject.

2.3.13 Gait

A person's gait is a description of the temporal-spatial way in which they walk (Nixon, Carter, Cunado, Huang & Stevenage 1998). This is thought to be distinctive. Although much research has been done on the effect of biomechanical defects upon gait, only recently has this work been extended to identification of a person.

Analysis of live video can extract gait characteristics attributed to a person.

2.3.14 Other Biometrics

There are a number of other human aspects which have been proposed for use as identification agents. DNA[32] profiling is the obvious identification method, being unique to each person (with the exception of identical twins) (Rudin 1998). However, for a number of reasons it's use as an identifier is mainly limited to

[31] BioPassword from Net Nanny Software Inc. Bellevue, WA, USA. Web: `http://www.biopassword.com`

[32] Deoxyriboneucleic acid

forensic applications. The primary reasons are as follows: DNA samples are easily collected from the unsuspecting, for example in exfoliated skin or hair; current DNA analysis technology is not adapted for fast, real time processing and DNA variations within the human genome occur only in a very small proportion of its structure. Added to these practical concerns, the storing of DNA data is widely viewed as a potential invasion of privacy.

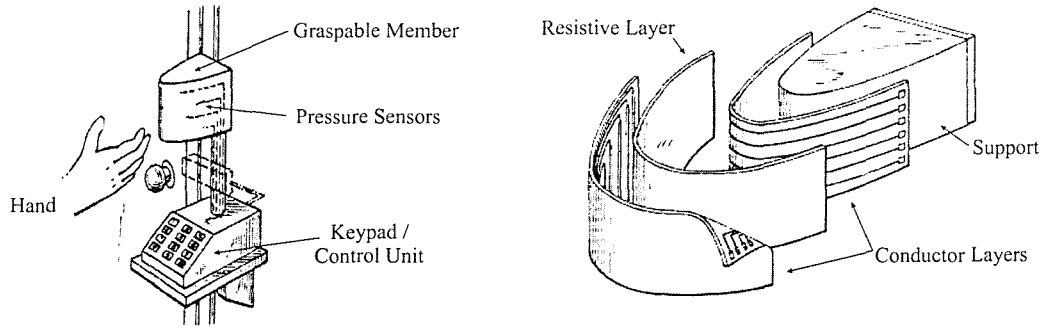
Differentiation of humans by their characteristic odour has been proposed and at least one attempt at a prototype verification system has been made (Persaud, Lee & Byun 1998). However, the use of deodorants or perfumes, diet and medication may change the *odour profile* of a person, making recognition difficult. This is still very much a research area.

Identification by means of nailbed pattern has been proposed (Topping, Kuperschmidt & Gormley 1998). The human nailbed comprises of a longitudinal series of parallel epidermal ridges, whose pattern is claimed to be unique and invariant. Using reflected infrared laser sources this pattern can be recorded, and used as the basis for identification. Nailbed verification is believed to be under development by AIMS Technology Inc.[33]

Another interesting patent initiates the concept of identity verification based upon the manner in which an individual grasps an object (Bellin 1989). Bellin proposes a *graspable member* with a *plurality of pressure sensors*, and includes some means of recording the pressure pattern resulting from a hand grasp. Figure 2.17(a) gives an overview of the system, controlling a door lock, whilst Figure 2.17(b) shows the construction of the pressure sensitive components. The surface of the object to be grasped is comprised of two conductive grids, separated by a compressible resistive layer - the resistance of which changes with applied pressure.

Bellin (1989) offers no evidence that this approach has been validated, nor gives any details of appropriate verification techniques. Nonetheless, this is an interesting idea in that the inclusion of temporal characteristics, during the grasp and release of the object, potentially adds an additional layer of discrimination.

[33]AIMS Technology Inc. Little River, SC, USA. Web: `http://www.nail-id.com`



(a) System Overview

(b) Pressure Sensor Schematic

Figure 2.17: Identification Based upon Grasping Pressures (After Bellin (1989))

2.4 Approaches to On-Card Verification

Clearly the small size and shape of a smartcard places severe restrictions upon an embedded verification system's ability to capture discriminatory human characteristics. These restrictions are imposed upon both the quantity being measured, and the mechanism with which to do so. For example quantities such as hand geometry and palmprints exceed the planar dimensions of a smartcard, whilst face, iris, retina and ear recognition require the use of optical imaging devices, currently too large to be integrated on-card. Whilst those involving optical or infra red illumination – such as hand vein pattern or nail-bed recognition - are equally excluded on the basis of space constraints.

The physical interactions between smartcard and fingers are an obvious basis for discrimination, but current implementations of these biometrics are not appropriate for smartcard integration, Fingerprints, for example, are either captured optically if with silicon-based capacitance, thermal or pressure sensors, and similarly, finger-geometry and finger-crease patterns are captured optically or electromechanically. Clearly optical or electromechanical sensing mechanisms are impractical, and silicon in its native brittle form does not comply with the mechanical flexibility of a smartcard. Nevertheless as indicated in Section 1.4, Infineon are reported to be developing flexible silicon-based fingerprint sensors, although this approach is likely to be expensive and will require further protection against chemical and electrostatic damage.

The approach taken in this thesis is to use compliant polymer thick film sensors to capture discriminatory characteristics. Although, it will become apparent (in Chapter 3) that such sensors do not exhibit the required spatial resolution of fingerprint sensors, the principles of pressure and capacitance sensing, involved in the capture of fingerprints, may be exploited to capture other biometric quantities. In Chapter 3, polymer thick film pressure sensors are described, and a new implementation of a capacitance sensor is proposed.

Finger-geometry systems capture both the planar and profile characteristics of the hand and finger, and use features such as finger length and width, knuckle positions, distance between knuckles, and finger height at a number of places. A smartcard sensing system is restricted to sensing within its plane, and this implies that only planar geometry can be captured. However, if the system is considered dynamically, and can capture images rapidly, then by 'rolling' a finger on the card's surface, both planar geometry and profile information can be captured. Alternatively, planar information only from two fingers could be captured, and features may include both finger-geometry and finger-crease patterns.

If a section of palmprint can be consistently placed on the card, then it may be possible to capture and use the principal lines for discrimination. This would require significantly lower spatial resolution than fingerprint sensors.

The concept of Bellin (1989), involving the capture of not only spatial, but additionally temporal information is interesting. If spatial resolution is limited by the sensor mechanism, then the inclusion of temporal characteristics may offer additional discriminatory features. Voice characteristics could be used to verify identity, provided a shallow, robust and flexible microphone can be developed. Under such circumstances, an on-card speaker recognition system may become a viable proposition. Further consideration of these approaches is deferred until the further work section of Chapter 7.

Further, as indicated in section 2.2.3, the complexity of features, matching algorithm and computational performance of the processor contribute to the time required for enrolment and verification. It is important, then, that the processor of a *typical* smartcard is capable of executing enrolment and verifications functions within a *reasonable* time duration. Such issues are considered in Chapter 6.

2.5 Concluding Remarks

In this chapter the emerging field of biometrics is investigated for mechanisms of strengthening the association between a smartcard and its legitimate holder. Current and proposed biometric techniques are considered in some depth and fundamental principles have been stated. Because of the commercial interest in this field, much of the work remains proprietary and, as a result, a significant proportion of this review had to rely upon marketing literature. Academic references have been provided where available.

In addition, two independent studies have assessed the performance of currently available commercial devices, and these results are cited where appropriate. For example, the study by Mansfield et al. (2001) investigated the unhabituated performance of: fingerprint verification, reporting an equal error rate of ~3%; hand geometry verification (EER of ~1.5%); and voice pattern recognition (EER of ~1%). The performance of iris recognition is the one which stands out from all others, with Mansfield et al. (2001) reporting a false rejection rate of around 2% and no false acceptances in over two million cross-matches.

The dimensions and manner of use of a smartcard places restrictions upon the scope of human-card interactions which are available for discrimination. The discriminatory characteristics of fingerprints, finger-geometry (and related finger-crease pattern), palmprint section (if consistent placement can be assured) and voice pattern have been identified as plausible.

Pressure, capacitive and thermal sensing mechanisms are employed in the capture of currently available biometrics systems, and may be useful in capturing human-smartcard interactions. The smartcard additionally places the restrictions of flexibility, robustness and (ideally) low-cost, upon an embedded sensor device. A compliant technology is that of polymer thick films, and will hence be assessed in detail for its potential to capture human-card interactions in Chapter 4. The results from Chapter 4 will be exploited in Chapter 5, whereupon a novel approach to identity verification will be proposed and demonstrated.

As previously indicated (in section 2.2.3), one principal advantage of embedding an identity verification system upon a smartcard is that the acceptance threshold and matching algorithm are held securely within the trusted smartcard processor.

Section 2.2.3 considered that the computational demands placed upon the matching processor are related to the complexity of both the discriminatory features and the matching algorithm. It is hence important that any approach to identity verification is demonstrated to execute within an *acceptable* time on a typical smartcard processor. This is considered in detail in Chapter 6.

The next chapter provides an outline of polymer thick film sensors, and assesses their technological properties.

Chapter 3

Polymer Thick Film Sensors

3.1 Introduction

Polymer thick film (PTF) technology is an established circuit fabrication method used in the production of flexible components, including conductors, resistors and dielectrics (Fu et al. 1981). Early development of the technology occurred during the mid 1970s and the approach is still used today in components for calculators, keyboards, mobile telephones and a multitude of other devices where low-cost, flexible circuits are required (Gilleo 1995). Furthermore, by exploiting the intrinsic material properties of thick films, sensing elements can be produced which are widely considered to be compact, robust and relatively inexpensive (White & Turner 1997, Harsányi 1995).

Thick film technology represents the precise, selective deposition of materials onto a substrate. Preprocessed thick film materials are called *pastes* or *inks*, and these terms are used interchangeably. PTF pastes generally consist of filler particles bound within a polymer matrix (usually epoxy, silicone, or phenolic resin), and it is the properties of these filler particles which determine the use of the paste. Gold and silver flakes or copper particles may be used for conductive pastes, whilst carbon particles may be used for resistive purposes. Solvents are added to improve printing characteristics and minerals are occasionally added to achieve desired electrical and mechanical properties (Papakostas & White 2000b).

Thick film inks are most commonly printed onto a substrate using a screen printing

process, whereby ink is forced through a patterned mask and onto the substrate. Hence, deposition is selective according to the pattern on the mask, and deposited layers are usually $\ll 100\mu$m. Generally, PTF materials are processed at temperatures of less than 200°C, allowing a wide range of thermally compatible substrates to be used. These include the mechanically flexible materials of polyester, polycarbonate, polyimide, polyvinyl acetate (PVAc) and acrylic (Papakostas & White 2000b, Hicks, Allington & Johnson 1980, Gilleo 1995). It is these properties of flexibility combined with robustness and low-cost which make PTF sensors attractive for integration with smartcards.

This chapter investigates the properties of polymer thick film sensors bonded to smartcards, with the ultimate aim of using these sensors to capture discriminating human characteristics. It begins by presenting an overview of thick film processing techniques, and then provides a review of PTF applications. The development of piezoelectric and piezoresistive PTF pressure sensors has been reported in the literature (Harsányi 1991, Arshak, Ray, Hogarth, Collins & Ansari 1995, Papakostas & White 2000b, Papakostas & White 2000c), and arrays of such sensors are considered in Section 3.4.1. The fabrication of piezoelectric and piezoresistive sensors onto flexible polyimide substrates is described in Section 3.5, and their principles of operation are derived in Sections 3.5.2 and 3.5.3.

3.2 Thick Film Fabrication

The most common method of thick film material deposition is the flatbed screen printing method (Loasby & Holmes 1976). This operates on the principle that material pastes are selectively forced through the fine mesh of a patterned screen mask. Thick film screens usually comprise a finely woven mesh (typically stainless steel or nylon), mounted under tension on a metal frame (Loasby & Holmes 1976). The mesh is bilaterally covered with a photo-sensitive emulsion, upon which the circuit pattern is photographically formed. This results in open regions from which the emulsion has been removed – a cutaway section of screen is shown in Figure 3.1.

Thick film paste is deposited on the surface of the screen suspended about 0.5mm above the substrate. (Figure 3.2(a)) A *squeegee* traverses the screen, under pres-

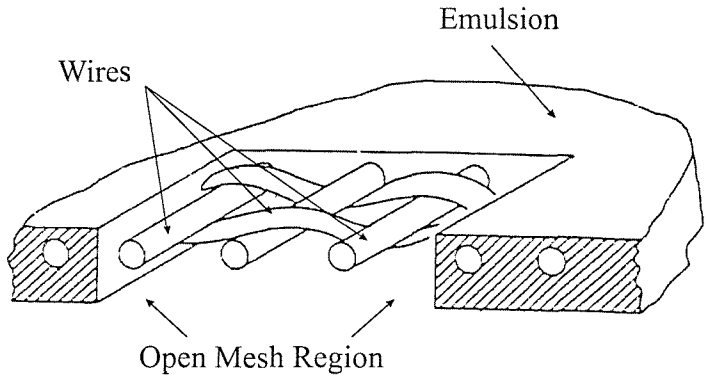


Figure 3.1: Thick Film Screen Mask (After Loasby and Holmes (1976))

sure, bringing the screen into contact with the substrate and driving the paste through the open mesh regions (Figure 3.2(b)).

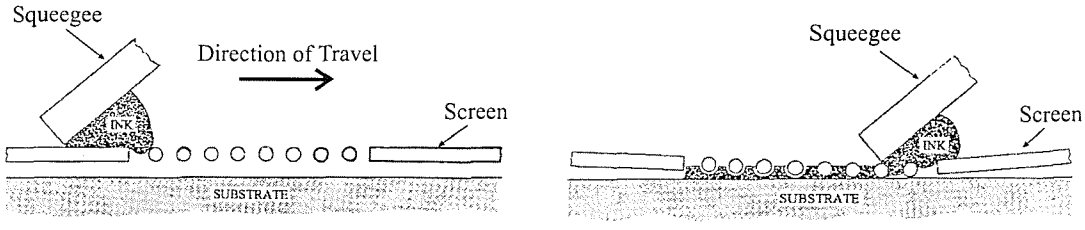


Figure 3.2: Thick Film Printing Process (After Savage (1976))

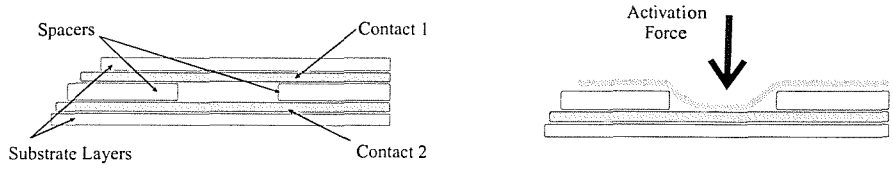
Printed layers are dried to remove solvents, at temperatures of around 100°C, then *cured* at temperatures generally less than 200°C. The curing process encourages *cross-linkage* between polymer monomers, resulting in polymer shrinkage and improvement in film stability (Papakostas 2001). The result is a solid composite film which is firmly bonded to the substrate. Further layers can be added as required.

Conventional screen printing processes are generally capable of producing minimum line widths of around 100-150μm (resulting in a pitch of 200-300μm) (Leppävuori, Väänänen, Lahti, Remes & Uusimäki 1994), although Robertson, Shipton & Grey (1999) presents a stencilling approach to screen fabrication capable of producing line widths of 50μm (100μm pitch). An alternative deposition method is that of *gravure offset printing* (Leppävuori et al. 1994) in which a plate is etched

with the desired circuit pattern. Recesses are flooded with ink and intermediately transferred, to an elastomeric pad by means of contact. Ink is then deposited onto the substrate in the same manner. Line widths of 50μm have been reported using this method.

3.3 Applications of Polymer Thick Films

The mechanical flexibility of PTF conductors is the enabling technology for membrane switches (Hicks et al. 1980, Gilleo 1995). These low-cost, reliable, compact devices consist of two polymer conductors printed on flexible dielectrics, separated by a thin spacer layer. The construction and operation of a simple membrane switch is shown in Figures 3.3(a) and 3.3(b), respectively. When a sufficiently large activation force is applied, the top electrode moves through the gap in the spacer layer and closes the switch.

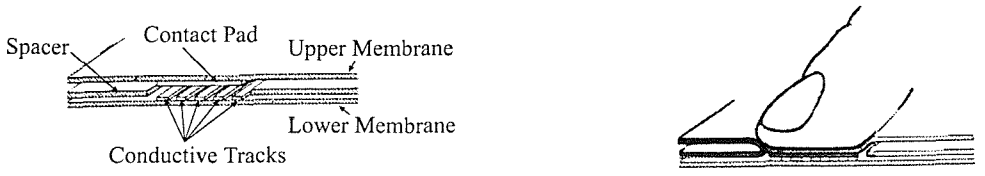


(a) Switch Construction – Open (b) Switch – Closed

Figure 3.3: Simple PTF Membrane Switch (After Gilleo (1995))

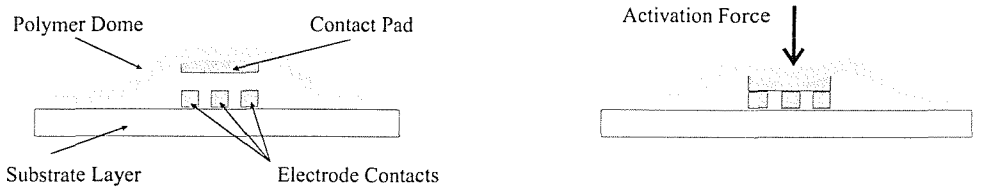
Variations on this simple membrane switch design are available, e.g. the upper membrane may contain an isolated conductive pad which moves beyond the spacer layer in response to applied pressure, completing conduction between interdigitated electrodes on the bottom layer. Figure 3.4(a) shows this schematically, whilst 3.4(b) demonstrates the operation.

Alternatively, the upper membrane may be replaced by a conducting dome structure, providing increased travel and tactile feedback to users. The dome may be fabricated by thermoforming a conductor printed on polymer layer or by inserting conductive material into molded elastomer. The resulting structure is shown in Figure 3.5.



(a) Switch Construction – Open (b) Switch – Closed

Figure 3.4: PTF Membrane Switch (After Hicks et al. (1980))



(a) Switch Construction – Open (b) Switch – Closed

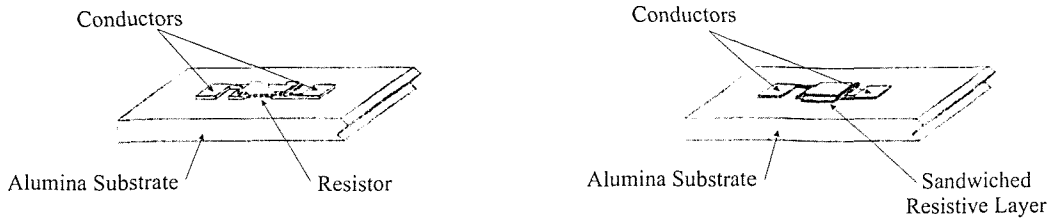
Figure 3.5: PTF Dome Switch (After Gilleo (1995))

The intrinsic properties of polymer thick film materials can be exploited for sensor purposes, and PTF resistors have been shown to exhibit resistive changes in response to applied strain (Hollingum 1984, Harsányi 1991, Arshak et al. 1995). As an illustration of this point, Arshak et al. (1995) describes two piezoresistive strain gauge structures, the first of which is a single resistive layer, whilst the second is a Metal-Resistor-Metal structure, as shown in Figures 3.6(a) and 3.6(b), respectively. Both structures exhibit resistance changes in response to applied strain.

Sensitivity to strain is expressed in terms of the *gauge factor*, which for an applied strain in the longitudinal (x) direction of the sensor is given by the longitudinal gauge factor, GF_L, defined as

$$GF_L = \frac{dR/R}{\varepsilon_x} \tag{3.1}$$

where R is the resistance of the sensor and dR is the change in resistance of the



(a) Planar Resistive Layer (b) Sandwiched MRM Structure

Figure 3.6: Piezoresistive Strain Gauge Structures (After Arshak et al. (1995))

sensor resulting from the application of a strain ε_x in the x direction. Note that GF_L is the proportional change in resistance due to a unit strain applied in the longitudinal direction. Whilst Arshak et al. (1995) reports a high gauge factor of around 80 for the MRM structure, in comparison to between 4 and 5 for the single resistive layer, he also notes that the MRM structure exhibits almost 5 times higher non-linearity and 4 times greater hysteresis in comparison to the simple resistive structure.

Arshak, M^{c}Donagh & Durcan (2000) make use of a capacitive structure similar to the MRM sensor described in Figure 3.6(b), differing in the use of a polymer dielectric, rather than resistive sandwiched layer. Its sensitivity is explained by the change in dielectric permittivity (and hence sensor capacitance) in response to applied strain. A gauge factor of around 3.5 is reported for this device printed on alumina.

Piezoelectric PTF pastes have been developed, printed on alumina substrates, and have been shown to exhibit good force sensitivity of around 10pC/N (Papakostas et al. 1998). Piezoelectric polymers have also been developed in aerosol form, allowing direct deposition onto structures (Hale & Tuck 1999).

Screen printable electroluminescent polymers and organic polymer LEDs have been reported (Zovco & Nerz 1999, Leung, Kwong, Kwok & So 2000), and resistive inks have been shown to be sensitive to temperature, humidity and certain gases (Lundberg & Sundqvist 1986, Papakostas & White 2000b).

Whilst the printing of polymer conductors on flexible substrates is well docu-

mented, relatively little research has been published on the use of flexible substrates for sensing purposes. By allowing the substrate to deform, higher stresses can be imparted to the sensor than is the case of an undeformed rigid substrate, and thereby exhibiting a higher sensitivity to load. Relevant approaches include the printing of polymer piezoelectric and piezoresistive thick film sensors on flexible polyester substrates (Papakostas & White 2000b, Papakostas 2001, Papakostas & White 2000c), and the printing of polymer piezoresistive sensors onto flexible epoxy membranes (Császár & Harsányi 1994). In the latter example, an applied pressure causes deformation of the membrane inducing surface strains, which are measured by the resistive elements.

Two patents explore the concept of a piezoresistive PTF sensor array on very thin flexible polymer substrates (Maness, Golden, Benjamin & Podoloff 1988, Maness, Golden, Benjamin & Podoloff 1989). Maness et al. (1988) proposes a resistor-resistor structure, which behaves in a highly non-linear switch-like manner, whilst Maness et al. (1989) details a similar structure employing proprietary piezoresistive inks to improve linearity. This approach is believed to be the core of a number of commercially available products[1] including: dental occlusion sensors; sensors for orthotic measurements; and industrial sensors for capturing component mating pressures. Such sensor arrays can have a total thickness as small as 0.1mm, and a sensor pitch of around 1mm. Figure 3.7 provides a simplified schematic.

The array comprises row and column electrodes separated by pressure sensitive resistive material, such that the intersection between a selected row and column is characterised by the resistance at that location. This structure allows efficient X-Y scanning of the array to extract individual sensor responses. A known voltage is applied to each row sequentially and during row activation, voltage is measured from each of the column electrodes. In this manner the temporal-spatial characteristics of applied pressure are captured.

3.4 PTF Sensor Considerations

From the review presented above, it is apparent that a number of pressure sensitive PTF mechanisms exist. The use of piezoresistive sensors to measure planar strains

[1] From Tekscan Inc., South Boston, MA, USA. Web: `http://www.tekscan.com`

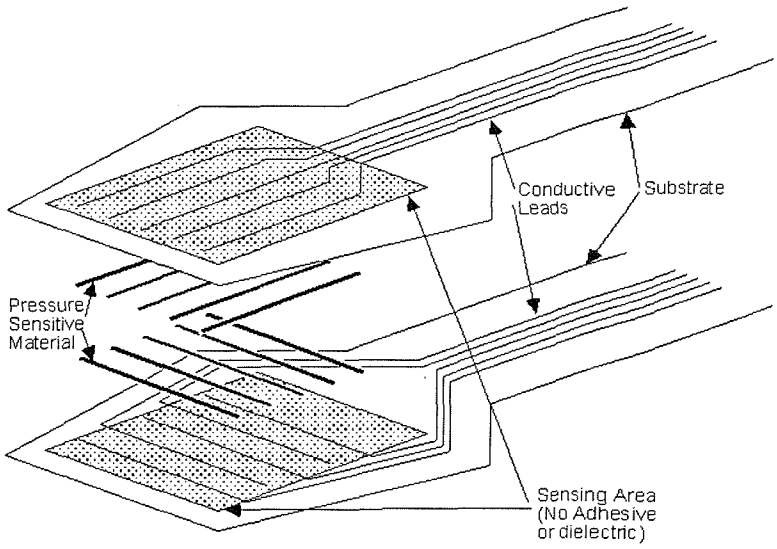


Figure 3.7: Piezoresistive Sensor Array (After TekScan Inc.)

is documented, and may be used to infer the pressure applied to a deformable substrate. PTF piezoelectric force/pressure sensors have been demonstrated on both alumina and polyester substrates.

As stated, PTF materials include conductors and dielectrics. This offers the possibility of realising capacitive sensing elements using insulated electrodes, as suggested by figure 3.8.

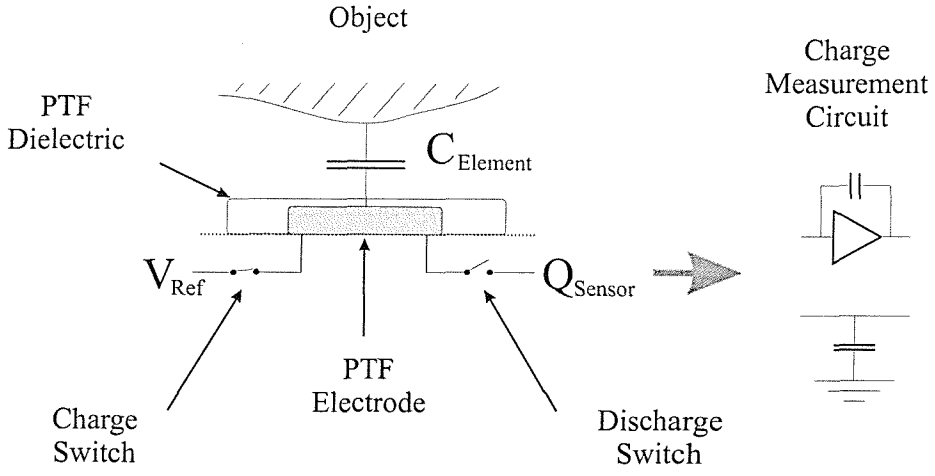


Figure 3.8: Realisation of PTF Capacitive Sensing Element

The PTF sensing element illustrated consists of an electrode pad insulated with dielectric material and forms the bottom half of a capacitor structure. The prox-

imity of conductive matter (for example the dermal layer of the skin – refer back to Figure 2.6) completes the structure. Such a sensing element is equivalent to the capacitive sensing mechanism outlined in Section 2.3.1 and Figure 2.9.

3.4.1 Sensor Architecture

Intuitively, the most effective arrangement of sensors on a smartcard is an array or grid formation, as a dense orderly packed array of sensing elements is most likely to fully capture the interaction between fingers and the smartcard. In this section the requirements for arrays of piezoelectric, capacitive and piezoresistive sensing elements are discussed.

Consider firstly that piezoelectric and capacitive sensing elements generate charge in response to pressure and proximity, respectively. Charge must be localised to each sensing element and all elements isolated from each other. This is the case in the capacitive silicon fingerprint sensor of Young (1997), as shown in Figure 3.9.

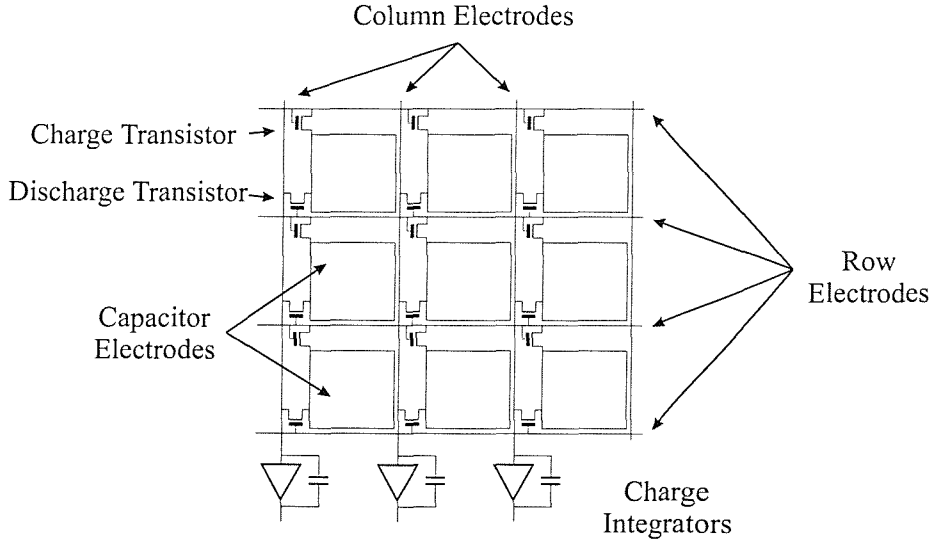


Figure 3.9: Capacitive Sensing Array (After Young (1997))

Here sensing elements are arranged into rows and columns, with each element containing two transistors. Considering the top-leftmost capacitor element, the *charge* transistor has the function of supplying charge to the capacitor electrode, whilst the *discharge* transistor is responsible for discharging the capacitor. The

gate and source of the charge transistor are connected to the upper row electrode, such that a voltage pulse both supplies and enables charge to flow to the capacitor electrode. The source and drain of the discharge transistor are connected to the capacitor electrode and column electrode respectively, whilst the gate terminal is connected to the lower row electrode. This pattern is repeated throughout the array, such that a voltage pulse on a row electrode simultaneously causes a charging of the lower row of capacitors and a discharging of the upper capacitor row. Scanning the array is performed by applying sequential voltage pulses to each row electrode individually, and charge is measured by charge integrators on a per column basis. This form of switched array is attractive because it is successfully able to localise charge to each capacitive element, whilst minimising the number of connectors required for the array.

Since polymer thick film technology offers the fabrication of passive components only, it provides no scope for implementing switching transistors nor indeed any other method of charge localisation. Hence, each charge generating PTF element is required to have an explicit connection to the array's perimeter whereupon its charge can be measured.

For an $m \times n$ array of sensing elements with explicit connections for each sensing element, mn connectors are required. Clearly this becomes impractical for large arrays. Moreover, the printing resolution of connectors becomes a primary limiting factor in the size and resolution of an array, as illustrated in Figure 3.10.

From Figure 3.10, it can be seen that the number of connectors leaving the array is limited by the inter-pixel spacing. If the minimum connector linewidth, λ_{min}, is equal to the minimum space between connectors, then the minimum connector pitch is $2\lambda_{min}$. From the architecture presented, half of the sensing elements are connected through each side of the array, and elements on the periphery may be connected directly, rather than through the inter-pixel space. Hence the number of connecting tracks required to pass through the inter-pixel space is $\frac{n}{2} - 1$ (for even n). Or conversely, given the distance between sensing elements, there will physically be space for $(Pixel\ Spacing)/2\lambda_{min}$ connecting tracks. It follows that the maximum number of sensing elements per row has an upper bound, given by

$$n_{max} \leq \frac{Pixel\ Spacing}{\lambda_{min}} + 2. \tag{3.2}$$

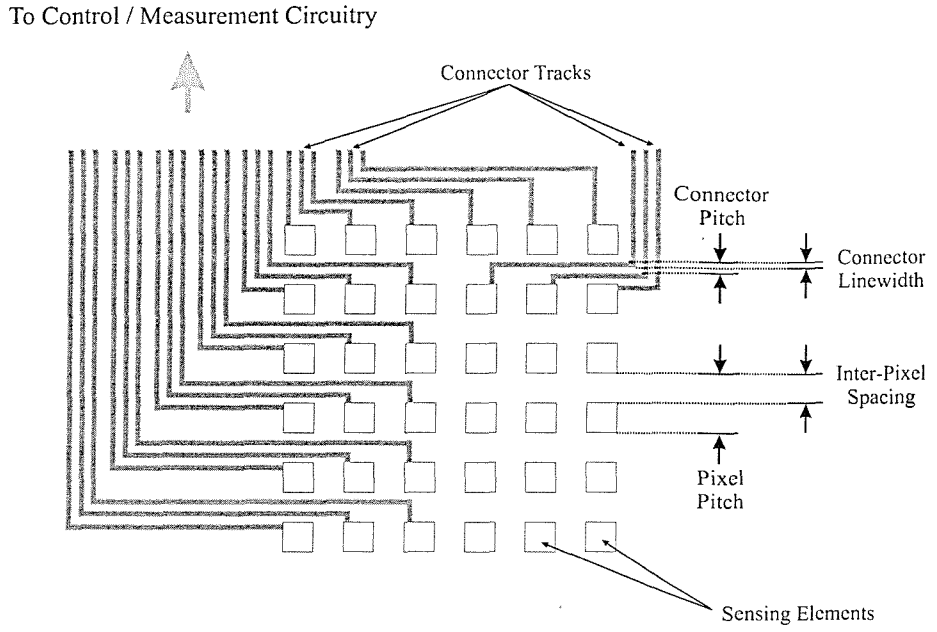


Figure 3.10: Partially Connected Array of Charge Generating Elements

As an example, consider a conventional screen process which prints with a minimum linewidth of 100μm and sensing elements spaced by a distance of 1mm. Then equation (3.2) shows that a maximum of 12 sensing elements can be connected per row. Assuming that element length is equal to the inter-pixel spacing, then this represents a sensing resolution of 5 pixels per centimetre, following the direction of columns.

An array need not be structured in the regular manner of Figure 3.10, and columns of sensors may be printed more closely together than row elements. The resolution of sensing elements in the direction following the rows of the array, is limited by pixel width and the line-spacing resolution of the printing process. Assuming 1mm^2 elements, and line-spacing is equal to linewidth (100μm) then sensor resolution along the rows of the array is around 9 pixels per centimetre.

One can imagine the concept of burying connective elements, such that conductor tracks pass directly underneath sensing elements and are insulated by a dielectric layer, as suggested by Figure 3.11. This is directly analogous to multi-layered PCB design. Using thick film processing, under this structure the array would be constructed using three layers, with the first layer containing conductive tracks, the second layer comprising of a dielectric material with vias, allowing electrode plates

of the third layer to connect with their associated conductive tracks. Conductive paste, printed during the third layer, would fill in the vias and forge connections to the first layer.

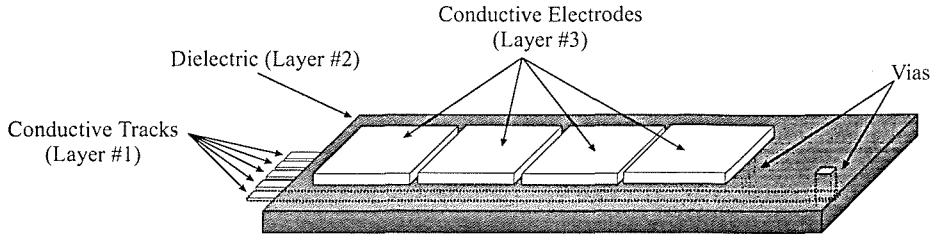


Figure 3.11: Row of Sensing Elements with Buried Conductive Tracks (Fifth Element removed to expose vias)

Figure 3.11 shows one half of a row of sensing elements: It is assumed that the elements belonging to the right hand half will have connective tracks in that direction. In this instance, the maximum number of sensing elements per row is given by

$$n_{max} \leq \frac{Pixel\ Spacing\ +\ Pixel\ Length}{\lambda_{min}}. \tag{3.3}$$

As an example, consider an array comprising of sensing elements 1mm in length with 1mm separation and a connector linewidth of 100μm. Then (3.3) shows that a maximum of 20 sensing elements can be connected per row. However, this calculation does not take into account the registration accuracy of the printing process, which could be as low as $\pm 50\mu$m (Gilleo 1995). Hence, a larger linewidth may be required to ensure connections between layers. Taking a linewidth of 150μm, equation (3.3) reveals a reduction to 12 sensing elements per row, in the above illustration.

It is possible to sacrifice the number of elements in a row for greater sensor coverage, if the inter-pixel spacing is reduced to the limit of the printing process, then fewer elements will fit along each row, but the array will be more dense and uniform. Taking 1mm^2 elements, in an array whose rows and columns are both separated by 100μm (0.1mm), each row will geometrically support 6 elements only. This represents a resolution of approximately 9 elements per centimetre, in both row and column directions.

Clearly there is a trade-off between the number of elements per row of an array, and the pixel-pitch (hence resolution) required to support this number of elements.

As a result, it is categorically not possible to capture high-resolution fingerprint characteristics using charge-generating polymer thick film sensors. Nevertheless, Chapter 2 identified a number of lower-resolution characteristics which potentially offer a basis for identity verification. These include finger-geometry, finger-crease pattern, section of palmprint, and some aspect of spatio-temporal grasping characteristics, and will be considered further.

Resistive sensing arrays (as per the array of TekScan) comprise of resistive intersections between row and column electrodes. As such the intersections are individually addressed by applying a known voltage to the appropriate row electrode and sensing the voltage from the desired column. This reduces the number of connective tracks required per sensing elements. For example, an $n \times n$ element array can be completely connected with $2n$ connective tracks. The pitch of sensors in this instance is limited by printing and registration resolution, rather than by the number of interconnecting tracks required to pass through the space between elements. If linewidths of 100μm and a registration accuracy of $\pm 50\mu$m between layers is assumed, then $150 \times 150\mu$m sensing elements are required. If these elements are separated by a distance of 100μm, then a pixel-pitch of 250μm, and a sensor resolution of 40 elements per centimetre, results. Since the ridge separation distance of fingerprints is reported to be around 330-400 μm (Cummins 1964), piezoresistive arrays are unlikely to offer sufficiently high resolution to fully capture fingerprint detail.

In addition to array architecture and the resolution of the PTF printing process, the mechanical properties of a smartcard will have an effect on the number and resolution of sensing elements. Compliant substrates, such as smartcards, deform in response to applied load, and resulting strains will be apparent across the substrate. These strains will be experienced by all sensors bonded to the substrate, and the degree to which strains propagate across the substrate will determine the proximity with which sensors may be placed to each other.

The next section describes the fabrication of PTF piezoresistive and piezoelectric pressure sensors, before presenting the detailed theoretical behaviour of these sensors. This theoretical analysis is employed in Chapter 4 to assess the effect of strain propagation on such pressure sensors, when bonded to smartcards.

3.5 Planar PTF Force Sensors

This section describes the construction and fundamental sensing principles of two approaches to pressure sensing. One sensor is based upon the piezoresistive principle, in which applied pressure causes a change in resistance, whilst the other is a piezoelectric sensor, generating charge in response to applied pressure. The planar geometry and inherent simplicity of these sensing structures is entirely consistent with smartcard integration.

3.5.1 Sensor Construction

Both sensors were printed onto 125μm thick flexible polyester[2], pre-shrunk by the manufacturer for thermal stability, and offering mechanical flexibility, good electrical characteristics, low moisture absorption, high solvent resistance and very low shrinkage at PTF processing temperatures (Gilleo 1995). These sensors are bonded onto smartcards for further analysis as is described in the next chapter.

Screen printing was performed using a DEK flatbed printer[3]. The piezoelectric sensor was constructed in the manner of (Papakostas & White 2000c). This is a three-layered device comprising of two silver conductive layers (ESL1107 from Electro Science Laboratories[4]) sandwiching an active PZT layer (85% Lead Zirconate Titanate and 15% phenolic resin by weight, 1:1 ratio by volume, providing optimal sensitivity and mechanical stability). Figure 3.12 provides a schematic, whilst the mask characteristics and dimensions are given in Figure 3.13.

The thickness of the PZT layer was measured to be 60μm using a stylus profilometer[5].

The piezoresistive sensor (see Figure 3.14) is a single resistive layer of polyimide-based, carbon-filled paste from Electro Science Laboratories (ESL RS15114, 10k$\Omega/\square$), whose thickness was measured to be 20μm. The dimensions of the active piezore-

[2]Mylar®, from DuPont Packaging and Industrial Polymers, Wilmington, USA. Web: `http://www.dupont.com`

[3]DEK 1750RS, from DEK Printing Machines Ltd. Weymouth, UK. Web: `http://www.dek.com`

[4]Electro Science Laboratories Inc. King of Prussia, PA, USA. Web: `http://www.electroscience.com`

[5]...

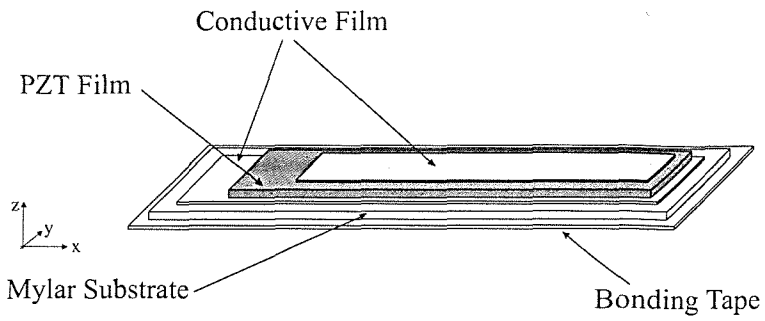


Figure 3.12: Piezoelectric Sensor Schematic

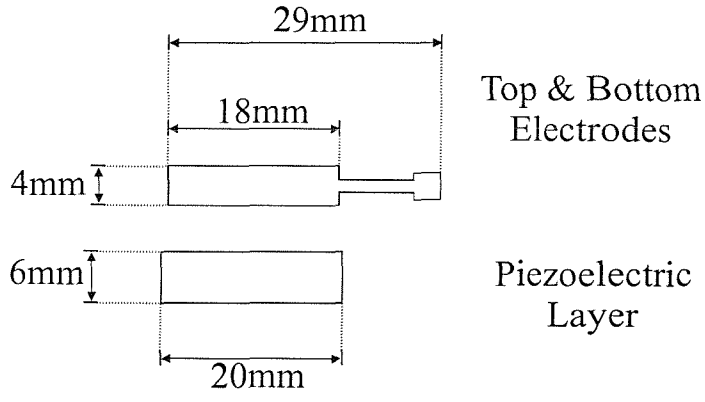


Figure 3.13: Piezoelectric Sensor: Mask Dimensions

sistive material between electrode connections were measured to be 15×9mm.

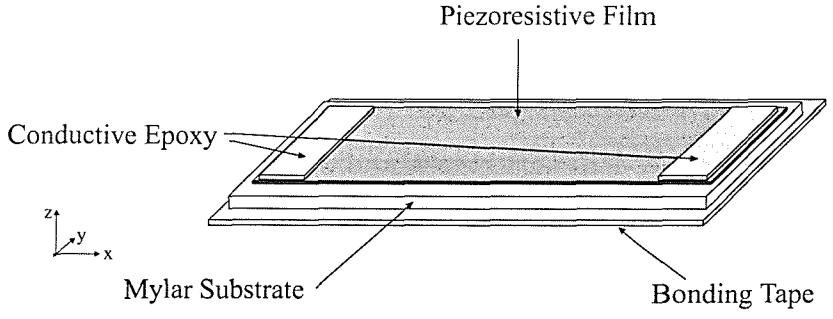


Figure 3.14: Piezoresistive Sensor Schematic

During construction, each layer was printed then dried for 10 minutes using a short wave infra-red dryer[6] at 110°C. Upon completion of the drying process the sensors were cured for 1 hour at 140°C. The piezoelectric sensor was polarized during the curing process with a poling voltage of 300V. Wire connections were made using

[6]DEK1209 IR Dryer

Circuitworks[7] conductive epoxy (CW2400) and 200μm diameter wire.

The resulting sensors are shown photographically in Figure 3.15, and these are bonded to smartcards as described in Chapter 4.

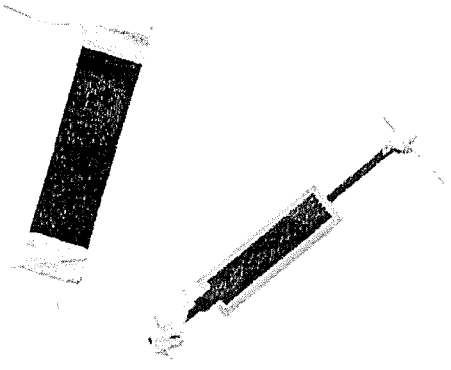


Figure 3.15: Piezoresistive and Piezoelectric (Left and Right) Sensors on polyester

3.5.2 Piezoresistive Principles of Sensing

As indicated, resistive components are comprised of conductive particles within a polymer matrix. Hence, electrical resistance is dependent upon the proximity and contact area of the conductive particles. As a force or pressure is applied to the material, strains are induced, causing a deformation of the structure and hence a change in the proximity of the conductive particles and their contact area. Such a change in resistance to the applied force or pressure is known as *piezoresistance.*

Prior to a detailed discussion of piezoresistivity, some definitions are required.

Stress, σ, is defined as

$$\sigma = \frac{F}{A} \qquad (in\ Pa) \tag{3.4}$$

where F is the force (in Newtons) applied to a normal cross-sectional area, A (m^2).

A body subjected to uniaxial stress deforms in response. For example, a body under compression contracts along the axis of applied stress and lengthens under tension. Also uniaxial stress causes materials to undergo transverse deformations, expanding in the plane normal to compression and contracting in response to

[7]Chemtronics Inc. Atlanta, GA, USA. Web: `http://www.chemtronics.com`

tension. These effects are illustrated in Figure 3.16, which displays the response of a block of thickness, t, breadth w and length l to an applied compressional stress, σ_z, in the z direction. The resulting changes in dimensions of the block are denoted δt, δw and δl respectively, here with δl, $\delta w \geq 0$ and $\delta t \leq 0$ (i.e. expansion in the x and y directions and contraction in the z direction.

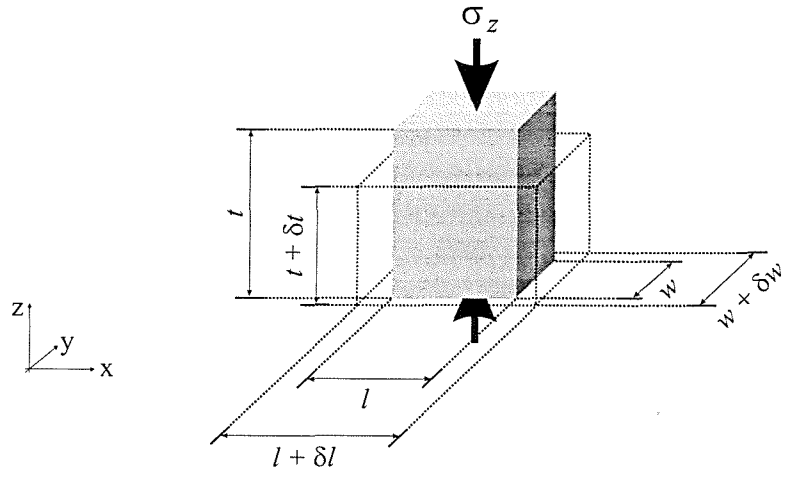


Figure 3.16: Material Deformation in Response to Compression

Deformations resulting from an applied stress are quantified by the proportional change in each orthogonal direction. For a change of thickness, δt, the strain, ε_z is defined as

$$\varepsilon_z = \frac{\delta t}{t} \tag{3.5}$$

where t is the undeformed thickness and ε_z is dimensionless.

Similarly, strains in the x and y directions are defined

$$\begin{aligned} \varepsilon_x &= \frac{\delta l}{l} \\ \varepsilon_y &= \frac{\delta w}{w} \end{aligned} \tag{3.6}$$

where l and w are the undeformed dimensions in the x and y directions respectively, and δl and δw are the deformational changes in length and breadth respectively.

If a body is subject to uniaxial stress, the magnitude of the transverse strains is governed by the fundamental material property known as *Poisson's ratio*. For a compressional stress, σ_z, in the z direction this is defined as the expansion per unit

breadth (or length) divided by the vertical contraction per unit thickness, i.e.

$$\nu = -\frac{\varepsilon_y}{\varepsilon_z} = -\frac{\varepsilon_x}{\varepsilon_z}. \tag{3.7}$$

Strain is related to stress by another fundamental material constant, called *Young's modulus*. This is defined as the stress per unit strain, and relates to the stiffness of a material. For a body subject to uniaxial stress, σ_z, the resulting strain, ε_z, is given by

$$\varepsilon_z = \frac{\sigma_z}{E} \tag{3.8}$$

where E is Young's modulus and (3.8) is known as Hooke's law.

Combining (3.7) and (3.8), the effect of applying a uniaxial stress in isolation can be seen as

$$\varepsilon_z = \frac{\sigma_z}{E} \Longrightarrow \varepsilon_x = -\nu\frac{\sigma_z}{E}, \quad \varepsilon_y = -\nu\frac{\sigma_z}{E} \tag{3.9}$$

for uniaxial stress, σ_z, in the z direction. Note that the strains in each direction are dependent on σ_z only since here we are considering the case of $\sigma_x = \sigma_y = 0$.

Similarly for a uniaxial stress σ_x in the x direction we have

$$\varepsilon_x = \frac{\sigma_x}{E} \Longrightarrow \varepsilon_y = -\nu\frac{\sigma_x}{E}, \quad \varepsilon_z = -\nu\frac{\sigma_x}{E} \tag{3.10}$$

and for uniaxial stress, σ_y,

$$\varepsilon_y = \frac{\sigma_y}{E} \Longrightarrow \varepsilon_x = -\nu\frac{\sigma_y}{E}, \quad \varepsilon_z = -\nu\frac{\sigma_y}{E}. \tag{3.11}$$

Expressions (3.9)-(3.11) were obtained for applying a stress in one of the orthogonal direction x, y or z only. Now for stresses with components in more than one direction, strains must be calculated using the general stress-strain relationships (Timoshenko & Goodier 1970). These are found by summing the strains in each orthogonal direction, arising from all stresses acting on the body (equations (3.9), (3.10) and (3.11)), giving for ε_x

$$\begin{aligned} \varepsilon_x &= \textit{Sum of strains in the x direction} \\ &= \frac{\sigma_x}{E} - \frac{\nu}{E}(\sigma_y + \sigma_z) \end{aligned} \tag{3.12}$$

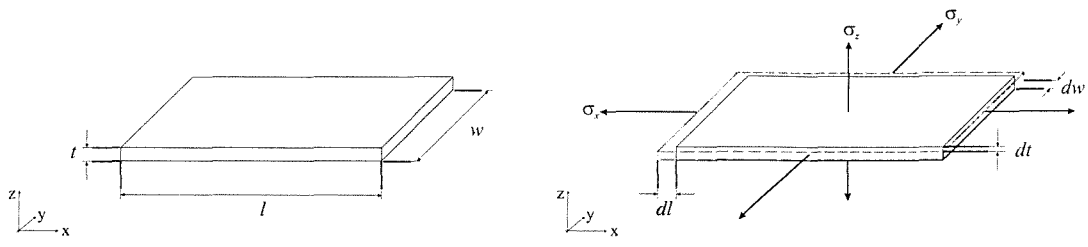
where ν and E are Poisson's ratio and Young's modulus for the material under

consideration, σ_x and σ_y are planar surface stresses and σ_z is the out-of-plane (normal) stress.

Similarly for ε_y and ε_z we obtain

$$\begin{aligned} \varepsilon_y &= \frac{\sigma_y}{E} - \frac{\nu}{E}(\sigma_x + \sigma_z) \\ \varepsilon_z &= \frac{\sigma_z}{E} - \frac{\nu}{E}(\sigma_x + \sigma_y) \end{aligned} \tag{3.13}$$

This situation is depicted in Figure 3.17.



(a) Sensor Dimensions (b) Response to Applied Stress

Figure 3.17: Stress-Strain Schematic

With the fundamental definitions in place, the description of a piezoresistive sensor begins by stating that the resistance, R, of a resistor, whose current flow is parallel to its length, is given by

$$R = \rho \frac{l}{wt} \tag{3.14}$$

where ρ is the resistivity of the resistor material and l, w and t are the length, width and thickness of the resistor in the x, y and z directions, respectively (Figure 3.17(a)).

The resistivity, ρ, of a polymer resistor is a function of both the temperature of the resistor and the applied stress. Thermal stress arises from the different thermal coefficients of expansion (TCE) of the polymer binders and conductive fillers, which results in expansion of the phases at different rates. However, at constant temperature resistivity is a function of external stresses only, resulting in the development of strains within the material. Resistivity can hence be described

as a function of the orthogonal strains within the material

$$\rho = f(\varepsilon_x, \varepsilon_y, \varepsilon_z) \tag{3.15}$$

with ε_x, ε_y and ε_z as defined in (3.5) and (3.6). It follows that

$$d\rho = \frac{\partial \rho}{\partial \varepsilon_x} d\varepsilon_x + \frac{\partial \rho}{\partial \varepsilon_y} d\varepsilon_y + \frac{\partial \rho}{\partial \varepsilon_z} d\varepsilon_z. \tag{3.16}$$

The *Piezoresistivity Coefficient*, G, in each of the three orthogonal directions is given by

$$\begin{aligned} G_x &= \frac{1}{\rho} \frac{\partial \rho}{\partial \varepsilon_x} \\ G_y &= \frac{1}{\rho} \frac{\partial \rho}{\partial \varepsilon_y} \\ G_z &= \frac{1}{\rho} \frac{\partial \rho}{\partial \varepsilon_z}. \end{aligned} \tag{3.17}$$

However, since PTF materials consist of randomly dispersed particles in a polymer matrix, it is reasonable to assume that the films are isotropic. That is

$$G_x = G_y = G_z = G. \tag{3.18}$$

Multiplying each side of the expressions in (3.17) by ρ and substituting into (3.16) gives

$$d\rho = \rho\, G d\varepsilon_x + \rho\, G d\varepsilon_y + \rho\, G d\varepsilon_z. \tag{3.19}$$

Differentiating (3.14) gives

$$\begin{aligned} dR &= \frac{\partial R}{\partial \rho} d\rho + \frac{\partial R}{\partial l} dl + \frac{\partial R}{\partial w} dw + \frac{\partial R}{\partial t} dt \\ dR &= \frac{l}{wt} d\rho + \frac{\rho}{wt} dl - \frac{\rho l}{tw^2} dw - \frac{\rho l}{wt^2} dt. \end{aligned} \tag{3.20}$$

Now, since $R = \rho l / wt$, (3.20) reduces to

$$\frac{dR}{R} = \frac{d\rho}{\rho} + \frac{dl}{l} - \frac{dw}{w} - \frac{dt}{t}. \tag{3.21}$$

Substituting (3.19) into (3.21) gives

$$\frac{dR}{R} = G\left(d\varepsilon_x + d\varepsilon_y + d\varepsilon_z\right) + \frac{dl}{l} - \frac{dw}{w} - \frac{dt}{t}. \tag{3.22}$$

But, by definition, $\frac{dl}{l} = \varepsilon_x$, $\frac{dw}{w} = \varepsilon_y$ and $\frac{dt}{t} = \varepsilon_z$. Hence, the proportional resistance change as a function of applied strains is

$$\frac{dR}{R} = G\left(d\varepsilon_x + d\varepsilon_y + d\varepsilon_z\right) + \varepsilon_x - \varepsilon_y - \varepsilon_z. \tag{3.23}$$

For a sensor bonded to a smartcard, and is thin in comparison to the card, then the planar strains of the sensor conform to the planar strains of the card. That is

$$\varepsilon_{x\,(f)} = \varepsilon_{y\,(c)} \tag{3.24}$$

$$\varepsilon_{y\,(f)} = \varepsilon_{y\,(c)} \tag{3.25}$$

where the labels $_{(c)}$ and $_{(f)}$ are used to denote the card and sensor film, respectively.

The out-of-plane, normal strain, $\varepsilon_{z\,(f)}$, is found by solving (3.12 & 3.13), to give

$$\varepsilon_{z\,(f)} = -\frac{\nu_f}{(1-\nu_f)}\left[\varepsilon_{x\,(c)} + \varepsilon_{y\,(c)}\right] + \left[1 + \frac{2\nu_f^2}{(1-\nu_f)}\right]\frac{\sigma_z}{E_f}. \tag{3.26}$$

Equation (3.23) shows clearly that a piezoresistive sensors exhibit sensitivity to planar strain. Since a PTF sensor is thin in comparison to a smartcard, it may be expected to conform to the planar strains of the smartcard, and hence, the flex of a card has a significant influence on the response of a sensor bonded to it. Chapter 4 makes use of finite element analysis to calculate the planar strains of a smartcard resulting from applied loads. These results are then used in conjunction with equations (3.23), (3.24) and (3.26), to calculate the response of sensors bonded to its surface, from which the properties of sensors bonded to a smartcard's surface can be assessed.

3.5.3 Piezoelectric Principles of Sensing

The direct piezoelectric effect describes the ability of certain crystalline materials to generate electric charge in proportion to an applied stress. The characteristics

of such a material is specified by its piezoelectric coefficients, relating the charge generated per unit stress. Piezoelectric materials exhibit different properties depending upon the direction of applied stress and the direction in which the electric field is measured. Hence, piezoelectric coefficients are fully specified by two indices, the first referring to the direction of the electric field, and the second corresponding to the direction of applied stress. Conventionally, the label $_3$ is used to indicate the polling axis, and labels $_1$ & $_2$ are arbitrarily chosen orthogonal axes, in the plane normal to $_3$. For example, d_{33}, is the piezoelectric coefficient which describes the charge generated in response to applied stress in the direction of the polling axis.

For the sensors used in this work, the polling axis is normal to the plane of the sensor, as shown in Figure 3.18.

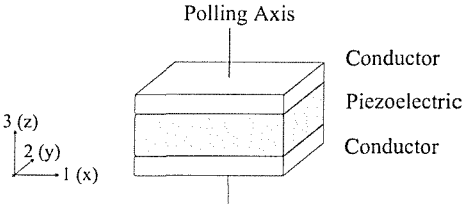


Figure 3.18: Piezoelectric Sensor Axes

Hence, the piezoelectric coefficients, of relevance to this work, are stated for constant electric field, Ξ, as

$$d_{3\,i} = \left(\frac{\partial D_3}{\partial \sigma_i}\right)_{\Xi} \tag{3.27}$$

where the $_i$ subscript indicates direction, and D_3 is known as the electric displacement, and is given as

$$D_3 = \frac{Q}{A} \tag{3.28}$$

and Q is the charge (Coulombs) generated over an area A (m^2).

It can be shown (Lefki & Dormans 1994) that

$$D_3 = \epsilon_{33}^{\sigma}\Xi + d_{31}\left(\sigma_{x(f)} + \sigma_{y(f)}\right) + d_{33}\sigma_{z(f)} \tag{3.29}$$

where ϵ_{33}^{σ} is the *permittivity* of the piezo material in the direction normal to the plane of the sensor and under constant stress. However, as a consequence of the conditioning circuit used in this work (Appendix B), the electric field across the

sensor, $\Xi = 0$. Hence, from (3.29) and (3.28), it follows

$$Q = A\left[d_{31}\left(\sigma_{x(f)} + \sigma_{y(f)}\right) + d_{33}\sigma_{z(f)}\right]. \tag{3.30}$$

For a sensor bonded to a smartcard, the stresses $\sigma_{x(f)}$ and $\sigma_{y(f)}$ are the stresses acting upon the PZT film, resulting from the planar strains of the card $\varepsilon_{x\,(c)}$ and $\varepsilon_{y\,(c)}$, whilst $\sigma_{z(f)}$ is the normal stress applied to the sensor.

For an unconstrained sensor under the influence of a compressive normal stress, $\sigma_{z(f)}$, the orthogonal planar strains, $\varepsilon_{x\,(unconstrained\,sensor)}$ and $\varepsilon_{y\,(unconstrained\,sensor)}$, are given by restatement of (3.9), to be

$$\begin{aligned} \varepsilon_{x\,(unconstrained\,sensor)} &= -\nu_f \frac{\sigma_{z(f)}}{E_f} \\ \varepsilon_{y\,(unconstrained\,sensor)} &= -\nu_f \frac{\sigma_{z(f)}}{E_f}. \end{aligned} \tag{3.31}$$

However, for a sensor bonded to a smartcard, the reactive planar stresses of the card restrict the unconstrained deformations given in (3.31), and shown in figure 3.19.

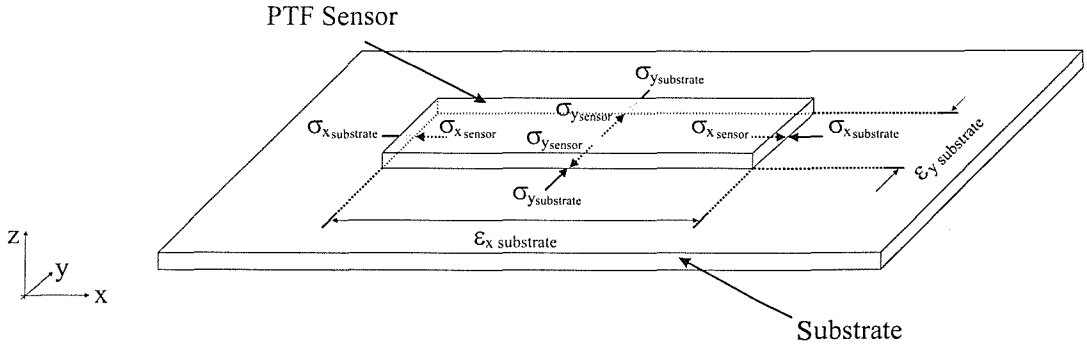


Figure 3.19: External Planar Stresses on Sensor

The planar strains of the sensor can be expressed in terms of the stresses acting on the film using equations (3.12 & 3.13), it is found that

$$\begin{aligned} \varepsilon_{x\,(f)} &= -\nu_f \frac{\sigma_{z(f)}}{E_f} + \frac{\sigma_{x\,(f)} - \nu_f \sigma_{y\,(f)}}{E_f} \\ \varepsilon_{y\,(f)} &= -\nu_f \frac{\sigma_{z(f)}}{E_f} + \frac{\sigma_{y\,(f)} - \nu_f \sigma_{x\,(f)}}{E_f}. \end{aligned} \tag{3.32}$$

The calculation of charge (equation (3.30)) requires that the planar stresses acting on the sensor ($\sigma_{x\,(f)}$ and $\sigma_{y\,(f)}$) are known. These are found using (3.24) and solving (3.32), to be

$$\sigma_{x\,(f)} = \frac{E_f}{1-\nu_f^2}\left(\varepsilon_{x\,(c)} + \nu_f \varepsilon_{y\,(c)}\right) + \frac{\nu_f}{1-\nu_f}\,\sigma_{z(f)} \tag{3.33}$$

and

$$\sigma_{y\,(f)} = \frac{E_f}{1-\nu_f^2}\left(\varepsilon_{y\,(c)} + \nu_f \varepsilon_{x\,(c)}\right) + \frac{\nu_f}{1-\nu_f}\,\sigma_{z(f)}. \tag{3.34}$$

This means that given the strains of the smartcard ($\varepsilon_{x\,(c)}$ and $\varepsilon_{y\,(c)}$), and the normal stress upon the sensor ($\sigma_{z(f)}$), the charge generated by a piezoelectric sensor can be found using (3.30) with the planar stresses, $\sigma_{x\,(f)}$ and $\sigma_{y\,(f)}$ given by equations (3.33) and (3.34).

3.6 Considering Sensors Bonded on Smartcards

The goal of this work on polymer sensors is to assess their potential to capture discriminating human characteristics. It has been shown that both piezoresistive and piezoelectric sensors exhibit sensitivity to planar strains of the substrate to which they are bonded. Moreover, it is known from experience that usual handling of a smartcard causes the card to flex, and this will generate strains within the smartcard. If an array of sensors is to capture physical characteristics, such as finger or thumb geometry, then it is desirable that they exhibit sensitivity to normal, rather than planar strains. The extent to which strains propagate across the smartcard determines how many sensors can be bonded to the surface, and how closely spaced they can be without suffering cross-sensitivities.

The preceding analysis of piezoresistive and piezoelectric sensor principles has made possible the calculation of their responses to known loads, given the planar strains of the smartcard to which they are bonded. As a load is applied to a sensor, compression of the sensor occurs, and the smartcard deforms in response. If the assumption is made that the sensor is thin compared to the smartcard, then the sensor conforms to the planar strains of the card. This is as stated in (3.24).

For the piezoresistive response, the surface strains, $\varepsilon_{x(c)}$ and $\varepsilon_{y(c)}$, and the normal

load, σ_z, can be used to calculate the normal strain of the sensor, ε_z, as stated in (3.26). The three orthogonal strains can then be used to calculate the proportional change in resistance, $\frac{dR}{R}$, as given in (3.21).

The piezoelectric sensor response, on the other hand, requires that the stresses applied to the sensor ($\sigma_{x(f)}$ and $\sigma_{y(f)}$), by the smartcard, are known. These are calculated using (3.33) and (3.34) with the planar strains of the substrate, $\varepsilon_{x(c)}$ and $\varepsilon_{y(c)}$, and the applied stress, $\sigma_{z(f)}$. Equation (3.30) then gives the charge response of the sensor.

The next chapter makes use of finite element analysis to calculate the surface strains of a smartcard in response to what may be expected as typical loads. Making use of the above sensor models, the characteristics and behaviour of these sensors, bonded to smartcards, are assessed.

3.7 Concluding Remarks

In this chapter a brief review of thick film technology is presented. Polymer thick films offer the advantage of low processing temperatures, and as a result can be printed onto a wide range of substrates, including flexible polymers. This property of flexibility, and in conjunction with those of planar geometry, mechanical robustness and low cost, make PTF sensors inherently suitable for integration with smartcards.

A review of thick film processing, and in particular, polymer thick films are considered. It is apparent that the intrinsic physical properties of PTF materials can be exploited for sensing purposes, and the fabrication of PTF piezoresistive and piezoelectric pressure senors is reported.

Since a dense orderly structured array of sensing elements is most likely to capture spatial finger characteristics, array architectures of resistive and charge-generating sensor elements is given detailed consideration. Piezoresistive elements can be connected on a per row and per column basis, are be addressed individually by measuring the voltage drop across a specific intersection. Spatial resolution of such an array is limited by the linewidth and registration resolution of the printing process. It is estimated that a minimum pixel pitch of around 250μm is achievable,

resulting in a spatial resolution of around 40 elements per centimetre.

PTF technology allows for the fabrication of entirely passive components, and offers no mechanism with which the charge generated by piezoelectric sensor elements can be localised. This property precludes the construction of switched sensing arrays in the manner of Young (1997) or the CCD array structures of Edwards (1984) and Tartagni & Guerrieri (1998). Hence, each individual charge-generating element must explicitly be connected to outside of the array, whereupon its charge can be measured. This leads to the situation whereby conductive tracks must either pass between, or underneath sensing elements, and hence, the spacing of elements is governed by the number of elements per row of an array. The design of a charge-generating PTF sensor array then becomes a trade-off between the required number of sensing elements and the desired spatial resolution.

From an entirely geometrical perspective, neither piezoelectric nor piezoresistive arrays offer sufficiently high resolution to capture the fine detail of a fingerprint. It may however be possible to construct arrays which are suitable for capturing alternative spatial characteristics, such as finger-geometry, finger-crease pattern or a section of palmprint.

Detailed theoretical models of both sensors are derived, and it is apparent that both sensors will exhibit sensitivity to planar strains. Since the flex of a smart-card results in planar strains propagating across the card, it is likely that planar sensitivity of the sensors, rather than array architecture, will become the limiting factor to spatial resolution. The extent of this effect is investigated in the next chapter.

A new implementation of a capacitive sensor using polymer thick film has been proposed, and consists simply of a base electrode, insulated by a dielectric layer. In the manner of the fingerprint sensors of Young (1997) and Tartagni & Guerrieri (1998), the presence of a conductive object (such as the dermal layer of the skin), completes the capacitor structure. A known voltage is applied, and charge accumulates on the base electrode, in proportion to the proximity of a conductive object. Whilst further consideration of this approach must be deferred until Chapter 7, an array of such charge-generating elements is required to satisfy the architectural limitations of piezoelectric arrays.

The conclusions of Chapter 2 indicate that thermal characteristics can be useful in the capture of discriminatory characteristics. Although pyroelectric cermet thick films have been shown to exhibit thermal sensitivity, the development of a suitable polymer pyroelectric material is beyond the scope of this thesis and will not be considered further.

Chapter 4

PTF Sensors On Smartcards

4.1 Introduction

The simple planar polymer thick film sensors described in Chapter 3 are attractive for smartcard integration because of their properties of low-cost, mechanical flexibility, robustness, and their thin, planar nature. Earlier work in Section 3.4 assessed printing resolution and array architectures of both charge generating piezoelectric, and resistive piezoresistive sensors. From this analysis it is clear that fingerprint sensors are not feasibly implemented using thick film technology. Nevertheless, from the review of biometric characteristics in Chapter 2, there exists a number of less demanding alternative approaches to demonstrating identity. These include finger-geometry, finger-crease pattern and palmprint characteristics.

It is envisaged that a smartcard owner will hold the card in one hand, and present the characteristics of the other hand to an on-card sensor for verification. However, presentation of any of the above characteristics involves applying a load to the smartcard, which from common experience will cause the card to flex, and strains will be induced across its surface. But theoretical analysis of both sensor types indicate sensitivities to planar strain, implying that sensors will respond to forces applied elsewhere on the card. From the perspective of capturing any of the above human traits, where spatial (or spatio-temporal) features form the basis for discrimination, this is clearly undesirable.

The work of this chapter assesses the extent to which planar strains propagate

across a smartcard, and the resulting effect this has on PTF sensors bonded to the surface of a card. This is performed using first-order finite element analysis (FEA), and is verified experimentally. Loads are applied to a finite element smartcard model, and surface stresses computed. The strains experienced by sensors bonded to smartcards are approximated, and used to predict sensor responses. This work forms the invited paper of Henderson, Papakostas, White & Hartel (2002).

This analysis is not concerned with absolute sensor responses, rather, its aim is an insight into the feasibility of using PTF pressure sensors to capture discriminating human characteristics. For this reason, both smartcard and sensor models are validated together, and it is the combined properties of smartcard and sensor which are measured.

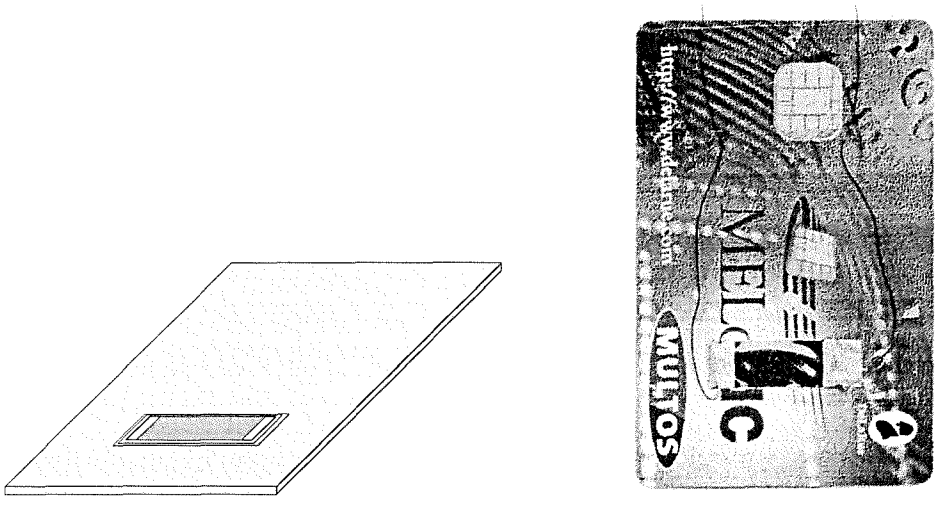
In addition, a number of assumptions have been made about the material properties of the smartcard and the sensors. Facilities to measure Young's modulus and Poisson's ratio of both sensors were unavailable to this work, and as a result assumptions have necessarily been made. These are clearly stated and references to relevant measurement techniques are given. It hence follows that predicted sensor responses are approximate. Nevertheless, the work of this chapter strongly indicates the effect of a smartcard flexing on both sensor types, and this is reinforced by experimental verification.

The chapter concludes with a study of the tactile interaction between user and smartcard. User interactions are limited but may include a person touching, pressing or tapping the smartcard. Both piezoelectric and piezoresistive sensors are assessed and compared in these respects. It is believed that this is the first time polymer thick-films have been assessed for identity verification purposes, and this is documented in Henderson, Papakostas, White & Hartel (2001).

4.2 Bonding Sensors onto Smartcards

The piezoelectric and piezoresistive sensors, described in Section 3.5.1, are printed onto 125μm polyester, and bonded to the top surface of two smartcard blanks.

This is performed using 50μm double-sided adhesive tape[1], and the result is shown schematically in Figures 4.1(a) & 4.2(a) and photographically in Figures 4.1(b) & 4.2(b) for the piezoresistive and piezoelectric sensors respectively.



(a) Schematic (b) Actual Smartcard

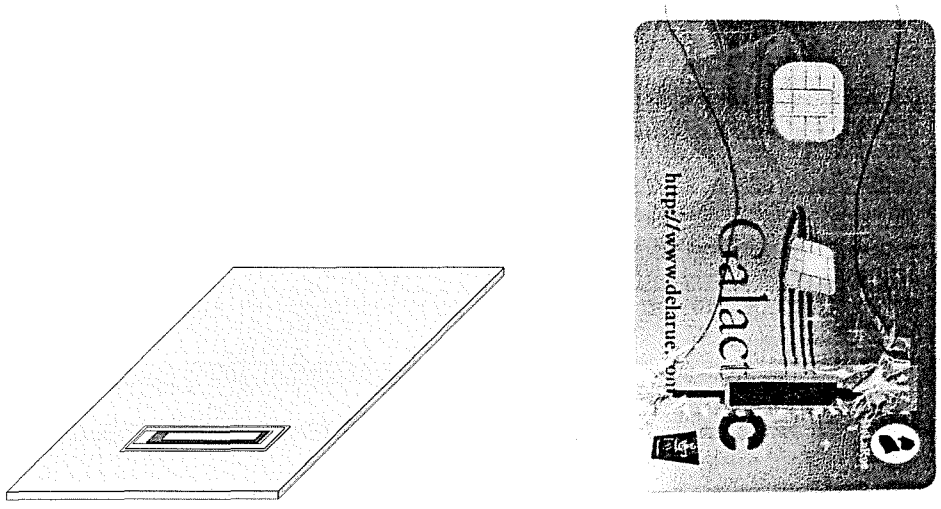
Figure 4.1: Piezoresistive Sensor Bonded to Smartcard

It should be noted, that whilst these experimental sensors are bonded to the surface of the smartcards, their thin planar dimensions are entirely compliant with the processes of modern smartcard manufacture. Smartcards are manufactured by the lamination of polymer foils (Rankl & Effing 1997), and it is quite conceivable that a sensor layer could be incorporated into the process.

4.3 Material Properties

Before commencing any finite element modelling, it is important to specify the material properties of both smartcard and sensors. The smartcard finite element model requires that Poisson's ratio, Young's modulus (as defined in Section 3.5.2), and the mass density of the smartcard are known. Poisson's ratio, Young's modulus, and additionally, piezoresistive and piezoelectric coefficients are required to

[1]Scotch Double Sided Tape, manufactured by 3M United Kingdom PLC, Bracknell, UK. Web: `http://www.3m.com`



(a) Schematic (b) Actual Smartcard

Figure 4.2: Piezoelectric Sensor Bonded to Smartcard

calculate sensor responses.

The smartcard blanks used in this are predominantly constructed of Polyvinylchloride (PVC). As indicated, the manufacture of smartcards involves lamination of a number of foil layers. The fine detail of this process and any other materials used in bonding or finishing, remain proprietary, and it is hence difficult to know the precise composition of the smartcards under test. For this reason, the approach taken here is to use the material properties of bulk PVC, except where greater accuracy is accessible through simple experimental testing. For example, Young's modulus is determined through straightforward experiment in the next section. Mass density and Poisson's ratio are taken to be those of bulk PVC, with $\rho_{smartcard}$ =1.38×10^3 kg/m^3, and $\nu_{smartcard}$ = 0.38, respectively (Callister 1997).

Poisson's ratio of polymer materials range between $\nu_{polymer}$ = 0.35 and $\nu_{polymer}$ = 0.45 (Callister 1997). In the absence of facilities to measure Poisson's ratio, we make the assumption that $\nu_{pR} = \nu_{pE} = 0.4$, where the subscripts pR and pE are used to denote the piezoresistive and piezoelectric sensors, respectively, and 0.4 is the mean of the range specified above.

4.3.1 Young's Modulus (Smartcard)

There are a number of standard methods for the measurement of Young's modulus (Callister 1997), including: *tension testing*; *cantilever deflection*; *torsion testing*; and *vibration resonance testing*. Perhaps the most straightforward approach is that of cantilever deflection, whereby the material under investigation is fashioned into a beam and supported at one end only. Known loads are applied to the free end, and its deflection measured. It is this method which is employed here, and Appendix A provides a detailed description of the mechanics.

Clamping and Loading Conditions

The measurement setup is shown in Figure 4.3.

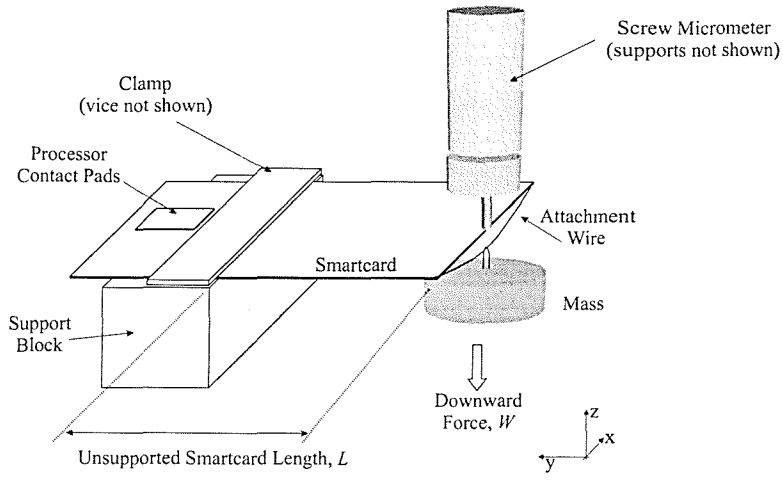


Figure 4.3: Measurement of Young's Modulus

Two rectangular alumina strips (of dimensions 101×12×0.75 mm), were used to sandwich the width of the card, ensuring that the clamping constraints were uniformly applied. A mechanical vice was then used to clamp this arrangement onto the measurement structure. It should be noted that the clamp was applied centerwards of the processor contacts, such that their contribution towards longitudinal bending stiffness will not affect these measurements.

Loads were attached to the free end of the card by means of fine-gauge wire. The wire was bonded to the bottom surface of the card, allowing a loop of wire to be

formed underneath the card. Measured loads were then attached to the underside loop. No discernable buckling across the width of the card was observed, and it was assumed that the load was uniformly applied across the width.

Vertical displacements were measured using a screw-micrometer with a resolution of 5μm.

Experimental Measurement of Young's Modulus

Three measured loads were applied to six smartcards from the same batch, in the manner outlined above. The resulting vertical deflections for each case were measured a number of times, and Young's modulus calculated according to (A.5). Using standard error analysis (Morris 1988), Young's modulus was calculated to be 3.1 $\pm$ 0.5 GPa with a standard deviation of 0.19 GPa. This is approximately 15% stiffer than bulk PVC, whose Young's modulus is 2.7GPa, and the difference is most likely explained by the lamination and adhesive treatments experienced during smartcard manufacture.

4.3.2 Young's Modulus (Sensors)

Sensor films printed onto substrates form composites, and as a result the straightforward approach of cantilever deflection can not be used. It is possible to use *composite beam theory* (Callister 1997) to describe the bending characteristics of a thick film printed upon a substrate, however, this approach requires that the substrate be uniformly smooth in comparison to the thickness of the film. For this reason, silicon is an appropriate substrate material for such a measurement, and an attempt to measure Young's modulus of a thick film, printed on silicon is given in Grabham (2002).

An alternative approach is to use the simple rule of mixtures for composite materials (Callister 1997), which is stated as

$$E_{composite} = P_1 E_1 + P_2 E_2 \tag{4.1}$$

where $E_{composite}$ is Young's modulus of the composite mixture, P_1 is the volume

proportion of material 1 with modulus E_1, and P_2 is the volume proportion of material 2 with modulus E_2.

Since this work represents a first-order analysis of thick film sensor responses to smartcard flexing, (4.1) is used to estimate Young's modulus of both sensor types.

Young's modulus of the piezoelectric sensor is specified in equations (3.33) and (3.34), and is required to calculate the planar stresses applied to the sensor by the card. As stated in Section 3.5.1, the PZT film is composed of a 1:1 mixing ratio of PZT ceramic to phenolic binder. Whilst Young's modulus of the phenolic resin binder is around 4.8GPa (Matweb 2002) and approximately 48GPa for PZT-5H ceramic (Morgan-Matroc 1998). Hence Young's modulus of the composite piezoelectric sensor, ν_{pE}, is estimated to be 26.4GPa.

Young's modulus of the piezoresistive sensor is only relevant during the case of a load applied directly onto the sensor (ie. $\sigma_z \neq 0$), as is apparent from equation (3.26). As stated the piezoresistive layer is comprised of carbon particles within a polyimide matrix. The precise composition of this paste is proprietary, and hence unknown. However, Young's modulus for carbon (graphite) and polyimide are similar, having values of 4.8Gpa and between 2.5-4.1GPa, respectively (Matweb 2002). Taking the median value of 3.3GPa for Young's modulus of polyimide, Table 4.1 shows Young's modulus for composite mixtures based on different volume ratios.

Carbon (% Volume)	Polyimide (% Volume)	$E_{composite}$ (GPa)
20	80	3.6
50	50	4.05
80	20	4.5

Table 4.1: Young's Modulus for Piezoresistor with Mixing Ratio

From the data of Table 4.1, it appears that making the assumption of a 1:1 mixing ratio is unlikely to be in excess of around 10% error. Hence the value of 4.05GPa is used for Young's modulus of the piezoresistive sensor.

4.3.3 Piezoresistivity Coefficient, G

In order to calculate the piezoresistive response to known strains, it is apparent from (3.23) that the piezoresistivity coefficient, G, of the resistive ink must be known. Unfortunately, this quantity is not ordinarily quoted on data sheets. However, G is related to the *gauge factor*, as indicated by equations (3.1) and (3.23).

Arshak et al. (1995) measured and found the longitudinal gauge factor, GF_L, of this resistive ink[2] to be between 4 and 5. However, these measurements were made for PTF resistors printed on alumina substrates. As Papakostas & White (2000a) point out, the material properties of a substrate affects the gauge factor of a piezoresistor. If the film sensor is thin compared to the thickness of the substrate, then it can be assumed that the planar strains developed within the film conform to the planar strains of the substrate, whilst the out-of-plane strains depend upon the properties of the film. This situation is illustrated in figure 4.4.

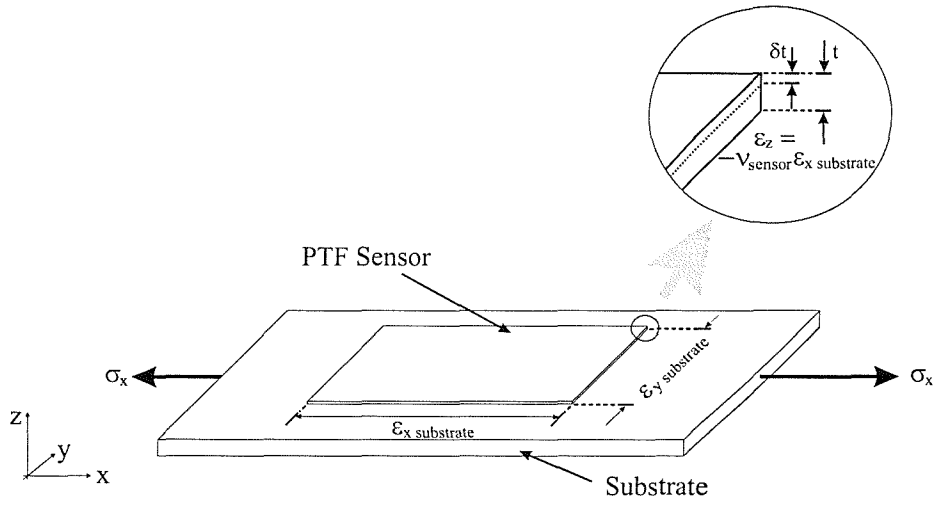


Figure 4.4: Dependency of Internal Film Strains on Substrate Characteristics

If a longitudinal strain, $\varepsilon_{x\,(substrate)}$ is the result of a uniaxial stress, σ_x, applied to a substrate then the resulting strain in the other planar direction, $\varepsilon_{y\,(substrate)}$ is related through Poisson's ratio,

$$\varepsilon_{y\,(substrate)} = -\nu_{substrate}\,\varepsilon_{x\,(substrate)}. \tag{4.2}$$

[2]RS 15114, from ESL

Assuming that the resistive film is thin compared with the substrate, then

$$\begin{aligned} \varepsilon_{x\,(f)} &= \varepsilon_{x\,(substrate)} \\ \varepsilon_{y\,(f)} &= -\nu_{substrate}\,\varepsilon_{x\,(substrate)}. \end{aligned} \tag{4.3}$$

The out-of-plane, normal strain, $\varepsilon_{z\,(f)}$, is found by solving (3.12 & 3.13) with (4.3) to be

$$\varepsilon_{z\,(f)} = -\frac{\nu_f}{(1-\nu_f)}\left[\varepsilon_{x\,(substrate)} + \varepsilon_{y\,(substrate)}\right]. \tag{4.4}$$

Using (4.4) into (3.23), combining and rearranging with (3.1), it is found that (Papakostas & White 2000a)

$$G = \frac{GF_L - 2}{1 - \nu_{substrate} - \nu_f \frac{1-\nu_{substrate}}{1-\nu_f}} + 1. \tag{4.5}$$

Taking Arshak's median measurement of $GF_L = 4.5$, and using $\nu_{substrate} = 0.22$ for alumina (Matweb 2002) and $\nu_{sensor} = 0.4$, it is found that $G = 10.62$. At the extremes of Arshak's measurements, $G = 8.69$ for $GF_L = 4$ and $G = 12.54$ for $GF_L = 5$.

4.3.4 Piezoelectric Coefficients, d_{33} & d_{31}

Papakostas (2001) reports on the development and characterisation of the piezoelectric sensors used in this work. The d_{33} and d_{31} coefficients were found to be between 7 to 11pC/N and -3 to -5pC/N, respectively. Hence the median values of d_{33} = 9pC/N and d_{31} = -4pC/N, are used in this analysis.

4.3.5 Material Properties Summary

Table 4.2 summarises the material properties used in this modelling work.

Property	Assumed Value	Source	Additional Information
E_c	3.1±0.5GPa	(Callister 1997)	Cantilever Deflection
ν_c	0.38	(Matweb 2002)	-
ρ_c	1.38×10^3kg/m^3	(Matweb 2002)	-
E_{pR}	4.1GPa	(Callister 1997)	Rule of mixtures
ν_{pR}	0.4	(Matweb 2002)	-
G	10.6	(Papakostas & White 2000a) (Arshak et al. 1995)	Inferred
E_{pE}	26.4GPa	(Callister 1997)	Rule of mixtures
ν_{pE}	0.4	(Callister 1997)	-
d_{33}	9pC/N	(Papakostas 2001)	-
d_{31}	-4pC/N	(Papakostas 2001)	-

Table 4.2: Summary of Material Properties

4.4 Smartcard Finite Element Model

It is the intention of this work to capture the general mechanical characteristics, rather than absolute detail, of a smartcard responding to an applied load. As a result some aspects of the real smartcard will be omitted, including the processor contact pads (see Figures 4.1(b) and 4.2(b)), the underlying laminate construction (Section 4.2) and the curved corners of a real card. This model treats the smartcard as a simple rectangular plate of uniform thickness and homogenous material.

Finite element analysis allows the modelling of complex geometrical structures by subdividing (or *discretising*) the structure into small, deterministic units called *elements* (Fagan 1992). The characteristics of elements, and hence their appropriateness to particular geometries, are defined by their number and relative positions of *nodes* and their degrees of freedom. The degrees of freedom specify the manner with which nodes are free to move relative to other nodes within the element, and include translational and/or rotational components. For example, 3-D solid elements can have cuboidal or tetrahedral shapes defined by 8 and 4 translational nodes, respectively. Hence, solid objects require to be stacked on top of each other to describe the bending behaviour of a structure. Shell elements, on the other hand, use nodes with both translational and rotational properties, and are hence able to adequately describe bending with a lesser number of nodes. It is

the material properties attributed to each element which determine the relative displacements and/or rotations of nodes in response to applied loads.

The manner with which an individual holds a smartcard places constraints upon the freedom of movement of the card. As indicated, it is envisaged that a smartcard's owner will support the card with one hand, whilst presenting the characteristics of the other for verification. Common experience suggests that the most practical manner of performing this action is for the supporting hand to grasp the longitudinal edges of the card, thereby restricting the out-of-plane movement of the edges. The extent to which the hand contacts with the card, and the extent to which the edges are supported is far from clear. Hence some assumptions are made in this regard, and are detailed in Section 4.4.2.

These assumptions are superimposed with those made earlier in section 4.3. Nevertheless, it is believed that this work provides a first order approximation of the behavior a card-sensor system, illustrating rather than specifying its characteristics.

4.4.1 Mesh Characteristics

Ideally all problems would be solved using solid cuboidal volume elements, however, in order to maintain a reasonable aspect ratio on the dimensions of each element, meshing a smartcard using solid elements requires that the length and breadth of each element be very small. This unacceptably increases the number of nodes and elements involved. Hence, the smartcard is modelled in this work with rectangular shell elements, generally considered to be appropriate for thin planar geometries in which thickness is small in comparison with the other dimensions (Fagan 1992).

The physical properties of smartcards are outlined in the ISO7816-1 standard[3] This document prescribes that smartcards adhere to the dimensions given in Table 4.3.

To remove the uncertainty in the ISO standard thickness, the thickness of six

[3]ISO7816-1:1998 Identification cards – Integrated circuit(s) cards with contacts – Part 1: Physical characteristics. International Organization for Standardization, Geneva, Switzerland. Web: `http://www.iso.ch`

Measurement	ISO Specification (mm)
Length	85.6
Width	54.0
Thickness	0.76 ± 0.08

Table 4.3: ISO Specifications for the Physical Dimensions of a Smartcard

cards from the same batch was measured. This was performed using a screw-gauge micrometer, and the mean thickness was found to be 0.815mm with a standard deviation of 0.005mm. This value is used as the element thickness.

The length and width of the smartcard's geometry was subdivided (or *discretised*) into a mesh of 17 × 11 elements, with each element having dimensions of 5.035mm length and 4.909mm width. These mesh characteristics were chosen to fit within the nodal restrictions of the analysis software[4], whilst still providing reasonable spatial resolution.

4.4.2 Constraints Conditions

With a smartcard held in hand, a user restricts the vertical displacement of the two longitudinal edges. The hand in contact with the smartcard may impart, or inhibit rotation of the edges, and compression may be applied across the width of the card. Furthermore, these constraints are non-uniform and asymmetrical, and are likely to vary between individuals. Complicating the situation further, real constraints are likely to be dynamic, and will change as an individual presents their characteristics for verification. In reality, the way in which a user interacts with a smartcard is complex, and this is a difficult modelling problem.

Without extensive analysis of real constraints imposed by a number of users holding cards in their hands, a quantitative assessment of constraints in beyond reach. For this reason it was decided to apply only simple constraints with which one may have some confidence. Hence, this FE model applies only vertical displacement constraints to the edge nodes of the card's longitudinal edges. This means that the edge nodes are free to move in both planar directions, and are able to rotate without boundary restrictions. Whilst this imposes yet further approximations upon

[4]Using AutoFEA v7.0, from AutoFEA Engineering Software Inc. CA, USA. Web: `http://www.autofea.com`

the model, this approach facilitates the experimental verification of its results (as is described in section 4.6).

4.4.3 Loading Conditions

As already stated, the manner of interaction between an individual and a smartcard sensor system is limited, but may be considered to include a finger touching, pressing and tapping upon the card. Whilst the absolute characteristics of this interaction are dependent upon the properties of the sensing mechanism, and hence cannot be determined at this stage, it was deemed important to apply loads similar in magnitude to those which might be expected during the presentation of real discriminatory characteristics. At the disposal of this work was a calibrated (ceramic) piezoelectric force sensor, but piezoelectric sensors tend to exhibit poor static or quasi-static responses, thereby eliminating the capture of a finger touching or pressing. Hence the force magnitudes of fingers tapping on the sensor were recorded.

To estimate the force magnitude with which a user taps, five volunteers were each asked to tap upon a calibrated piezoelectric pressure sensor printed on alumina. Their instruction was to tap the sensor with as little or as much force as desired, randomly varying the strength of their taps as the session progressed. Each participant provided between 250 and 400 taps. A Visual Basic program was written to extract the peak force values from the captured data. Figure 4.5 shows the force distribution of the taps collected.

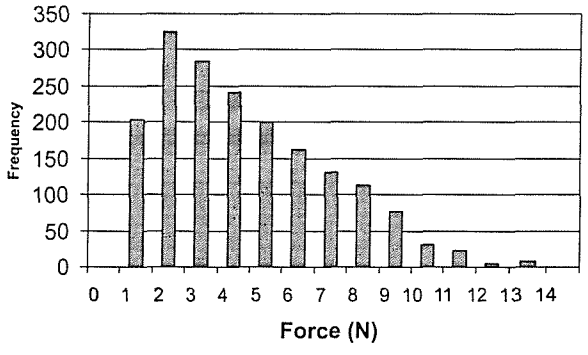


Figure 4.5: Peak Tapping Forces

Figure 4.5 does not exhibit normal distribution characteristics, and visually appears more akin to a log-normal distribution. This is reminiscent of the well documented phenomena of logarithmic sensory perception in human auditory and visual systems[5] (Stevens 1960), although no claim is being made to this effect, and further research is necessary to justify such a claim.

From the data collected, the modal tapping magnitude was found to be around 2.5N. It is assumed that a reasonable estimate of fingertip area is $\sim$1cm^2, and hence a static 2.5N load is distributed over 1cm^2 and used as the input to our model. In practice this means distributing the force over 9 nodes. Therefore, 0.2778N was applied to each of the 9 nodes.

Four load cases were considered, each being applied along the central longitudinal axis of the card, and whose centre-nodes are 1, 2.5, 4 and 5cm from the short edge, furthest from the processor contacts. Hence, 5.5cm of longitudinal distance is investigated, and believed to cross most of the region available to an on-card sensor system. Figure 4.6 shows the smartcard model with edge constraints and loading conditions, superimposed upon the region covered by the piezoresistive sensor. The piezoelectric sensor is not shown for reasons of clarity, however, its footprint covers the lower (edgewards) eight nodes of the piezoresistive sensor.

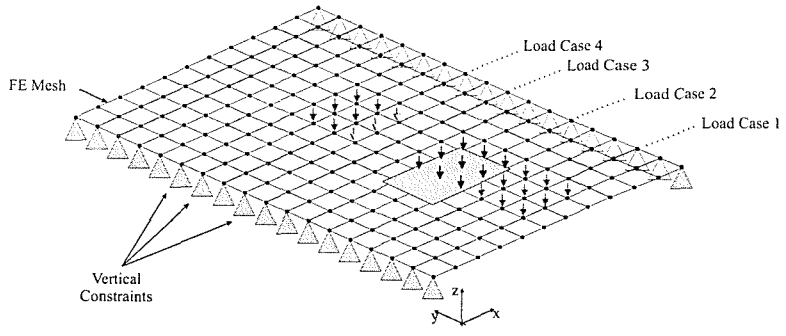


Figure 4.6: Mesh Position of Load Cases: Each of the four load cases is applied over nine nodes, each covering different 1cm^2 regions. The coordinates of each central node are given in parenthesis.

[5]Known as the *Weber-Fechner* law.

4.5 Finite Element Results

The FEA results are used to calculate the mean planar strains of the smartcard's surface corresponding to the sensor positions of Figures 4.1 and 4.2. These mean strains were then used with equations (3.24), (3.26) and (3.23) to compute the theoretical piezoresistive sensor responses, and with equations (3.33), (3.34) and (3.30) to compute the theoretical piezoelectric sensor responses.

Section 4.5.1 presents the displacement and stress maps resulting from this analysis.

4.5.1 Displacement and Stress Maps

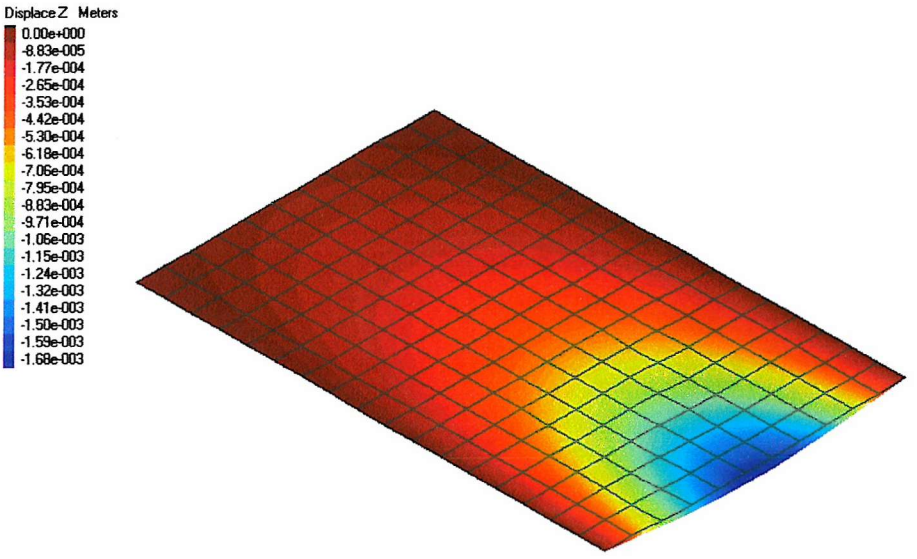
Figures 4.7–4.10 provide FEA results for load cases 1 through 4, as specified by Figure 4.6, respectively. Out-of-plane displacement maps are given in sub-figures (a), whilst sub-figures (b) and (c) show the planar X and Y direction stresses, respectively.

It is clear from Figures 4.7 through 4.10 that a load, which can be considered typical of those applied during the presentation of finger characteristics, causes a smartcard to flex. Table 4.4 shows the maximum deflection resulting from each load case.

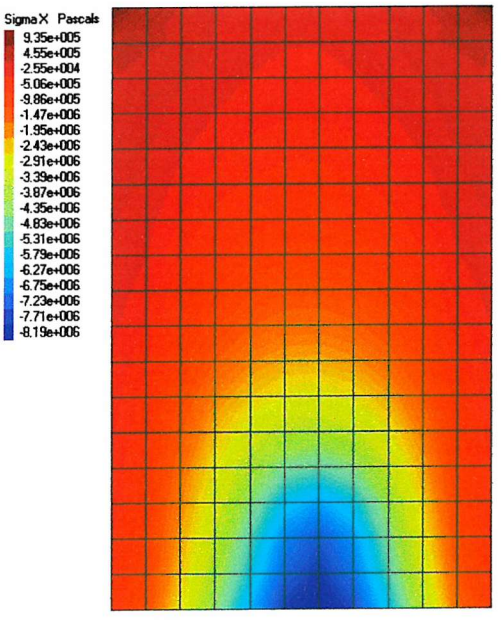
Load Case	Maximum Out-of-plane Deflection (mm)
1	-1.68
2	-0.85
3	-0.72
4	-0.74

Table 4.4: Maximum Out-of-Plane Deflections

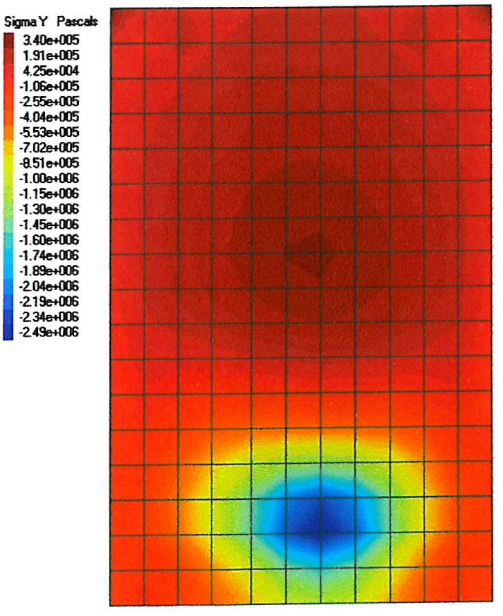
The extent to which the card deforms depends upon the loading position, as is evident from Table 4.4. Load case 1, for example, causes the greatest vertical deflection, whilst load case 3 causes the least. This is due to the material of the card constraining deflections: Case 1 is applied close to an unconstrained edge which is more able to deform than case 3 which is applied towards the centre of the card.



(a) Vertical Deformation

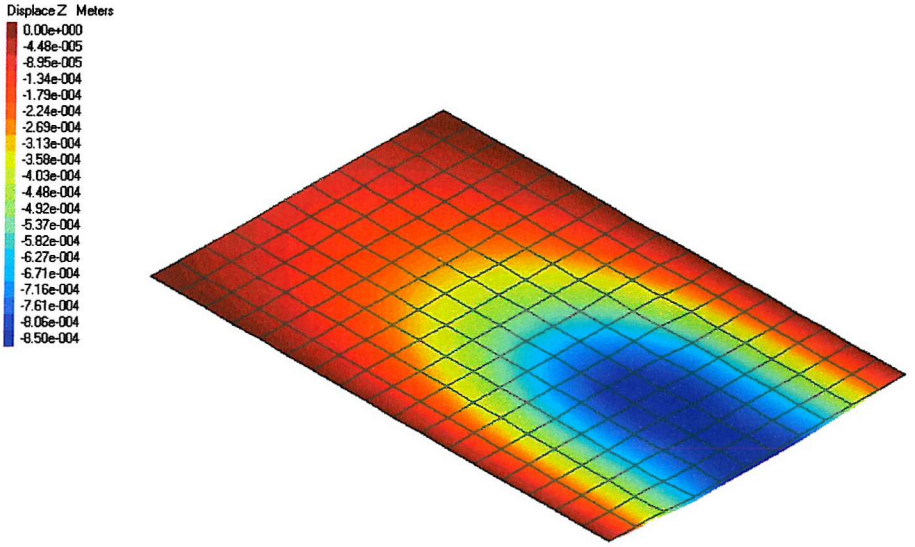


(b) X-Direction Stress

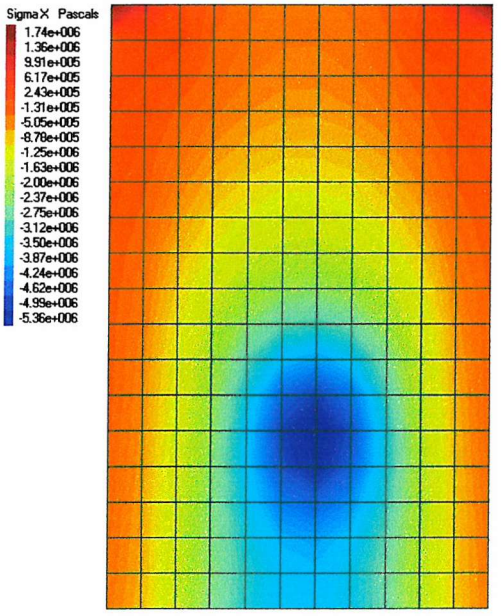


(c) Y-Direction Stress

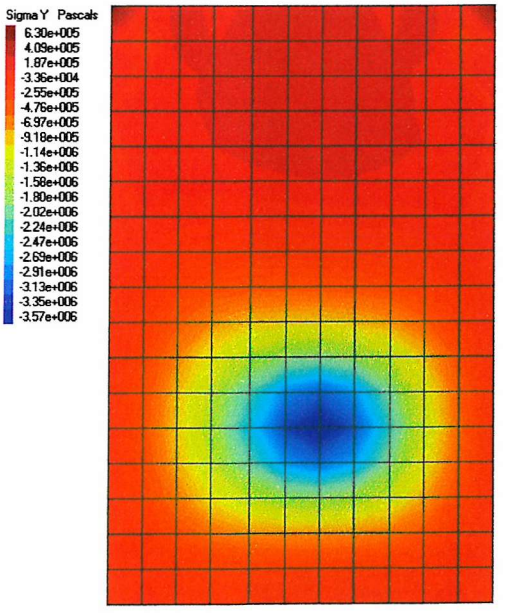
Figure 4.7: Load Case 1- Simulation Results



(a) Vertical Deformation

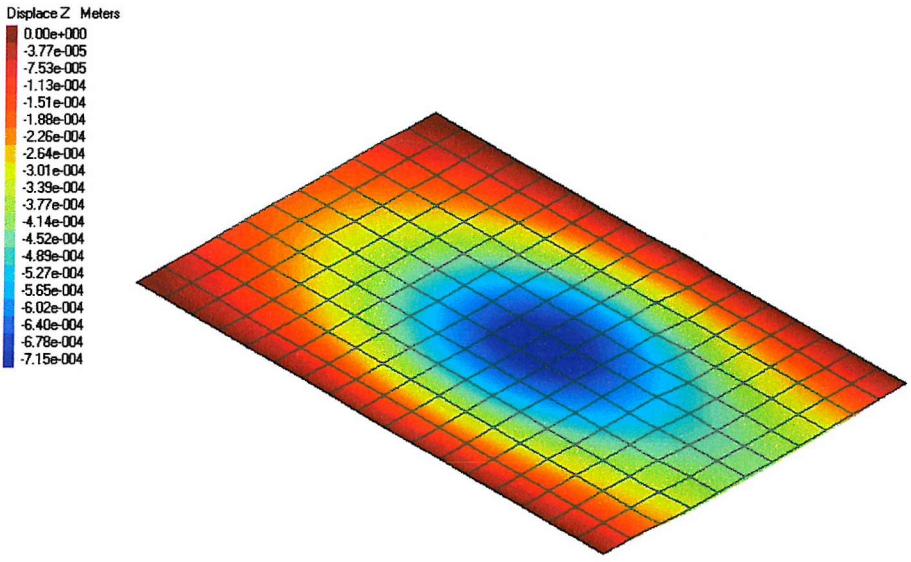


(b) X-Direction Stress

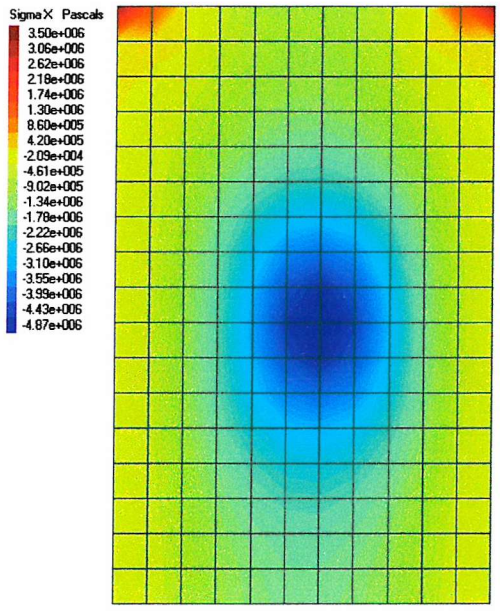


(c) Y-Direction Stress

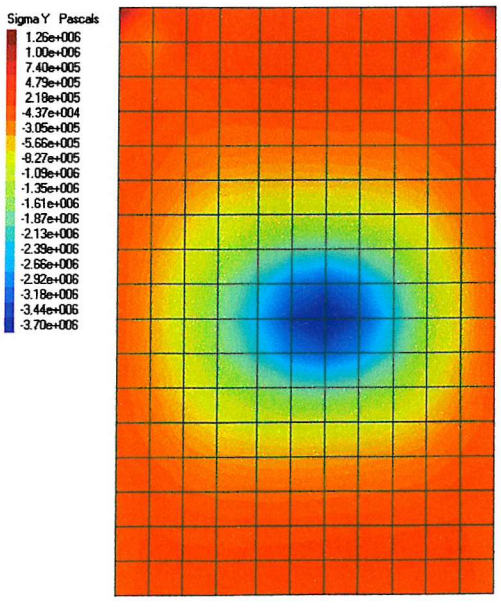
Figure 4.8: Load Case 2- Simulation Results



(a) Vertical Deformation

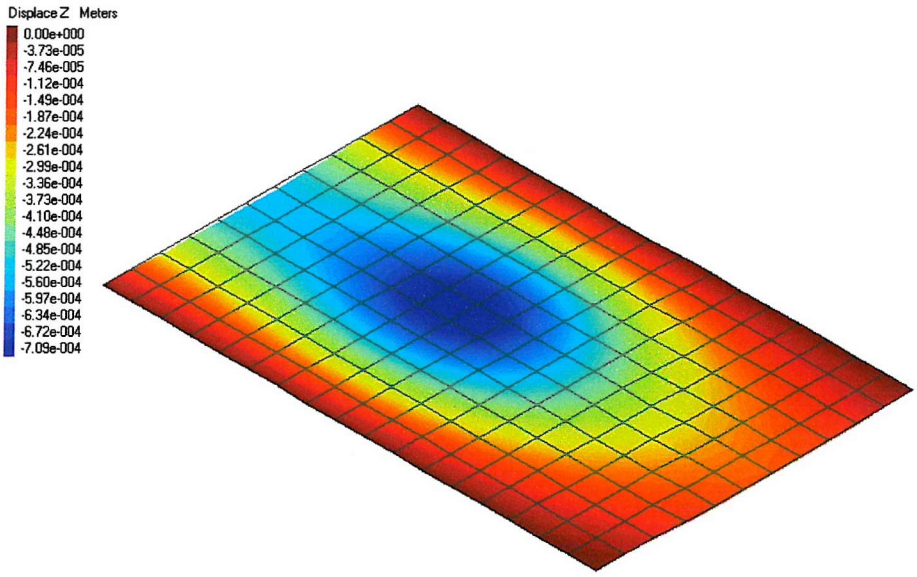


(b) X-Direction Stress

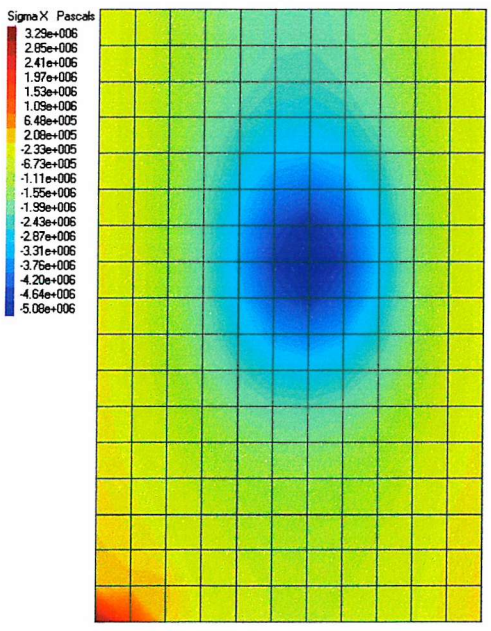


(c) Y-Direction Stress

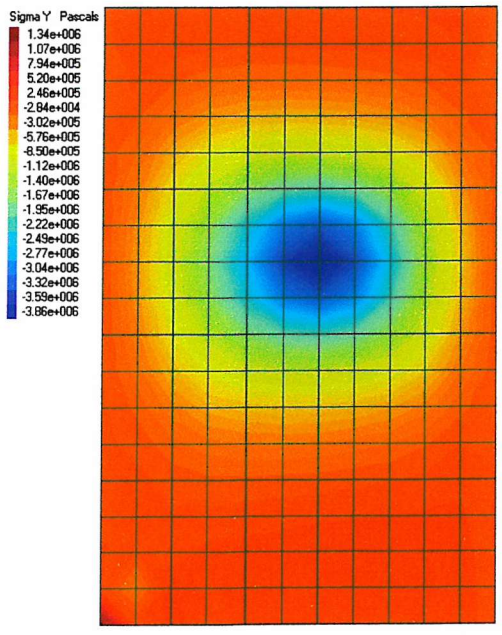
Figure 4.9: Load Case 3- Simulation Results



(a) Vertical Deformation



(b) X-Direction Stress



(c) Y-Direction Stress

Figure 4.10: Load Case 4- Simulation Results

But both sensor models require the planar strains of the surface in order to calculate their theoretical response to loading. Strains can be derived from the planar stresses of the card, and this is performed in the next section.

4.5.2 Strains Experienced by Sensors

The assumption that sensors bonded to a smartcard will conform to the planar strains of the card's surface is made in Section 3.5 and continued throughout this analysis. The FEA results are used to calculate the mean stresses of the smartcard in the regions corresponding to both sensors (as shown in Figures 4.1 and 4.2, and as specified in Figure 4.6), from which the mean surface strains of these regions are calculated. Table 4.5 presents the mean internal smartcard stresses corresponding to the position of both the piezoresistive and piezoelectric sensors.

Load Case	Piezoresistive Footprint Mean Stress (MPa)		Piezoelectric Footprint Mean Stress (MPa)	
	$\overline{\sigma}_{x(c)}$	$\overline{\sigma}_{y(c)}$	$\overline{\sigma}_{x(c)}$	$\overline{\sigma}_{y(c)}$
1	-3.86	-0.761	-4.29	-1.01
2	-4.84	-3.15	-4.95	-3.21
3	-3.10	-1.34	-2.79	-0.947
4	-2.47	-0.767	-2.23	-0.531

Table 4.5: Internal Smartcard Stresses Corresponding to Both Sensor Regions

Mean surface strains of the smartcard, corresponding to the regions under both sensors, are found by solving equations (3.12) and (3.13) for the planar smartcard strains, $\varepsilon_{x(c)}$ and $\varepsilon_{y(c)}$, with the smartcard material properties (ie. $\nu = \nu_c$ and $E = E_c$) and the mean stresses of Table 4.5. These are presented in columns 2&3 and 4&5 of Table 4.6, for the piesoresistive and piezoelectric sensors respectively.

Load Case	Piezoresistive Footprint		Piezoelectric Footprint	
	$\overline{\varepsilon}_{x(c)}$	$\overline{\varepsilon}_{y(c)}$	$\overline{\varepsilon}_{x(c)}$	$\overline{\varepsilon}_{y(c)}$
1	-1.15×10^{-3}	2.28×10^{-4}	-1.26×10^{-3}	2.00×10^{-4}
2	-1.18×10^{-3}	-4.23×10^{-4}	-1.20×10^{-3}	-4.29×10^{-4}
3	-8.36×10^{-4}	-5.23×10^{-5}	-7.84×10^{-4}	3.65×10^{-5}
4	-7.03×10^{-4}	5.54×10^{-5}	-6.54×10^{-4}	1.02×10^{-4}

Table 4.6: Mean Surface Strains Corresponding to Both Sensor Regions

4.5.3 Theoretical Sensor Response

The piezoresistive sensor responses to each load case are found with equation (3.23) using the strains presented in Table 4.6 and the normal out-of-plane strain, as given by (3.26). The material properties required by (3.26) and (3.23) are as stated in Section 4.3. The out-of-plane normal stress, $\sigma_{z(f)}$ acting on the sensor is non-zero for load case 2, only, and is assumed to act uniformly upon the sensor, and $\sigma_{z(f)}$ =1.67MPa is used. Table 4.7 gives the results.

Load Case	dR/R (%)
1	-0.527
2	-0.756
3	-0.452
4	-0.348

Table 4.7: Theoretical Piezoresistive Response

Similarly, the strains of Table 4.6 are used to compute the theoretical piezoelectric sensor responses. Equations (3.33) and (3.34) are used to calculate the planar stresses acting on the sensor due to the smartcard, which are then used with equation (3.30) to give the charge response of the sensor. The sensor's area is taken to be 0.64cm^2, and its material properties are specified in Section 4.3. The out-of-plane normal stress is applied only during load case 2, and is taken to be 3.9MPa. Table 4.8 provides the results.

Load Case	Charge Generated (nC)
1	-11.9
2	-18.4
3	-8.420
4	-6.21

Table 4.8: Theoretical Piezoelectric Response

Although the largest sensor responses occur during load case 2 (as one would expect), this analysis indicates that both sensor types respond to loads applied elsewhere on the card. Taking for example the sensor responses due to load case 4, in which the centre of the load is applied 1.5cm from the sensor, the theoretical sensor responses are 46% and 34% of the response due to direct loading of the sensors. The next section aims to experimentally verify these results, which will then be discussed in greater depth.

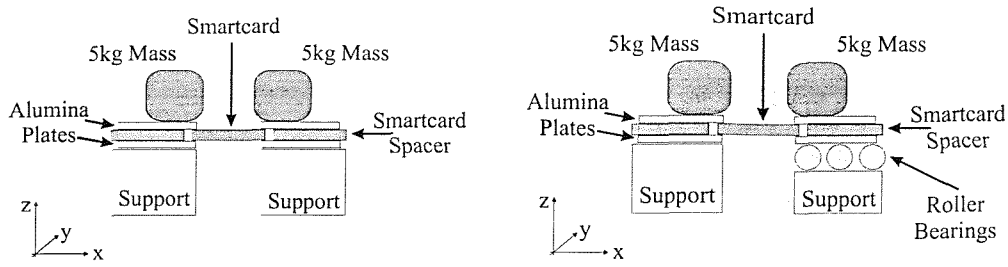
4.6 Experimental Verification

In order to verify the behaviour of the smartcard FE model, one must try to replicate exactly the conditions set out in the model. This is a difficult thing to do as the finite element model is an idealised perspective of the real-world. For example the boundary conditions used to restrict vertical movement are applied precisely on the longitudinal edges of the model smartcard. They constrain the model by fixing these edges, disallowing their movement in the z-direction, whilst allowing complete freedom of movement in the planar x and y directions. Whilst one can imagine complex mechanical structures, involving razor-edged clamps, vertically constraining the outermost region of both longitudinal edges, and setting these clamps upon low-friction bearings allowing unconstrained planar movement, such apparatus was unavailable to this experiment. Furthermore, the loading conditions of the FE model implies a perfectly uniform load distribution over the load's contact area with the card. In reality a rigid mass applied to a smartcard will cause the card to flex. As the card flexes, the load distribution will concentrate around the edges of the mass.

Nonetheless, this section describes an attempt to replicate, as closely as possible, the characteristics of the finite element model.

4.6.1 Constraints

A number of constraining methods were considered for this experiment. Firstly a method of rigidly clamping both longitudinal edges was investigated. This is shown schematically in Figure 4.11(a). In this approach approximately $\frac{1}{2}$mm of both longitudinal edges are sandwiched between two alumina plates. The alumina plates provided flat, uniform surfaces with which the edges could be vertically constrained. Two 5kg masses provided the vertical constraining forces, and were applied onto each of the edge plates. Clearly this method allows little scope for planar movement of the card. In an attempt to improve the freedom of movement in the planar directions, the scheme presented in Figure 4.11(b) was considered. In this scheme, both longitudinal edges were sandwiched between two alumina plates in the manner described above. One of the edges is rigidly constrained with the application of a 5kg mass, whilst the other is offered some freedom of movement by



(a) Rigid Constraints (b) Roller Constraints

Figure 4.11: Possible Realisation of Constraint Boundary Conditions

placing rollers underneath the constraint's bottom plate. The rolling edge was also constrained vertically by means of a 5kg mass. Visual inspection of this method confirmed that the application of a load onto the card caused the constraining mass to move just perceptively inwards. Whilst this method offered improved freedom of movement in the x-direction, it was found to be limited both by the inertia of the constraining mass and friction of the rollers.

The preferred method of constraining the edges is shown schematically in Figure 4.12. In this method the outermost $\frac{1}{5}$ mm of the card's longitudinal edges are simply supported by means of flat plastic blocks. By simply supporting the extreme edge of the card, the downwards force of the card's weight and that of the load is transferred to the supports through the edge of the card. This means that the edges are unable to rise, and are in effect constrained by the weight of the card and by the force of the load itself. The card is loosely attached to the supports using a loop of adhesive tape, helping to maintain the card's position on the supports, whilst providing some structural connection between the card and both supports. An additional advantage of loose bonding is that some freedom of movement in the x, and y directions is exhibited.

4.6.2 Mechanism for Applying Loads to the Card

An efficient mechanism for applying loads to the card is to use a simple pulley system, as outlined in Figure 4.12. The load is suspended just above the smartcard's surface by means of thin 200μm diameter wire, and released in a controlled manner allowing the load to rest on the smartcard. The contact area between load

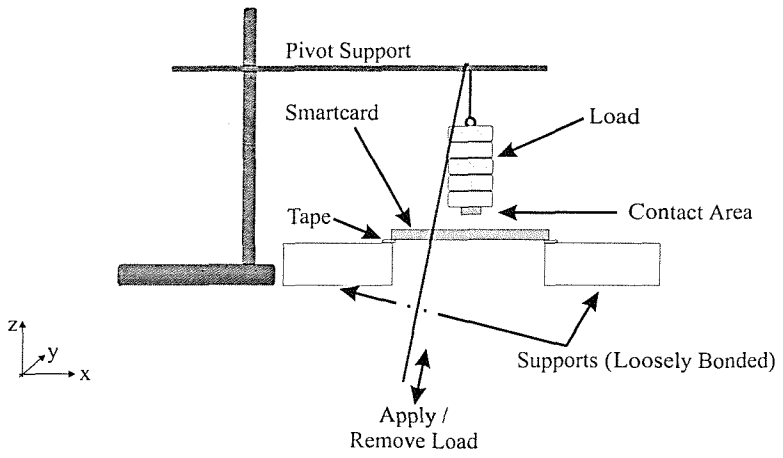


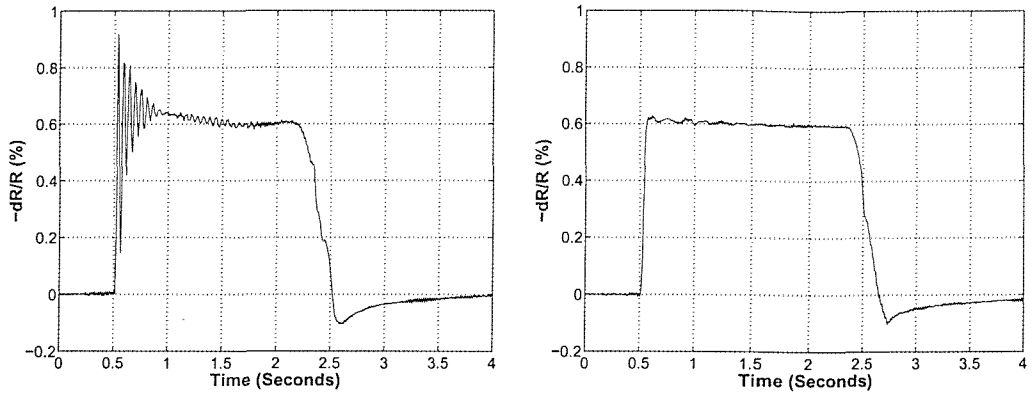
Figure 4.12: Mechanism for Applying Loads

and smartcard is determined using a square of smartcard material, cut to an area of $1cm^2$, and bonded to the bottom surface of the load. Figure 4.13(a) shows the the response of the piezoresistive sensor to a load applied onto the card in this manner.

Applying the load directly onto the card can induce a number of problems. From Figure 4.13(a), the sudden application of a load can be seen to cause mechanical oscillations of the card. Although energy is dissipated, and the card quickly assumes its steady state, it is desirable to reduce these oscillations as much as is possible. A further concern with this method of loading is that loads have a tendency to swing as they are removed from the card. This can result in the load being applied to slightly different surface positions each time it is placed in contact with the card.

A solution to the above is to attach a small ($1cm^2$ area with 5mm thickness), high density foam rubber pad to the position of interest on the card's surface. This has the effect of damping oscillations, whilst also reducing the positional variance of the contact point. Figure 4.13(b) shows the sensor's response to a load applied in this manner. The mechanical oscillations have been reduced, and the steady-state output of the sensor is of comparable magnitude to the response of the direct contact approach.

Rather than applying load to the piezoelectric sensor, it is good practice to remove loads thereby avoiding dynamic loading. Piezoelectric sensors suffer from charge



(a) Load Applied Directly Onto Card

(b) Load Applied Onto Bonded Foam Pad

Figure 4.13: Piezoresistive Sensor's Response to Applied Loads

leakage across sensor contacts, and leakage arising from the conditioning electronics (Appendix B). Therefore load response measurements must be taken immediately after the load has been applied. As a result, it is good practice to remove, rather than apply, loads to avoid dynamic loading. The change of sense is accounted for in presented data.

To be consistent with the finite element model, a vertically downwards force of 2.5N must be applied to the card. Newton's third law states

$$F = m\, a_{gravity} \tag{4.6}$$

where F is the downwards force due to the gravitational acceleration, $a_{gravity}$, of mass m. The required mass is calculated as 255.1g, taking $a_{gravity}$ to be 9.81ms^{-2} (Halliday & Resnick 1989). A combination of small standard masses provided a total measured mass of 254.3g, resulting in a downwards force of 2.49N.

4.6.3 Signal Conditioning and Data Acquisition

Piezoresistive responses are conditioned using a resistive bridge circuit, whose output is amplified with an AD622 instrumentation amplifier from Analog De-

vices.[6]. Piezoelectric signal conditioning, on the other hand, was performed using the LMC6001 Ultra-Low Input Current Amplifier from National Semiconductor[7], configured as an op-amp integrator. Appendix B provides schematics and further descriptions of the signal conditioning circuits used.

Data acquisition from both sensors was performed using a Data Translation[8] DT9802 acquisition board, sampling with 12-bit resolution (between -5 to 5V), at a rate of 5kHz.

4.6.4 Experimental Results

For both sensor cards and each load case, the load was applied (removed in the piezoelectric case) to the card at least 30 times. Tables 4.9 and 4.10 show the experimental means (and standard deviations) for the piezoresistive and piezoelectric sensors, respectively. The measured responses are also quoted in proportion to the theoretical predictions. Figures 4.14(a) and 4.14(b) show typical responses to all load cases for the piezoresistive and piezoelectric sensors, respectively.

	Piezoresistive Response $dR\ /\ R$ (%)	
Load Case	Experimental (Std. Dev.)	Expt. / Theoretical
1	-0.457 (0.036)	86.7%
2	-0.582 (0.022)	77.0%
3	-0.388 (0.017)	85.8%
4	-0.293 (0.034)	84.2%

Table 4.9: Experimental Piezoresistive Response to Known Loads

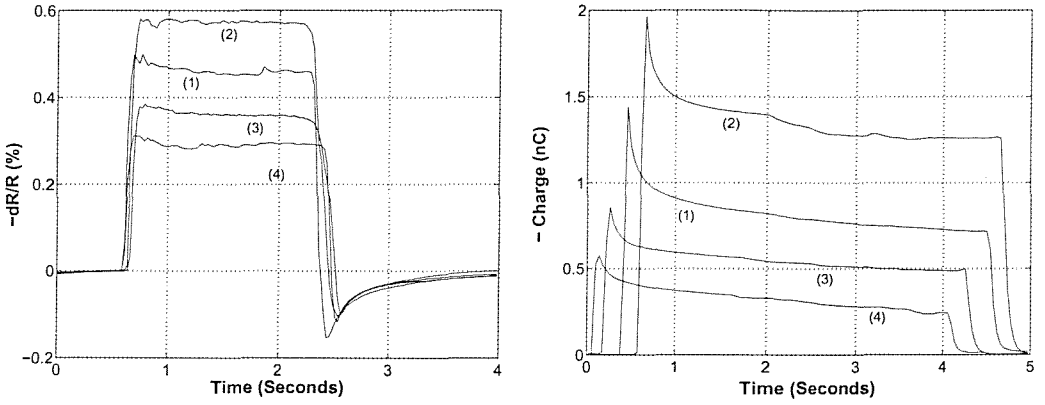
Both sets of experimental results show clearly the effects of card flexing. Again using load case 4 for illustration, it is observed that the sensor responses are 50% and 38% of the direct sensor loading case, for piezoresistive and piezoelectric responses respectively.

Whilst the experimental results for the piezoresistive response compare well with theory, and load cases 1, 3 and 4 are within approximately 15% of theoretical val-

[6]Analog Devices, Norwood, MA, USA. Web: `http://www.analog.com`
[7]National Semiconductor, Santa Clara, CA, USA. Web: `http://www.national.com`
[8]Data Translation, Inc. Marlboro, MA, USA. Web: http://www.datx.com

	Piezoelectric Response Charge Out (nC)	
Load Case	Experimental (Std. Dev.)	Expt. / Theoretical
1	-1.53 (0.22)	12.9%
2	-2.10 (0.22)	11.4%
3	-1.06 (0.10)	12.6%
4	-0.808 (0.07)	13.0%

Table 4.10: Experimental Piezoelectric Response to Known Loads



(a) Piezoresistive Response

(b) Piezoelectric Response

Figure 4.14: Response to Known Loads: Load Case is indicated in parenthesis.

ues, there are a number of possible explanations for these discrepancies. Firstly, whilst every effort was made to experimentally reproduce the conditions of the finite element model, it is likely that differences exist. In particular, the constraining conditions of the experimental method are not precise representations of the finite element model. Frictional forces between the smartcard and its supports will restrict freedom of movement in the planar directions thereby inhibiting planar strains, and hence sensor responses.

Further, some material properties have been estimated rather than measured and can have a significant influence upon theoretical responses. For example, 0.4 was taken as the value for Poisson's ratio of the piezoresistive sensor, and whilst this can be considered a reasonable assumption, investigation of the sensor model demonstrates particular sensitivity to this property. For example taking Poisson's ratio of the film to be 0.35, rather than 0.4, results in a predicted resistance change of

-0.641%, whilst a value of 0.45 results in a resistance change of -0.392%, for load case 1. Indeed consideration (and rearrangement) of equation (3.23) shows that

$$\frac{dR}{R} = (G+1)\varepsilon_{x(c)} + (G-1)\varepsilon_{y(c)} + (G-1)\varepsilon_{z(f)} \tag{4.7}$$

and since the planar strains, $\varepsilon_{x(c)}$ and $\varepsilon_{y(c)}$, depend upon the strains developed within the smartcard, and G is a material property of the piezoresistive film, it follows that

$$\frac{dR}{R} \propto \varepsilon_{z(f)}. \tag{4.8}$$

From equation (4.4) the normal strain, $\varepsilon_{z(f)}$, is a function of Poisson's ratio of the film, and the planar strains, $\varepsilon_{x(c)}$ and $\varepsilon_{y(c)}$. For a fixed static load these strains are determined by the smartcard's mechanical response and shall be considered constant. That is

$$\varepsilon_{x(c)} + \varepsilon_{y(c)} = k \tag{4.9}$$

Hence, from (4.4)

$$\varepsilon_{z(f)} = f(\nu_f) = \frac{-\nu_f}{(1-\nu_f)}\, k. \tag{4.10}$$

Further investigation of (4.10) reveals $\varepsilon_{z(f)}$ to be a monotonically increasing function of ν_f within the permissable bounds of $-1 \leqq \nu_f \leqq 0.5$.

In addition, the piezoresistivity coefficient, G, was not measured directly, rather G was inferred from related measurements reported in the literature (Arshak et al. 1995). The predicted piezoresistive responses of table 4.7 were generated using Arshak's median measurement of gauge factor ($GF_L = 4.5$), from which the piezoresistivity coefficient was inferred. However, if his lower gauge factor measurement is used ($GF_L = 4$) then a value of $G = 8.69$ is calculated. With this value of G, the predicted sensor responses tend towards the experimental values, with measured responses being 97.4%, 90.2%, 98.2% and 95.4% of the respective theoretical value for each load case.

It must be stated that this discussion attempts only to illustrate the model's sensitivity to certain parameters, and makes no claim that their values can be determined from this work.

On the other hand, experimental and theoretical magnitudes for the piezoelectric responses deviate significantly, with experimental measurements being only around

12.5% of their corresponding theoretical values. The large discrepancy is believed to stem from the estimation of Young's modulus of the piezoelectric composite. The rule of mixtures method, (4.1), was used and assumes that the two materials in the composite are exclusively and homogeneously mixed. This is most likely not the case, as the presence of air gaps within PZT-phenolic films have been observed (Papakostas 2001). Air gaps within the composite will significantly reduce Young's modulus, and although further determination is beyond the scope of this work, reducing Young's modulus of the PZT film has been shown to bring theoretical results closer to experimental. This is clearly demonstrated by considering equations (3.33), (3.34) and (3.30). Combining and rearranging shows

$$Q \propto E_{pE}. \tag{4.11}$$

As an example, and with the same caveat as above, taking E_{pE} = 5.5GPa, causes the theoretical responses to descrease, such that measured responses become 101.7%, 90.9%, 99.9%, and 103.1% of theoretical values.

The results presented in column 2 of Tables 4.9 and 4.10, show that the direct loading responses of both sensors (load case 2), are significantly lower in proportion to their theoretical values than is the case for the other loading positions. This is believed to be due to the sensor film and its polymer substrate dispersing the applied load over an increased area. For example, applying the same 2.5N load over an area of 2cm^2, rather than 1cm^2, and re-running the FEA simulation, lower surface strains are generated. Repeating the approach of Section 4.5.3, the piezoresistive response was calculated to be $\frac{dR}{R}$ = -0.649% and the charge generated by the piezoelectric sensor, Q = -16.2nC. The experimental measurements in comparison to these new theoretical results are 89.6% and 12.9%, for the piezoresistive and piezoelectric sensors respectively, and are comparable to the results of the other load cases. It should be noted that this effect will have no influence if the sensor and its substrate are smaller than the area over which a load is applied.

4.7 Spatial Characteristics

The above analysis demonstrates that applying a load to a smartcard causes flexing and strain propagation, which in turn results in sensors responding to loads

applied elsewhere on the card. This is an unfortunate consequence and does not encourage the design of pressure sensors for the capture of discriminating spatial characteristics. However, the above considers only the response of fixed sensors to loads applied at different positions. Making use of the smartcard model we can investigate the converse in which sensors at locations across the card respond to a load applied at one fixed position.

Before doing so the discrepancies between model and experiment must be accounted for. As the preceding work shows, both sensor models are sensitive to changes in the values of their material properties, and a judicious choice of property value can have a significant effect on the model's response. Whilst there exists some uncertainty about actual property values, these cannot be inferred by comparing theoretical with experimental results. The reason for this is simply that the model's response is a consequence of the interaction of all components, and changing component properties simultaneously can result in the same ultimate response. For example, the piezoresistive response to load case 1 was calculated to be $\frac{dR}{R} = -0.527\%$ with $\nu_{pR} = 0.4$ and $G = 10.62$. The same ultimate sensor response can be achieved with $\nu_{pR} = 0.45$ and $G = 18.6$. A more appropriate method of accounting for experimental differences is to scale the model's response to the experimental observations. By taking the mean of experimental to theoretical results, scale factors of 0.86 and 0.13 are found for the piezoresistive and piezoelectric sensors respectively.

4.7.1 Array Sensor Response

The finite element model can be used to simulate the response of an array of sensors bonded to a smartcard. The following analysis considers a grid of 4×4mm sensing elements, each centered on one node of the FE mesh, as shown in Figure 4.15.

In order to assess the extent to which individual sensing elements respond to a single load, the smartcard model is loaded with a 2.5N force applied over an area of 1cm^2. This is the loading condition of case 2, and is believed to be typical in magnitude of forces applied during the presentation of discriminatory finger characteristics. The resulting nodal stresses are used to calculate sensor responses

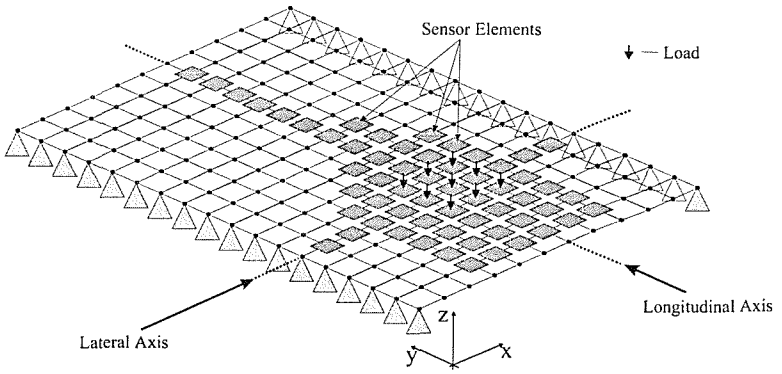
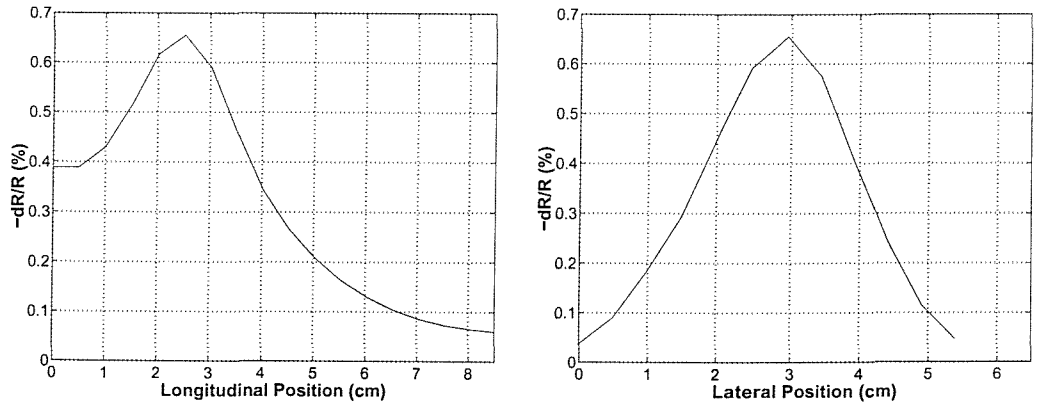


Figure 4.15: Simulated Sensor Array: Sensing elements are centered on FE nodes, and the array is only partially completed for clarity.

in the manner prescribed above (section 4.5.3).

Rather than presenting results for all possible sensing elements, this analysis considers two perpendicular cross-sectional responses, through the longitudinal and lateral axes of the card, and intersecting with the load center. This situation is shown in Figure 4.15 and is believed to sufficiently demonstrate the response characteristics of a sensor array.

Considering firstly an array comprising of piezoresistive elements, whose current flow is in the lateral, or x, direction of the card. The response characteristics of these elements is shown in Figures 4.16(a) and 4.19(b) for longitudinal and lateral cross-sections, respectively.



(a) Longitudinal Array Cross-Section (b) Lateral Array Cross-Section

Figure 4.16: Piezoresistive Sensor Element Response with Position

As reasonably expected, the peak sensor response corresponds to the centre of the applied load, and the response of sensing elements decreases with distance from this position. For example, sensing elements 1.5cm either side of the load in the longitudinal direction exhibit approximately 65% and 53% of the peak response, for edgewards and centerwards elements respectively. This is significant, given that both elements are a full 1cm from the edge of the applied load. The cross-sectional responses through the lateral axis are seen to decrease more rapidly with distance, but remain greater than 30% of the peak response for the 3.5cm central section of card.

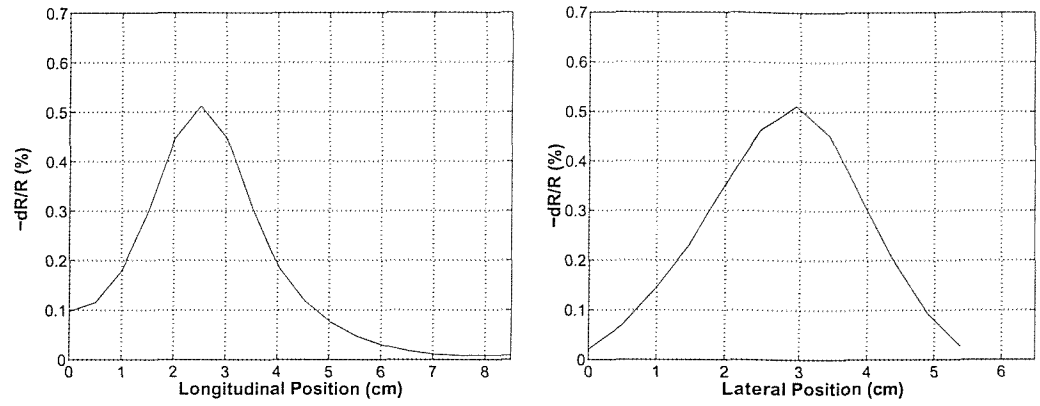
Considering the piezoresistive sensor description given in equation (4.7), it can be seen that piezoresistive response is more sensitive to strains in the direction of current flow, $\varepsilon_{x(c)}$ in this case, than in the other orthogonal directions. From experience, and from the displacement maps presented in Figures 4.7(a)–4.10(a), it is observed that a smartcard will flex along its longitudinal axis, if supported in a manner typical of being held in hand. As a result, strains are higher in the lateral orientation and this is indicated by the stress maps of Figures 4.7(b)&4.7(c) – 4.10(b)&4.10(c). On the other hand, longitudinal, or y orientation, strains are more confined to the region surrounding the load, as indicated by Figures 4.7(c)–4.10(c).

These properties can be exploited by configuring the sensing elements, such that current flow is parallel to the longitudinal axis of the card. In this instance, the piezoresistive response is given by

$$\frac{dR}{R} = G\left(\varepsilon_{y(c)} + \varepsilon_{x(c)} + \varepsilon_{z(f)}\right) + \varepsilon_{y(c)} - \varepsilon_{x(c)} - \varepsilon_{z(f)} \tag{4.12}$$

and is more insulated from high lateral strains. Figures 4.17(a) and 4.17(b) show the longitudinal and lateral cross-section responses, respectively.

The immediate difference between sensor responses is a reduction in peak response, from approximately 0.66% for piezoresistive elements whose current flow is in the lateral direction, to 0.51% for these rotated sensor elements. This, as explained, is due to sensor elements being insulated from the higher lateral strains of the card. The array exhibits a sharper longitudinal cross-sectional response to applied load, with the response of elements being approximately 30% of the peak, at a distance of 1.5cm from the load center. Due to the confined circular region of high stress in



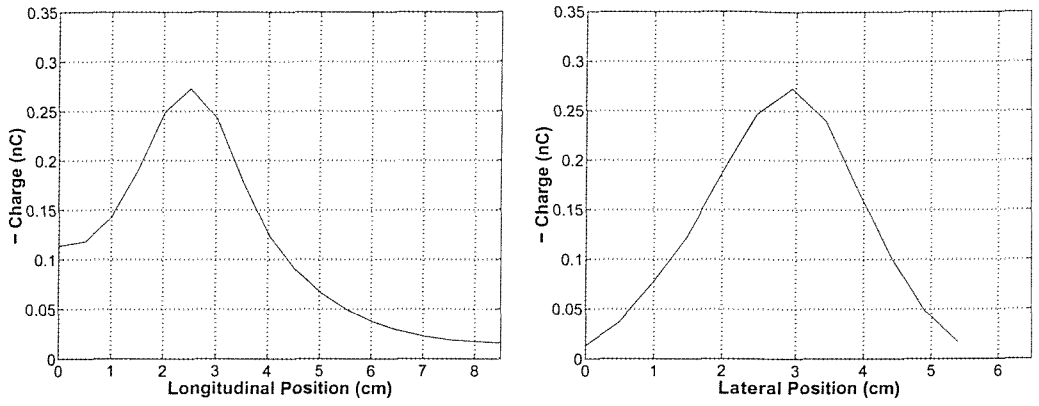
(a) Longitudinal Cross-Section (b) Lateral Cross-Section

Figure 4.17: Piezoresistive Sensor Element (Rotated through 90°) – Response with Position

the longitudinal direction, the lateral cross-section responses are similar to those presented in Figure 4.16(a).

Figures 4.18(a) and 4.18(b) present the results for an array of piezoelectric sensing elements. As can be seen, the peak amplitude of approximately -0.27nC is significantly smaller than than the experimental measurements, presented in Section 4.6.4. Equation (3.30) clearly shows the piezoelectric activity to be proportional to the area of a sensor, which in this case is 0.16cm^2 (0.4×0.4cm^2) as compared with 0.64cm^2 for the experimental sensor. Cross-sectional responses appear similar to those of the piezoresistive array, and since the piezoelectric model does not distinguish between planar directions, rotating the sensors relative to the card offers no advantage.

In conclusion to this section, the array characteristics presented above are normalised and plotted in Figure 4.19. Sensor elements respond to loads applied elsewhere on the card, and cross-sensitivity between elements is demonstrated. The sharpest cross-sectional response is exhibited by piezoresistive elements in which current flows in the longitudinal direction of the smartcard. However, in order to capture spatial characteristics, sensing elements require to be well isolated from each other, and not respond significantly to a measurand applied to other elements. This is clearly not the case with the sensing elements considered here.



(a) Longitudinal Cross-Section (b) Lateral Cross-Section

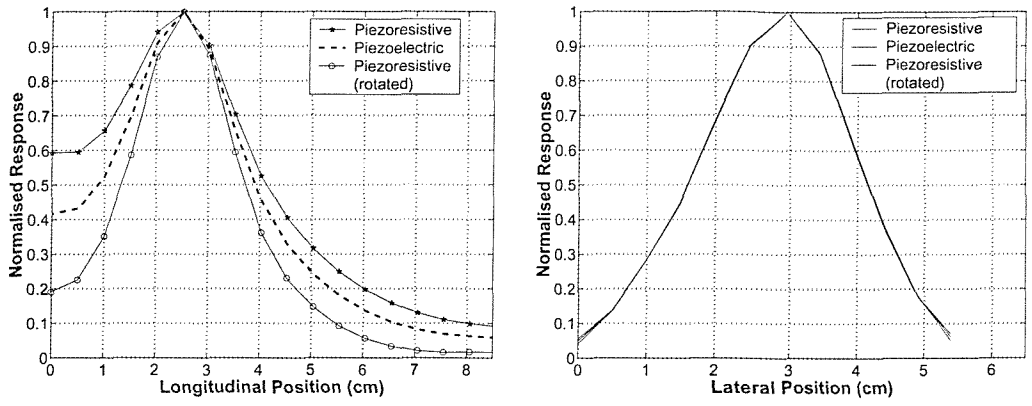
Figure 4.18: Piezoelectric Response with Position

Despite their attractive properties of low-cost, mechanical ruggedness and flexibility, PTF piezoresistive and piezoelectric sensors are unsuitable for the capture of spatial human characteristics. Nevertheless, both sensor types can be used to capture temporal characteristics, and their potential to do so is investigated in the next section.

4.8 Temporal Interactions

As indicated in Section 4.1, the tactile interactions of a person with a smartcard are effectively limited to tapping and pressing. Tapping may be considered a fast dynamic action, whilst pressing, or pressing and holding, represents a slower quasi-static interaction. This section compares the characteristics of piezoelectric and piezoresistive sensors in responding to these different interactions. Whilst the previous section made use of simulation to assess PTF sensors for capturing spatial characteristics, this section makes use of experimental measurement to assess the effectiveness of both sensors to capture temporal characteristics.

The sensor cards and data capture apparatus used for the experimental verification of the smartcard & sensor model (section 4.6), were also used in this work, and it was the author who pressed and tapped the sensor cards during all of the following measurements. The card was held in one hand, and tapped or pressed with the



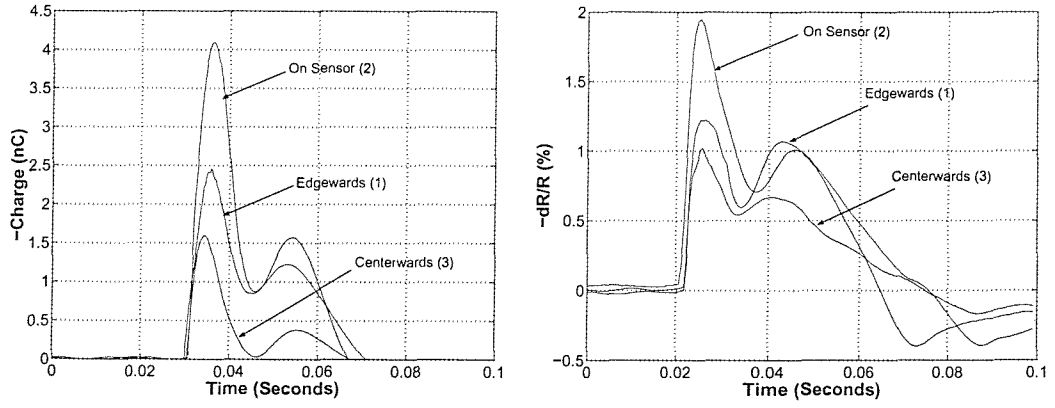
(a) Responses with Longitudinal Position

(b) Responses with Lateral Position

Figure 4.19: Normalised Sensor Responses with Position

fingers of the other. Three points of contact are considered and correspond to load cases 1,2&3, of the above analysis.

Figures 4.20(a) and 4.20(b) show sensor responses to single taps on the piezoelectric and piezoresistive smartcards, respectively. Both methods capture the dy-



(a) Piezoelectric Sensor

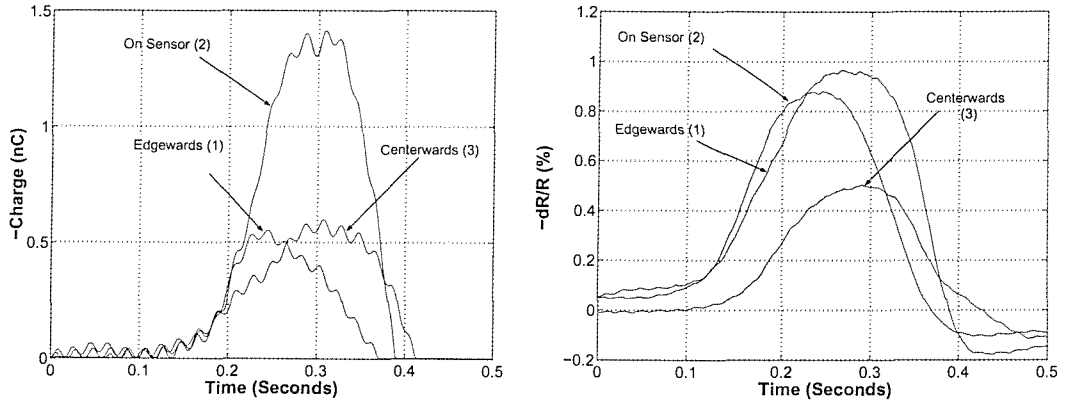
(b) Piezoresistive Sensor

Figure 4.20: Sensor Response to Tapping Loads

namics of this interaction adequately, as is clear from the above. However, there is some repeated evidence of hysteresis on the piezoresistive response. Hysteretic behaviour is observed on previous piezoresistive responses (see for example Figures

4.13 and 4.14(a)), however its influence is quite distorting in comparison to the piezoelectric responses (Figure 4.20(a)). The responses to each contact position are in broad agreement with FEA results, in that the tap on sensor generates the largest response, followed by tapping edgewards and then centrewards of the sensor.

Figures 4.21(a) and 4.21(b) show the piezoelectric and piezoresistive responses to a finger pressing-on, then releasing-from the card. It appears that the piezoresistive sensor offers better characteristics than the piezoelectric, whose response is affected by the prolonged proximity of a finger. The effect of prolonged finger contact on

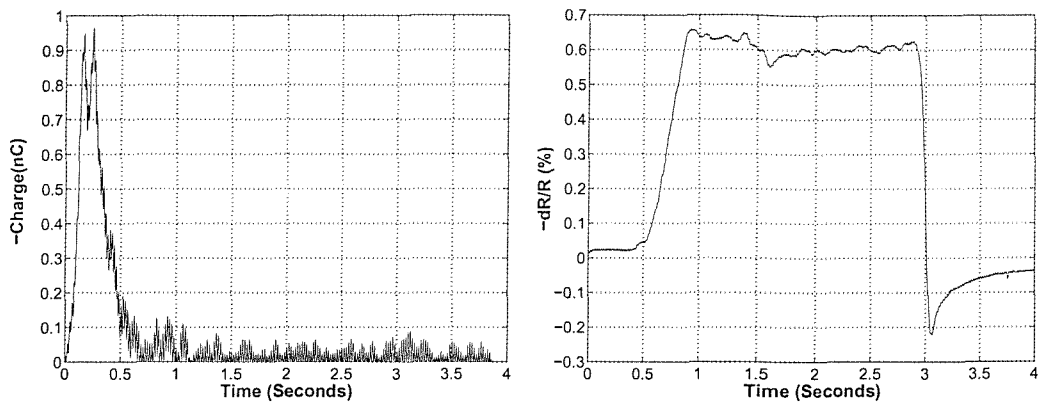


(a) Piezoelectric Sensor (b) Piezoresistive Sensor

Figure 4.21: Press Responses

the piezoelectric sensor is starkly demonstrated with Figure 4.22, which presents the response of both sensors to a finger pressing and continuing to hold for a short interval. The piezoelectric noise component, arising from 50Hz mains noise and stray capacitances, dominates the signal, and it is apparent that piezoresistive sensors offer significantly better static and quasi-static responses.

These results indicate that piezoresistive PTF sensors offer greater versatility than their piezoelectric counterparts. They are able to capture the dynamics of a fast on-off tap, whilst offering good quasi-static measurements. However, hysteresis and poor stability are the downside. Piezoelectric sensors, on the other hand, exhibit good low-noise dynamic properties and poor, noisy quasi-static characteristics. Further, piezoelectric measurements can be made with simple charge amplification circuitry, rather than the resistive bridge circuit used for piezoresis-



(a) Piezoelectric Sensor (b) Piezoresistive Sensor

Figure 4.22: Press and Hold Responses

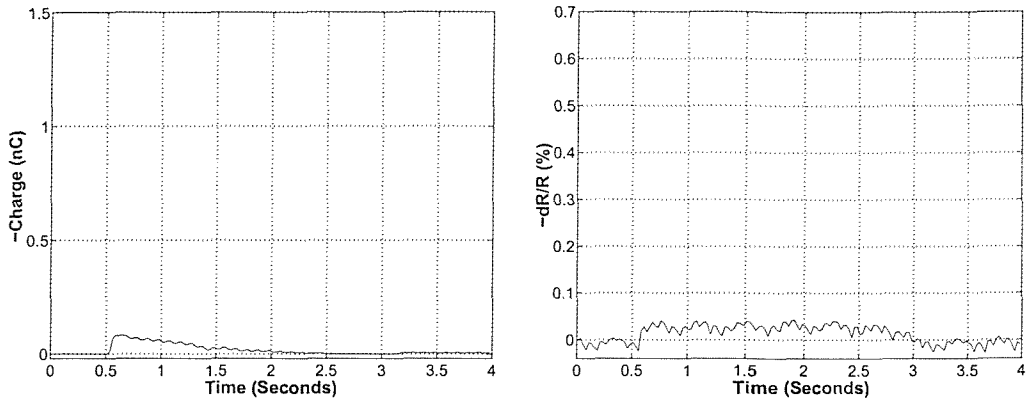
tive measurements. This offers significant advantage to the smartcard arena, in terms of circuit complexity, manufacturing and initialisation efforts.

4.9 Serendipitous Sensitivity

The flexing of a smartcard, in response to an applied load, has been shown to induce cross-sensitivities in multiple sensors bonded to a card. As such it may be viewed as a hindrance. However, the work of this section identifies flex of a smartcard as an important agent in enhancing the sensitivity of singular sensors bonded to a card.

In this experiment, both sensor cards were placed upon a flat rigid surface, and a 2.5N load was applied onto the sensor. Figures 4.23(a) and 4.23(b) show typical piezoelectric and piezoresistive responses, respectively. Both sensors exhibit significantly reduced responses in comparison to the responses of edge-supported cards, as shown in Figures 4.14(a) and 4.14(b). Clearly it is the flex of a smartcard which induces planar strains, contributing to the sensitivity of both sensor types.

Completely inadvertently, a method of improving the sensitivity of a constrained sensor has been found. In an attempt to print sensor layers directly onto the surface of a smartcard, rather than firstly onto a flexible substrate, it was discovered a



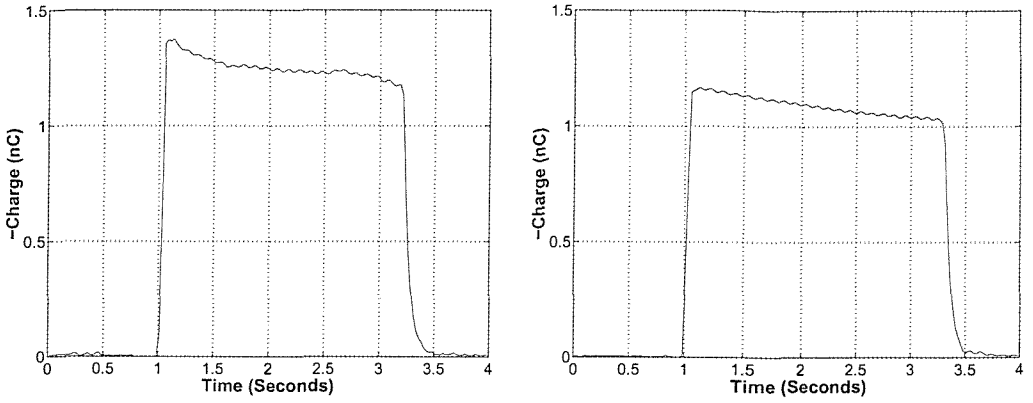
(a) Piezoelectric Sensor (b) Piezoresistive Sensor

Figure 4.23: Sensors on Mylar – Smartcard on Flat Surface

dome structure, analogous to the dome membrane switches of Figure 3.5, formed underneath the sensor. This is believed to be due to different thermal coefficients of expansion (TCE) of the smartcard and sensor materials, such that the sensor cooled at a faster rate than the card. With this sensor card fully constrained in the z-direction, the dome is able to deform in response to applied load, and hence induces increased strains to the sensor film.

Figure 4.24(a) shows the response of this modified smartcard sensor, being supported along its longitudinal edges, to an applied 2.5N load. Its reponse to the same applied load whilst constrained on a flat rigid surface is given in Figure 4.24(b), and is observed to be of comparable magnitude.

It should be noted that this was an entirely serendipitous discovery, and has only been demonstrated with piezoelectric sensors. The yield of this one-off fabrication batch was low, with only one working sensor from fifteen smartcards. This thesis does not consider further details or development of this approach to sensor fabrication, although it may offer some advantages in allowing PTF sensors to be used upon rigid components, and is hence considered in the further work section of Chapter 7.



(a) Smartcard on Supporting Edges (b) Smartcard on Flat Surface

Figure 4.24: Modified Piezoelectric Smartcard

4.10 Conclusions

Theoretical analysis of the pressure sensors considered in Chapter 3 indicated that sensors exhibit sensitivity to planar strains. Since a smartcard deforms in response to an applied load, strains propagate across the card, and this implies that sensors will respond to loads applied elsewhere on the card. First-order finite element analysis is used to assess the extent of strain propagation across a smartcard, and the extent to which an array of sensing elements respond to a single applied load, typical of human-finger interactions.

The FEA smartcard model represented a smartcard as a single plate of uniform thickness and material homogeneity. Where data was unavailable, assumptions about material properties and constraint conditions have necessarily been made. FEA results were used as input values to the piezoresistive and piezoelectric sensor models of Chapter 3, and theoretical sensor responses were calculated. These results were then compared with experimental measurement, and it was found that theoretical piezoresistive results were within $\simeq$15% of measured values, whilst piezoelectric measurements were approximately $\frac{1}{8}^{th}$ of theory. The discrepancy in piezoresistive results can be explained by factors such as imprecise modelling of constraint conditions and uncertainties in material property values. The very large discrepancy between piezoelectric values is likely to be dominated by inaccuracies in the composite mixture approach used to approximate Young's modulus of the

PZT sensor layer. The composite mixture model assumes uniform and exclusive mixing of two materials, and does not account for non-uniformities, such as air-gaps whose presence have been reported in the literature.

Nevertheless, this analysis captures the gross mechanical behaviour of a loaded smartcard, and with appropriate scaling, can be used to simulate on-card sensor arrays. Simulating arrays of small piezoresistive and piezoelectric sensors, it is found that sensors across the surface of the card will respond significantly to a singular load applied elsewhere. Whilst sensor responses are observed to diminish with distance from an applied load, they remain significant in proportion to the peak response. This effect becomes the limiting factor in bonding sensors to smartcards, and it is not practical to bond more than one sensor, without suffering cross-sensitivities between sensors.

However, it is shown that both sensors can capture aspects of the temporal (rather than spatial) interaction between a finger and the card. Piezoresistive sensors adequately capture both static and dynamic signals, whilst piezoelectric sensors are better suited to the capture of fast-changing dynamic signals rather than static or quasi-static signals. But piezoresistive sensors suffer from hysteretic properties, and whilst this is not characterised, it is observed to distort fast changing, dynamic interactions.

Finally, it is shown that the flex of a smartcard is an important agent in the sensitivity of both sensors, and constraining the vertical out-of-plane freedom of a smartcard significantly reduces sensitivity. A way around this has been discovered, and involves the formation of a dome structure underneath the sensor. In response to an applied load, the dome deforms, and the resulting strains are transferred to the sensor.

Chapter 5

A Novel Approach to Identity Verification

5.1 Introduction

The work of Chapter 4 demonstrates that planar PTF pressure sensors are not suitable for the capture of spatial characteristics, such as finger-geometry or finger-crease pattern. It is shown that the extent to which planar strains propagate across a smartcard, in conjunction with the sensing properties of PTF pressure sensors, causes sensors to respond to forces applied elsewhere on the card. It is this effect which renders the concept of a spatial sensing array infeasible, and hence, alternative discriminatory characteristics are sought.

Although the flex of a smartcard effectively limits the number of independent sensors to one single sensor, Chapter 4 also shows that smartcard flexing is largely responsible for the absolute sensitivity of PTF pressure sensors to applied loads. This effect can be exploited to capture temporal, rather than spatial characteristics: Piezoelectric sensors were shown to capture the dynamics of a fast changing interactions, whilst piezoresistive sensors performed best at capturing static and quasi-static, signals.

This chapter presents and demonstrates a novel approach to identity verification which is based upon the temporal interactions between a finger and pressure sen-

sor. This new approach is founded upon the principles of keystroke dynamics, and is first reported in Henderson & Hartel (2000) and with greater depth in Henderson, White, Veldhuis, Slump & Hartel (2002a). In this proposed approach, an individual taps out a self-selected rhythm on a singular pressure sensor, bonded to a smartcard. In common with much of the work on keystroke dynamics, inter-tap and tap-duration times are used as features. However, the use of a pressure sensor allows the additional feature of pressure amplitude, providing further scope for discrimination. Whilst two authors suggest modifying keyboards to include pressure sensitive elements (Spillane 1975, Young & Hammond 1989), this proposed mechanism is quite different in its use of only one single pressure sensor, rather than a full keyboard.

Background justification for this approach is given, and similarities with keystroke dynamics become apparent. A comprehensive review of keystroke dynamics is thus presented and used as the basis for an experimental trial. An experiment involving 34 volunteer participants is described and significant visual differences between the characteristics of individuals are observed. Appropriate verification functions are investigated and an equal error rate of 2.3% is found to be achievable under controlled laboratory conditions.

5.2 Background

From an empirical viewpoint, the recognition of people from a series of taps or pulses has some precedent – early telegraphic operators recognised other operators by the way in which they keyed information. Operators developed a distinctive *fist* or telegraphic style which could be recognised (Bryan & Harter 1897). This is further illustrated with an example from French Military Intelligence during the First World War: Telegraph interceptors were able to recognise the enemy operator responsible for transmitted signals, whilst a network of receiving stations could triangulate the source of radio transmissions. Combining information from these sources, Intelligence officers were able to surmise the movements of individual battalions thereby providing strategic advantage to the Allies (Singh 1999).

On a more mundane level, Umphress & Williams (1985) observe that:

> "Anyone who sits within earshot of a typist, or has an office next to a keypunch room is usually able to recognise typists by keystroke patterns."

They go on to suggest that:

> "Keyboard characteristics are rich in cognitive qualities and give great promise as a personal identifier."

In fact, since the publication of this paper much work has gone into making recognition by keystroke patterns, or keystroke dynamics, a realistic and viable biometric (Obiadat 1998). An explanation for the discriminating basis of keystroke dynamics is offered by Rumelhart & Norman (1982), suggesting that differences in typing style are due to both the physical characteristics of the hand, such as finger length and agility, and the level of motor control of a person.

Indeed we have everyday experience of 'tap' recognition in recognising a person by the way in which they knock on a door. Furthermore, the use of prescribed knocking patterns has long been employed as an indication of belonging to so-called secret societies, or of group membership. For an example see Clancy (1999).

The characteristic patterns of footsteps can cause dogs to bound towards their owners, or bark in warning at the unknown. Moreover, as pointed out by Obiadat (1998):

> "Since the beginning of time, humans have been able to recognise [a] person from the sound of steps of the individual."

A practical implementation of this is illustrated in the 'Smart Floor' (Orr 2000), in which an instrumented floor covering is the basis of a system for identifying people from their footsteps. There is overlap between this method of identification and that of Gait Recognition (Nixon et al. 1998). Gait recognition looks for biomechanical differences in leg and body movements of a person's stride, whilst the Smart Floor method seeks identification based upon the resulting pattern of reaction forces, between feet and ground.

In a similar manner, the pressure sequence method seeks recognition by the pattern of pressure pulses between fingertip and pressure sensor, resulting to some measure

from the biomechanical characteristics of a person's hand and wrist. That the human hand is complex and offers scope for discrimination between people is demonstrated in its composition of 19 bones, 19 joints and 20 muscles, combining to give 22 degrees of freedom (Kandel 1981).

Shaffer (1982) investigates rhythm and timing in common tasks. It is found that:

> "...temporal rhythm is the realisation of a schedule in a motor program..."

and:

> "...given a schedule, the motor system can produce a movement to the next temporal target, taking a previous target as a reference point for the movement trajectory."

This is applied to the task of 'tapping':

> "In repetitive tapping, with one finger, the previous tap provides the reference point for the next one."

Hence, tapping is in some way a function of a person's motor program, or neurophysiology. It appears that the expression of some rhythm (or sequence of taps) is tempered both by the neurophysiological and biomechanical make-up of a person, and one would expect the output of even the same rhythm by two people to be subtly different.

5.2.1 Objectives

For the pressure sequence method to gain merit as a verifier a high level of discrimination must be demonstrated. Since there are strong similarities between the proposed pressure sequence method and that of keystroke dynamics, the discrimination methods of keystroke dynamics are investigated for their potential to discriminate between people, based only on a sequence of taps on a single pressure sensor. This is justified in that both result from similar neurophysiological and biomechanical mechanisms, and at a minimum, both consider time intervals between finger taps.

5.3 Keystroke Dynamics

Keystroke dynamics refers to the identification or verification of an individual by making use of their characteristic typing style. This can be performed on a one-off basis, generally at the start or login of a computer session, or it can be performed on a continual basis throughout the lifetime of a session. These approaches are known as *static* and *dynamic* keystroke dynamics, respectively. Since verification of a smartcard holder occurs once during a session, it is apparent that static verification is most appropriate, nevertheless certain aspects of dynamic verification are useful, and both approaches are described for completeness.

Static verification tends to be based upon the typing pattern of a fixed short login string, such as the person's name or username, and the features of *inter-key*[1] and or *key-hold* times form the basis of discrimination. Dynamic recognition (or verification), on the other hand, must be able to deal with arbitrary free text-independent prose, and in order to do this text must be broken down into small recognisable and characteristic units. One primary way of doing this is to record keystroke latencies between specific character pairs, called *digraphs*. Using only the lowercase ASCII character set there are 26x26 possible digraphs or 27x27 if the space character is included. Some commonly occurring three-character units, called *trigraphs* (for example *ion*), and four-character groups, called tetragraphs (for example *tion*) are also found to be typed consistently and may be used in the discrimination process. Di–, tri–, and tetra– graphs are compared continuously to reference profiles throughout the session.

It has been suggested that typing style is analogous to handwriting, since both stem from similar neurophysiological factors and from the complexity of the human hand (Joyce & Gupta 1990). Indeed, static verification may be considered the keystroke equivalent of hand-written signature verification, in that both analyse the expression of a fixed string. Dynamic recognition / verification, then, is analogous to handwriting recognition given an arbitrary sample of prose.

[1] inter-key times are also known as *keystroke latencies*

5.3.1 Static Fixed-String Verification

Spillane (1975) first proposed a keyboard with means of recording time and pressure patterns as a user keys an entry code. He suggests that this pattern will be unique to each user, and proposes its use as a means of verifying identity. By varying the length of the entry code, Spillane suggested that any desired level of security could be achieved. No experiments or results are reported.

The patent of Garcia (1986) proposes using timing information between keystrokes entered upon a simple existing keyboard to verify the identity of a computer user. Garcia suggests using a common, often used string such as a person's name or login username, which should be entered with a high degree of consistency. His method is to enrol users by entering their names *a number of times*, thereby creating a database of timing vectors whose components are the latency times between subsequent keypresses. At some future time, when a user requires access to the system, the claimant enters their name, from which latency times are extracted. A timing vector is generated and statistically compared to the enrolment timing vectors. If these are *sufficiently* similar, access is granted.

In generating a reference vector, Garcia advocates the removal of outliers – suggesting that isolating the enrolment environment from noise and distractions will produce a cleaner reference vector. Garcia proposes using the Mahalanobis distance function to determine the statistical similarity between the enrolment and test vectors, and is defined as follows:

$$D(\mathbf{R}, \mathbf{T}) = (\mathbf{R} - \mathbf{T})^* \cdot \mathbf{C}^{-1} \cdot (\mathbf{R} - \mathbf{T}) \tag{5.1}$$

where $\mathbf{R}$ is the reference vector, $\mathbf{T}$ is the test vector, C^{-1} is the inverse of the covariance martix generated from an individual's enrolment vectors, and the operator $(\cdot)^*$ denotes the transpose matrix.

If $MD < 50$, a user will be immediately accepted. For $50 \leq MD < 100$, a request is made for the user to reenter their name. If $MD \geq 100$, the user is denied access. With these thresholds, Garcia claims a probability of false acceptance of 0.0001 (0.01%) and a probability of false rejection of 0.5 (50%).

Garcia widens the scope of his patent, limiting it not only to standard computer

keyboards, with his suggestion that a variety of input devices could be used, including; telephone keypads, numeric keypads, electronic piano keyboards, or even a single telegraphic key. The critical feature of his scheme is that: "each successive piece of information is distinct and measurable, with a resolution of between 1 and 500 μs".

According to Garcia, the discriminating mechanism is based upon the concept that: "The co-ordination of a person's fingers is neurophysiologically determined and unique." He continues, by saying: "Any situation in which a person has to reproduce a rapidly changing pattern on one or more keys will produce a unique signature, in terms of the time delays between each key pressed." Whilst offering no substantive evidence for this theory, it is broadly aligned with that of (Shaffer 1982) referenced above.

Young & Hammond (1989) propose the use of a specially designed keyboard, to capture both interkey times and keystroke pressure, in order to verify a computer user. Authorized users enrol by typing a selected passage of text, from which a *plurality of features* may be extracted. Features are suggested to include interkey times between selected di or trigraphs, and keystroke pressures. They propose generating a reference vector, $\mathbf{R}$, whose d components are selected features of the above. A test vector, $\mathbf{T}$, containing the specified feature set, is constructed either statically at login or dynamically during a session, when a user wishes access or is using their system.

Their verification scheme involves computing a Euclidean distance measure (ℓ_2 norm):

$$\|\mathbf{R} - \mathbf{T}\|_2 = \left[\sum_{i=1}^{d} (r_i - t_i)^2\right]^{\frac{1}{2}} \tag{5.2}$$

where r_i is the i^{th} component of $\mathbf{R}$ and t_i is the i^{th} component of $\mathbf{T}$. If $\|\mathbf{R} - \mathbf{T}\|_2$ is less than a certain threshold, then the test and reference vectors are deemed to be *sufficiently* similar and the user is either granted access, or allowed continued access to the facilities.

No information is given about threshold determination, nor achievable error rates with their proposed system. Indeed, it is not clear from their patent, whether the authors conducted trials with their apparatus, or are merely logging stake to a

claim.

Joyce & Gupta (1990), proposed the identity verification of computer users based upon the entry of a short signature string. Their analysis method is similar to that of Garcia, relying solely upon keystroke latencies, captured to a resolution of 0.01s, rather than specific digraph latencies. Their login signature comprises of username, password, firstname and lastname. In common with Garcia (1986), they suggest that a 'well-known' regularly typed string can be quite consistent. In their experiment, 33 people were enrolled each providing 8 login samples from which a reference profile was created. After typing their enrolment sequences each user provided 5 further login attempts, yielding a total of 165 self login attempts. Six from the 33 participants were randomly chosen as targets, and the remaining 27 participants each attempted 5 impostor logins per target. It should be noted that the impostors were not present during the target's enrolment sessions, and that all data, both test and reference, were captured during a single session.

By considering each signature as a vector whose components are the keystroke latencies between each character, each signature will be composed of 4 vectors of latency values. The mean reference signature will be given by:

$$\mathbf{R} = \{\mathbf{R}_{username}, \mathbf{R}_{password}, \mathbf{R}_{firstname}, \mathbf{R}_{lastname}\} \tag{5.3}$$

The mean of each latency is calculated from the eight enrolment signatures. Each latency is then compared to its corresponding mean. Any outliers, greater than 3 standard deviations will be removed, and the mean of that latency recalculated.

The difference between a test vector, $\mathbf{T}$, and the mean reference vector, $\mathbf{R}$, is calculated as the ℓ_1 norm:

$$\|\mathbf{R} - \mathbf{T}\|_1 = \sum_{i=1}^{d} |r_i - t_i| \tag{5.4}$$

where $\mathbf{R}$ is the d dimensional reference-vector, generated from the mean of each of the components in the user's enrolment vectors, $\mathbf{T}$ is a d dimensional test vector, and r and t are the components of $\mathbf{M}$ and $\mathbf{T}$ respectively.

By looking at the way in which each enrolment signature differs from the mean reference vector, a mean difference and standard deviation are calculated for each

user. If each of the enrolment signatures are $\mathbf{S}_1$, $\mathbf{S}_2$, ... , $\mathbf{S}_j$, respectively, then, $\|\mathbf{R}-\mathbf{S}\|_1$ is calculated for $j = 1$ to 8. The mean and standard deviation of these norms are calculated, and used to decide an acceptance threshold for each user. A test sequence, $\mathbf{T}$, will be accepted if $\|\mathbf{R}-\mathbf{T}\|_1$ falls within the mean $\pm$ a proportion of the standard deviation of the enrolment norms.

Using an acceptance threshold of 1.5 standard deviations, false rejection and false acceptance rates of 16.67% and 0.25%, respectively, are reported. Increasing the threshold to 2.5 standard deviations resulted in a false rejection rate of 6.67% and a false acceptance rate of 1%. Joyce and Gupta, graphically characterise further results at a range of thresholds, and visual inspection reveals an equal error rate of around 3%.

Although these results are impressive, the authors suggest that the 'shape' of each signature could offer further discrimination. Differences in latency values across two different signatures may be small, but the 'shape' or slopes of the signatures could be significantly different. Hence, the authors propose a further classifier based upon the difference between successive latencies, 'Slope Difference'. If $\mathbf{I} = \{i_1, i_2, ..., i_{n-1}\}$ is the vector of slopes of the mean reference signature, $\mathbf{R}$, and $\mathbf{J} = \{j_1, j_2, ..., j_{n-1}\}$ is the corresponding vector of slopes from the test vector, $\mathbf{T}$, then a measure of slope difference can be given by:

$$\sum_{k=1}^{n-2} \left(|i_k - j_k| + |i_{k+1} - j_{k+1}|\right) w_k \tag{5.5}$$

where

$$w_k = \frac{|i_{k+1} - i_k|}{\sum_{m=1}^{n-1} \max |i_{m+1} - i_m|} \tag{5.6}$$

Since the objective in this classifier is to bring out distinctive features of the signature, the slope differences are weighted by the amount of slope change in the reference signature, w_k. This classifier proved to be slightly worse than the ℓ_1 norm classifier, and visual inspection of the author's results suggests an equal error rate of around 5%.

Bleha et al. (1990) describe an experiment in which identity verification is based upon the interkey times of a user's name. Using a dedicated PC in a separate office, 14 volunteers each entered 30 enrolment samples from which a user's reference

file was created. Data held in the reference file was used to calculate a mean reference vector and covariance matrix. A further 25 volunteers were designated the role of impostor (13 of whom were encouraged to practice the person's name whose identity they were claiming). After the enrolment process, volunteers were encouraged to participate freely depending upon their availability and willingness. Reference files were updated on a weekly basis, using the last 30 login samples, and a total of 539 valid login samples and 768 impostor attempts were captured, over an eight week period.

Two minimum distance classifiers were used, and both were normalised to accommodate variation in name length. The first classifier is given as:

$$D_j(\mathbf{T}) = \frac{(\mathbf{T} - \mathbf{R}_j)^* (\mathbf{T} - \mathbf{R}_j)}{\|\mathbf{T}\| \, \|\mathbf{R}\|} < \theta_1 \tag{5.7}$$

whilst the second as

$$d_j(\mathbf{T}) = \frac{(\mathbf{T} - \mathbf{R}_j)^* \, \mathbf{C}^{-1} (\mathbf{T} - \mathbf{R}_j)}{\|\mathbf{T}\| \, \|\mathbf{R}\|} < \theta_2 \tag{5.8}$$

where the participant is claiming to be user j , $\mathbf{T}$ is the test vector, $\mathbf{R}$ is the mean reference vector, $\mathbf{C}^{-1}$ is the covariance matrix, and θ_1 and θ_2 are the threshold values. Values of 0.029 and 0.000029 were used for θ_1 and θ_2, respectively.

Under Bleha's scheme, a test vector will be accepted as valid if it satisfies both classifiers. From a total of 539 valid logins, 44 were rejected, from 768 impostor attempts, 22 were accepted, resulting in a false rejection rate of 8.1% and a false acceptance rate of 2.8%.

In a similar experiment involving 10 valid users and 14 impostors, Bleha & Obaidat (1993) assessed the use of the perceptron algorithm to verify a user's keystroke characteristics. Valid users typed their first and last names, impostors were given the names of the legitimate users and asked to break the verifier. Data samples were collected over a period of eight weeks, each person providing 2 samples per session, allowing at least one day between sessions. A total of 50 valid and 50 impostor samples were collected per legitimate user, 40 of which were used for network training, the remaining 10 for testing.

The authors achieved an average false rejection rate of 8%, and a false acceptance

rate of 9%. It is not clear which 10 samples were used for testing, nor whether the impostors each attacked one specific user, or a mixture of users.

Brown & Rogers (1993), with the intention of verifying identity solely upon entry of a short login string, extended the techniques outlined above. The authors captured keyhold times in addition to simply keystroke latencies. As a user entered their name on login, key press and key release times were time-stamped and recorded to millisecond resolution. From these measurements key hold times were calculated by subtracting the key press time from the key release time from each character typed. Interkey times were calculated by subtracting the first key release time from the second key press time. This, the authors note, was frequently negative, indicating that the second key was pressed before the first fully released.

Their experiment enlisted a total of 46 people, each typing their own name several times, interspersed with the names of others which served as impostor data. Each user participated in two data capture sessions – data from the first serving as a training set, the second as test data. The sessions were separated by an interval of at least one week. Each input was converted into a vector whose components were keyhold and interkey latency times.

Three classifiers were compared in the experiment; two neural techniques – an *ADALINE* and a *Backpropagation* Network; and a Euclidean distance measure. An average of 37.7 user data sets and 35 impostor sets were used for network training, per user. Networks were tested with an average of 12.4 user test sets and 30 impostor sets. Each of the classifiers was set up to minimise false acceptance errors, with their collected data and set thresholds false acceptance rates of 0% were achieved. This, as the authors point out, does not mean that no impostor will ever be accepted by these methods, rather that the classifiers were biased against allowing impostor passes.

The 46 volunteers arose from two distinct groups of 25 and 21 students. As a result Brown and Rogers have stated their results for each group. Data from the first group, of 241 self login attempts, generated 42 false rejections with the ADALINE technique (17.4%), 29 using the backpropagation method (12.0%) and 36 with the distance measure (14.9%). Meanwhile the second group of 330 self login attempts, resulted in 132 false rejections using the ADALINE classifier (40%), 70 using the backpropagation method (21.2%) and 78 with the distance classifier (23.6%). The

authors are unclear about the reasons for second group's poor performance. They suggest that 'individual variation' and an 'unfamiliarity' with the computing environment is most likely the cause. Combining the results from both groups, we find false rejection rates of 30.5%, 17.3% and 20%, for the ADALINE, Backpropagation and Euclidean methods, respectively.

Obaidat & Sadoun (1997) used a modified PC interrupt handler to record both key hold and keystroke latencies, to 0.1 millisecond resolution, during login attempts. In common with (Brown & Rogers 1993), (Bleha et al. 1990), (Bleha & Obaidat 1993) and (Garcia 1986), the authors avoided potential security problems, by using a string other than a user's password. Looking for a string which is typed regularly and consistently, they proposed using a person's username as the verification string.

Fifteen users each provided 255 valid usernames per day, over a period of 8 weeks. Fifteen impostors provided 15 login attempts using each of the valid usernames, resulting in 225 impostor attempts for each of the valid users, per day. The captured data was split into two halves – one for classifier training, the other for test data.

The authors employ and compare a range of classical pattern recognition techniques: *K-Means Algorithm*, *Cosine Measure Algorithm*, *Minimum Distance Algorithm*, *Bayes Decision Rule* and a *Potential Function Algorithm*. A number of neural techniques were also employed: Backpropagation Network, *Counterpropagation* Network, *Fuzzy ARTMAP*, *Radial Basis Function* Network, *Learning Vector Quantization* Network, *Sum of Product* Network and a *Hybrid Sum of Product* Network.

Using combined keyhold and keystroke latency times, it was found that the most successful pattern recognition technique was the Potential Function algorithm, offering a false rejection rate of 1.9% and false acceptance rate of 0.7%. According to the authors, the Bayes Decision Rule was the next most successful classifier giving only marginally poorer results. Visual inspection of their results shows that the Minimum Distance Measure algorithm came third, offering a false rejection rate of around 11% and a false acceptance rate of around 7%.

The Potential Function Algorithm is described as follows: Each vector of a class defines a point in d-dimensional space, these points are iteratively weighted to

generate a decision plane in conventional recognition schemes, such as the perceptron algorithm. In this scheme, however, rather than using the absolute positions of each feature vector to define a decision surface, a potential function method is used. The decision surface is modified by the sum of potential functions, resulting from each of the training vectors of a class. This method must be iteratively trained, to generate a discriminating decision surface.

The potential function used by Obiadat is given as:

$$K(\mathbf{T}, \mathbf{R}_j) = e^{\{-\alpha \|\mathbf{T}-\mathbf{R}_j\|^2\}} \tag{5.9}$$

where $\mathbf{R}_j$ is the mean feature vector for user j, $\mathbf{T}$ is the test vector and α is a positive constant, determining the relative amplitudes of the potential function. In Obaidat's implementation α is set to 1.

Bayes decision rule as implemented by Obiadat, is given as

$$P\langle W_j|\mathbf{T}\rangle = \frac{P\langle W_j\rangle \, P\langle \mathbf{T}|W_j\rangle}{P\langle \mathbf{T}\rangle} \tag{5.10}$$

where $\mathbf{T}$ is the test vector, and W_j is the j^{th} user class.

The decision boundary is given by

$$d_j = \ln[P\langle W_j\rangle] + \mathbf{T}\mathbf{C}_j^{-1}\mathbf{R}_j^{-1/2}\mathbf{R}_j\mathbf{C}_j^{-1}\mathbf{R}_j^* - \ln[\mathbf{C}_j] \tag{5.11}$$

where $\mathbf{R}_j$ is the reference vector of the j^{th} user class W_j and $\mathbf{C}_j$ is the covariance matrix for this user class.

Obiadat's minimum distance verifier is given as

$$D_j = \|\mathbf{T} - \mathbf{R}_j\| = \sqrt{(\mathbf{T} - \mathbf{R}_j)^* \, (\mathbf{T} - \mathbf{R}_j)} \tag{5.12}$$

Squaring and expanding (5.12), gives

$$D_j^2 = \|\mathbf{T} - \mathbf{R}_j\|^2 = \mathbf{T}^* \, \mathbf{T} - 2(\mathbf{T}^*\mathbf{R}_j - \frac{1}{2}\mathbf{R}_j^* \, \mathbf{R}_j) \tag{5.13}$$

Since $\mathbf{T}^* \mathbf{T}$ is independent of j for all classes, the minimum distance occurs when

$$d_j = \mathbf{T}^* \mathbf{R}_j - \frac{1}{2}\mathbf{R}_j^* \mathbf{R}_j \tag{5.14}$$

is at a maximum. This is the distance measure used.

The neural classifiers improved upon these error rates with the Learning Vector Quantization, Radial Basis Function Network and the Fuzzy ARTMAP schemes all offering zero errors for both false rejection and false acceptance. The Backpropagation, Hybrid Sum of Product and Sum of Product Networks, were successful, offering false rejection rates of 0%, 1% and 4% and false acceptance rates of 1%, 0.5% and 2.5% respectively.

The use of keyhold times only and keystroke latencies only was also investigated. It was found that, although keyhold times offered better discrimination than keystroke latencies, combining keyhold and latency times offered the best discrimination. For example, the Potential Function algorithm offered a FRR of 2.9% and 4.7% for keyhold and keystroke latencies, respectively. False acceptance rates with the same classifier were 1.6% and 2.2% for keyhold and latency times.

Robinson et al. (1998) address some of the problems encountered by other researchers, namely the reluctance of users to type more than their username and password, and the security implications of including password information in the timing vector. Hence, they propose using only the timing pattern contained within a person's username. In common with Brown & Rogers (1993), Obaidat & Sadoun (1997), and Monrose & Rubin (2000), key hold and keystroke latency times were both recorded. In this experiment, key hold and latency times were measured to a resolution of 0.1milliseconds, using a modified DOS keyboard handler. During the routine use of computer facilities, valid login attempts from 10 valid users were captured. Several hundred 'forgeries' were collected from a further 10 people.

Three different classifier schemes were used; a Minimum Intra-Class Distance Classifier, a *Non-Linear Classifier* and an *Inductive Learning Classifier.* These approaches are defined as follows

The *Minimum Intra-Class Distance Classifier* uses a thresholded Mahalanobis dis-

tance approach, defined in (5.1). A test vector, $\mathbf{T}$, will be accepted if

$$D(\mathbf{T}, \mathbf{R}) < \theta \tag{5.15}$$

where $\mathbf{R}$ is the reference vector, and θ is the threshold distance for acceptance.

Non-Linear Classifier: This classifier independently compares each component, t_i, of the test vector $\mathbf{T}$, with the corresponding component of the reference vector, and its standard deviation, σ_i. A test vector will be accepted as belonging to the user class W if

$$|t_i - r_i| < \tau\sigma_i \quad \forall i \tag{5.16}$$

In their experiments, Robinson et al. used an acceptance threshold of 3 for τ.

Inductive Learning Classifier: Based around a classifier proposed by Chan (Chan & Wong 1990), this inductive learning classifier is trained with 10 valid samples and 20 invalid 'forgery' samples.

It was found that in all three classifiers, key hold times alone, performed better then key stroke latencies, alone. The best classifier proved to be the inductive learning classifier, using both key hold and interkey times. Its error rates were found to be 10% and 9% for false rejection and false acceptance rates. The MICD and Non-Linear classifiers, using both key hold and interkey times, offered false rejection rates of 23% and 31%, and false acceptance rates of 24% and 31%, respectively.

5.3.2 Dynamic Free-Text Verification

Gaines, Lisowski, Press & Shapiro (1980) sought to answer the question:

> Can people be identified by the way in which they type?

Six professional typists were enlisted and asked to type three paragraphs of prose on two occasions separated by an interval of four months. The time intervals between each keypress were measured and recorded. Entered text consisted of only lower case letters and spaces, resulting in 27x27 possible digraphs. From these 729 digraphs, most did not occur, and others only infrequently. Therefore to ensure representative statistics, their analysis was limited to those digraphs

occurred at least ten times. By comparing the probability distributions of the second set of data with the first, Gaines discovered that each typist indeed had a characteristic 'typing signature', which could be used as a basis for discrimination. This method gave a False Acceptance Rate of zero and a False Rejection Rate of 4%. Further analysis identified five *key* or *core* digraphs (*in*, *io*, *no*, *on* and *ul*) which offered zero error rates.

Building upon the work of Gaines et al. (1980), Umphress & Williams (1985) investigated the use of typing style as a means of replacing existing username & password security schemes. Their ultimate objective was the identification of a person as they begin typing, and it is further suggested that their approach could be used to provide constant surveillance throughout the lifetime of a computer session (Dynamic Verification).

Their experiment involved 17 people, each typing approximately 1400 characters of prose, which served as a reference and was followed a few days later by a test text of around 400 characters. Each keystroke was time-stamped to the nearest hundredth of a second, and recorded. According to Shaffer (1982) typists first look at text to be typed, 'load' an amount into a memory 'buffer' then output the buffer contents onto the keyboard. The length of the buffer is around 6 to 8 characters long, manifesting itself as pauses in typing, occurring with a frequency of 6 to 8 characters. In an effort to clean their data, Umphress and Williams considered only the first 6 characters of a word and discarded the others. Each digraph time was stored in the appropriate location of a 26x26 element matrix, and the mean and standard deviation of each digraph were calculated, and additionally the mean latency for all digraphs.

The test text from each person was pre-treated in the same manner, and resulting digraphs compared to the reference digraphs. If a test digraph fell within 0.5 standard deviations of the reference it was accepted as valid. If a sufficiently high proportion of individual digraphs passed the reference comparison ($\geq$60%) then the test profile passed their digraph test. A second test considered the gross structure of a person's typing style, and was characterised by the mean keystoke latency time for all digraphs, thereby providing a measure of the person's typing speed.

From the outcome of both tests, a degree of confidence was assigned to the test

profile arising from the same person as the reference profile. If a participant passed both tests then a high degree of confidence would be assigned, and their sequence would be accepted, otherwise their sequence was rejected. A false acceptance rate of approximately 6% at 12% false rejection is reported.

Leggett & Williams (1988), performed essentially a replication of Umphress & Williams (1985), with the goals of determining which digraphs offered most discrimination, and to test the effect of a larger experimental population. 36 people participated, each providing a reference sample of 1075 characters, followed a few days later by a test sample of 537 characters. A false acceptance rate of 5% with 5.5% false rejection, using all lowercase and space digraphs, is reported.

Leggett, Williams & Usnick (1991) further extended their work into dynamic verification, and suggested the possibility of continuously monitoring a person's typing style. Using the data of Leggett & Williams (1988), sequential statistics were applied to characters as if they were being typed live, resulting in a false acceptance rate of 12.8% at 11.1% false rejection.

The experiment reported by Monrose & Rubin (2000), describes an attempt at developing a non-static, or dynamic keystroke identifier. Two regimes were investigated; typing of structured, text-dependent prose and the typing of arbitrary, text-independent prose. Data was collected from 63 users over an 11 month period. Subjects were asked to retype a few sentences from a list of available phrases (fixed-text), or to type a few sentences on the spur of the moment (free-text). The experiment software was downloaded and run on the user's local machine, from which results were automatically emailed back to the authors. This enabled each person to undertake the experiment at many and varied times, rather than providing both enrolment and test data during the same session. Keystroke durations and specific digraph latencies were collected over a number of sessions and split into training and testing data sets. No mention of the average length of the gathered text samples was made.

Four analysis methods were investigated: *Euclidean distance measure* (5.2). Since the author's intention was identification, an unknown test vector, $\mathbf{U}$ is attributed to the reference vector which minimises this distance measure.

Non-Weighted probability measure. If a d dimensional pattern vector, $\mathbf{R}$, represents

a user's mean reference vector, then each component of $\mathbf{R}$ can be described by the quadruple

$$\mathbf{R}_i = \langle \mu_i, \sigma_i, o_i, x_i \rangle \tag{5.17}$$

where μ_i, σ_i, o_i and x_i, each represent the mean, standard deviation, number of occurrences and data value of the i^{th} component of the reference vector $\mathbf{R}$, respectively.

Assuming that each feature is distributed normally, a score is calculated between an unknown vector, $\mathbf{U}$, and the reference vector, $\mathbf{R}$, as

$$S(\mathbf{R}, \mathbf{U}) = \sum_{i=1}^{d} s_{u_i} \tag{5.18}$$

where

$$s_{u_i} = \frac{1}{o_{u_i}} \left[\sum_{j=1}^{r_i} P\left(\frac{x_{ij}^{(u)} - \mu_{r_i}}{\sigma_{r_i}} \right) \right] \tag{5.19}$$

and $x_{ij}^{(u)}$ is the j^{th} occurrence of the i^{th} feature of $\mathbf{U}$ and $P(\,)$ is the Gaussian probability function. The score for each component, u_i, is based upon the probability of observing the value u_{ij}, in the reference vector, $\mathbf{R}$, given the mean, μ_{r_i}, and standard deviation σ_{r_i}. The unknown vector, $\mathbf{U}$, is associated with the reference vector which maximises $S(\mathbf{R}, \mathbf{U})$.

Weighted probability measure. Some digraph features occur more often than others and are hence typed more consistently and reliably. For this reason the authors investigated the use of weight factors, weighting more heavily those digraphs which occur most frequently. The score between vectors $\mathbf{U}$ and $\mathbf{R}$ is given as

$$S(\mathbf{R}, \mathbf{U}) = \sum_{i=1}^{d} s_{u_i} w_{u_i} \tag{5.20}$$

where w_{u_i} is the weight of feature u_i, the ratio of its occurrences relative to all other features in the feature vector $\mathbf{U}$. In common with the Non-Weighted probability measure, the unknown vector, $\mathbf{U}$, is associated with the reference vector which maximises $S(\mathbf{R}, \mathbf{U})$.

Bayesian Classifier. If $\mathbf{U}$ and $\mathbf{R}$ are the unknown and reference feature vectors, respectively, ς the interclass dispersion vector and w_i the weight vector, then the

distance between the two feature vectors is expressed as

$$\Delta^{\alpha}(\mathbf{U},\mathbf{R}) = \sum_{i=1}^{k} w_i \left(\frac{\mathbf{u}_i - \mathbf{r}_i}{\varsigma_i} \right)^{\alpha} \tag{5.21}$$

where the feature vectors u_1, u_2, ... ,u_k and r_1, r_2, ... ,r_k are derived through factor analysis, reducing the dimensionality of $\mathbf{U}$ and $\mathbf{R}$ whilst preserving the correlation between features. See (Monrose & Rubin 2000) for details. The authors found that identification based upon structured fixed-text was more reliable than arbitrary free-text. Correct identification rates for fixed-text identification were 83.2%, 85.6%, 87.2% and 92.1% for the Euclidean, Non-Weighted Probability measure, Weighted Probability measure and the Bayesian classifiers, respectively.

Alexandre (1997) investigates an entirely different method of keystroke dynamics. Taking the adage: 'It's not what you type, but the way in which you type it.', to its natural conclusion, his method is to recognise a user based upon the correlation between randomly pressed keys. Termed *Keyboard Behavioural Signature*, 30 users enrolled by randomly typing a sequence of around 1000 keypresses on keys 1 through 8. Alexandre considers this data to be 10 sequences of 100 keystrokes each. From these sequences a single reference pattern, for each user is generated.

Two approaches have been considered to verification. The first method is to simply use the frequency of each keypress in the reference pattern, $\mathbf{R}$, and compare this to the keypress frequency of the test signature, $\mathbf{R}$. An Euclidean distance measure is employed:

$$D(\mathbf{T},\mathbf{R}) = \sqrt{\sum_{i=1}^{8} (F_{t_i} - F_{r_i})^2} \tag{5.22}$$

Where F_{t_i} and F_{r_i} are the frequencies of key i being pressed in the test and reference sequences.

Alexandre's second verification technique involves the use of the frequency of the appearance of key pairs, representing the correlation between each key. This data generates an 8×8 reference matrix. A similar Euclidian distance measure is used:

$$D_2(\mathbf{T},\mathbf{R}) = \sqrt{\sum_{i=1}^{8}\sum_{j=1}^{8} \left(F_{t_{(i,j)}} - F_{r_{(i,j)}} \right)^2} \tag{5.23}$$

Where $F_{t_{(i,j)}}$ and $F_{r_{(i,j)}}$ are the frequencies of occurrence of key i followed by key j in the test and reference patterns respectively.

Although the precise method of comparison and the acceptance threshold(s) are not stated, Alexandre claims that a verification algorithm based upon both distance measures, results in a false rejection rate of zero and a false acceptance rate of less than 1%. As Alexandre points out, however, not all of the key pairs provide the same power of discrimination, some are extremely variant and should not be taken into account during the verification process.

Another approach adopted is neural network based. Whilst the first two methods account for the first and second order similarities, a neural technique is investigated to recognise higher orders. Alexandre's neural approach is an implementation of the Backpropagation algorithm, with 64 (+bias) inputs, corresponding to the appearance frequency of each key pair. The network was tested with new patterns from the genuine users and other impostor patterns. With this method, zero false acceptance and false rejection errors were reported. Training required two seconds on a 50MHz 486 PC.

Alexandre's intention was to run his recognition algorithm on a smartcard, and as a results data and program memory requirements were considered. Data memory was estimated to be around 1324 bytes, and program requirements of around 2kbytes. Unfortunately, Alexandre quotes memory requirements for only his neural technique. No comparison is made with his structural verifiers. Nor is the execution time considered. Furthermore, it should be emphasised that these quoted requirements are for the trained verification network executing upon a smartcard. No attempt was made to train the network using a smartcard processor, and as a result, Alexandre's approach must rely upon external processing elements. As indicated in Chapter 1, such an approach exposes the possibility of tampering.

5.3.3 Literature Review – Conclusions

The approaches to capturing data of Umphress & Williams (1985), Leggett & Williams (1988), Brown & Rogers (1993), Monrose & Rubin (2000) and Robinson et al. (1998) are more realistic than those of Joyce & Gupta (1990) and Alexandre (1997). In capturing enrolment data during one session, then collecting test data

during a subsequent session, their approach reflects the likely real-world use of a biometric system in which enrolment will take place, followed by actual use of the system at some later time. Practical problems arise, however, when trying to coax a large number of people to participate in two separate sessions. It is clear from their reports that the above authors had some degree of control over their university's computing facilities; authors such as Robinson, were able to embed a transparent logging program into their computing network; Brown and Rogers had access to groups of students using their computer labs. These authors were hence able to capture data from a large group of people during two distinct sessions. Whilst capturing data from a small number of people during multiple sessions is indeed straightforward, logistical and practical obstacles hinder two session capture from a large group of users. As a result, data for both training and testing will be captured during a single session.

Garcia (1986) and Joyce & Gupta (1990) both suggested creating a feature vector of typing characteristics, allowing vector comparison techniques to be applied to the problem of keystroke verification. This is a succinct and elegant way of combining and summarising a person's characteristics into an easily processable form. The pressure sequence method offers three key discriminating characteristics; peak pressure level, pressure duration and inter-press durations. These characteristics will hereon be referred to as *pulse height*, *pulse width* and *interval duration*. Working with standard computer keyboards, the majority of authors reviewed above record keystroke latencies, some additionally extend their measurements to record key hold times. For a sequence of n keypresses, a feature vector of only keystroke latencies will have $(n-1)$ dimensions. If the vector is extended to include both keystroke latencies and key hold times the feature vector will then have $(2n-1)$ dimensions. Since the pressure sequence method records an additional characteristic for each tap – pressure, a feature vector comprising pulse height, pulse width and interval duration offers $(3n-1)$ components. Pulse width and interval duration are the equivalent of key hold and keystroke latency respectively. According to Robinson et al. (1998) a higher dimensionality may make for better discrimination. The desire to reduce verification string size led Brown & Rogers (1993), Obaidat & Sadoun (1997) and Robinson et al. (1998) to increase the dimensionality by recording both key hold times and keystroke latencies. It is however difficult to directly assess the effect of increased dimensionality since previous authors used

different verification methods and strings, captured a different number of enrolment and test sequences, and biased the acceptance thresholds for different reasons. Brown & Rogers (1993) suggests that key hold times offer better discrimination than keystroke latencies, but better still discrimination is found when both these characteristics are employed. Once a suitable verification algorithm has been discovered, the effects of combining pulse height, pulse width and interval duration will be assessed.

The verification methods used in early work by Gaines et al. (1980), Umphress & Williams (1985) and Leggett & Williams (1988) are not directly relevant. These methods make use of digraphs as the characterising features of a person's typing style. Although the proposed pressure sequence method uses a singular sensor, one idea from Umphress & Williams (1985) may have some use in pressure sequence verification. Umphress suggests that a sequence of characters be accepted if a certain proportion of digraphs pass a comparison test. Rather than digraphs, this method could be applied to the components of a pressure sequence, accepting a sequence if a certain proportion of components fall close enough to the reference template.

Some commonly used pattern recognition methods have been explored by Garcia (1986), Young & Hammond (1989), Joyce & Gupta (1990), Bleha & Obaidat (1993), Monrose & Rubin (2000) and Robinson et al. (1998). These are the ℓ_1 norm, Euclidean Distance (ℓ_2 norm), Mahalanobis Distance, Minimum Distance Measure, and non-linear classifiers. Although, as stated earlier, it is difficult to make direct comparisons between each method, used by different authors, a number of these methods have demonstrated good discrimination power. For example, the ℓ_1 norm, used in (Joyce & Gupta 1990), offered an equal error rate of around 3%, working with keystroke latencies in a user's first name, last name, username and password. Enrolment of each user is reported to require only eight sequences. Applying these recognition methods to verification requires that a suitable decision threshold be set. Bleha et al. (1990) employs a fixed threshold for all users, whilst Joyce & Gupta (1990), Brown & Rogers (1993) and Robinson et al. (1998) determine the threshold on a per user basis. Applying a fixed acceptance threshold for all users may lighten the enrolment processing load at the expense of verification accuracy, whilst determining a suitable threshold for each user may improve accuracy at the expense of enrolment processing requirements. Both approaches

will be explored in subsequent sections.

A number of authors report upon the improved discrimination potential of neural techniques over conventional recognition methods. Obaidat & Sadoun (1997) contrasts a number of neural techniques with conventional methods; he finds that the backpropagation algorithm performs best, offering 0% and 1% false recognition and false acceptance rates, respectively. He found this compared with a false recognition rate of 11% and 7% for a minimum distance classifier. Obiadat used an exceptionally large number of sequences for network training and testing; 15 valid users each provided 255 username samples per day over an eight week period, whilst 15 impostors each provided 15 impostor samples per username per day – a total of 255 impostor sequences per valid user per day, over the same time period. This data was split into training and testing halves. Brown & Rogers (1993), also compared the performance of a backpropagation algorithm with a minimum distance (Euclidean) measure. With an average training set of 37.7 valid user samples and 35 impostor samples and an average test set of 12.4 valid user samples and 30 impostor samples for testing, Brown reports false rejection rates of 17.3% and 20% for the neural and distance techniques, respectively. Decision surfaces were purposefully biased against false acceptances, resulting in zero false acceptances for both methods. The verification string in Brown's work was each person's name, rather than usernames, in Obaidat's work. Although the average string length of 15.6 characters in Brown's work is approximately double that of the 7 characters reported by Obaidat, Brown's reported performance of the backpropagation algorithm is clearly worse than stated in Obiadat. Allowing for differing threshold biases, the distance results from both papers seem comparable. This indicates very strongly that a large number of training samples is required for good neural performance. The number of samples captured by Obaidat is clearly impractical for a single session enrolment scheme.

The iterative nature of network training is likely to result in intensive processing requirements during the enrolment stage. Furthermore, network training requires the availability of a representative group of impostor samples, in order to calculate a decision boundary which separated the samples of a legitimate card-holder, and those of all others. Since enrolment should occur on-card (see Section 2.2.2), the use of a neural verifier necessitates the storage of impostor data on-card. This is clearly an undesirable requirement for the constrained memory capacity of a

smartcard, and as a result such techniques will not be considered further. Obaidat & Sadoun (1997) reports upon the use of a potential function recognition method, although this is not a neural network, it is iteratively weighted in a similar way to the perceptron algorithm.

Monrose & Rubin (1997) compares the performance of two statistical classifiers to an Euclidean distance measure. He finds that both weighted and non-weighted probability functions perform slightly better than the distance measure. These methods should also be assessed.

In conclusion, the commonly used methods of ℓ_1 norm, ℓ_2 norm (Euclidean Distance), Mahalanobis Distance and a minimum distance measure will be investigated, under both fixed thresholds and thresholds based upon the variability of each user. An implementation of the non-linear classifier, outlined in (Robinson et al. 1998) and a simple linear classifier based upon the linear deviation of each of a test vector's components, from mean reference components, will also be explored. The approach of (Umphress & Williams 1985), in which a test sequence is accepted if a fixed proportion of test vector components are sufficiently close to those of the reference, will be applied to these linear and non-linear classifiers.

5.4 The Pressure Sequence Method

With a comprehensive review of keystroke dynamics in place, this section describes a series of experiments to assess the discrimination potential of finger tapping upon a pressure sensor. This approach has been dubbed '*pressure sequence*', and this term is used from here on. We begin with a description of the experimental method and apparatus used to capture sequences of taps from a volunteer population of individuals. Once data has been captured, the focus of the remainder of the chapter is to investigate the verification potential of the methods of keystroke dynamics.

5.4.1 Experimental Apparatus

To capture pressure sequences, a piezoelectric sensor whose construction is described in detail in Section 3.5.1 was bonded onto a smartcard blank in the manner

of Section 4.9. Recall that this method involves the direct screen printing of the PTF sensor layers directly onto the card, resulting in improved sensitivity of the sensor, when the card is placed upon flat surfaces.

The signal conditioning circuitry is that of the charge amplifier, as described in section B.2. The data capture electronics are, however, somewhat different to those described in the previous chapter.

As indicated in Section 5.3.3, capturing sequences from users on multiple occasions, rather than during one singular session represents more realistically the manner in which users would interact with a so-called real-world system. As also indicated previously, capturing data from each participant during a number of separate sessions, is difficult. Unlike the capture of keystroke dynamics data which lends itself to observation of participants within their usual computing environment (see for example (Robinson et al. 1998) (Brown & Rogers 1993) or (Monrose & Rubin 2000)), the use of a prototype sensor-card in this series of experiments precludes such transparency.

In an attempt to improve matters, data acquisition electronics were designed, for portability, and a prototype built. This should have offered a means of visiting participants at their convenience, rather than requiring them to present themselves at the author's laboratory on more than one occasion. It was believed that whilst volunteers may be willing to be inconvenienced in this manner for a single session, they could not be expected, nor would be happy, to do this more than once.

Unfortunately, due to unforseen logistical circumstance, the prototype electronics could not be manufactured into the desired portable robust unit, within an acceptable time period. As a result data capture sessions had to be conducted within the controlled environment of the laboratory. This in turn biased the data collection method towards single session thereby maximizing the number of participants.

The data acquisition hardware's function is the digitization of voltage signals from the sensor's charge amplifier, and the communication of this data to PC. It was based around Microchip's 8-bit 16F84 microcontroller[2], whose software was compiled using Hi-Tec's C cross compiler (v7.83)[3] and downloaded to chip with Microchip's suite of tools; MPLAB (v4.0). PC communication software was written

[2]Microchip Technology Inc. Chandler, Arizona, USA Web: `http://www.microchip.com`
[3]HI-TECH Software, Alderley, QLD, AUSTRALIA. Web: `http://www.hitech.com.au/`

using Microsoft's Visual Basic (v5.0)[4].

Figure 5.1 provides an overview of the circuit, whilst appendix D provides a detailed circuit diagram, and description of the enabling software.

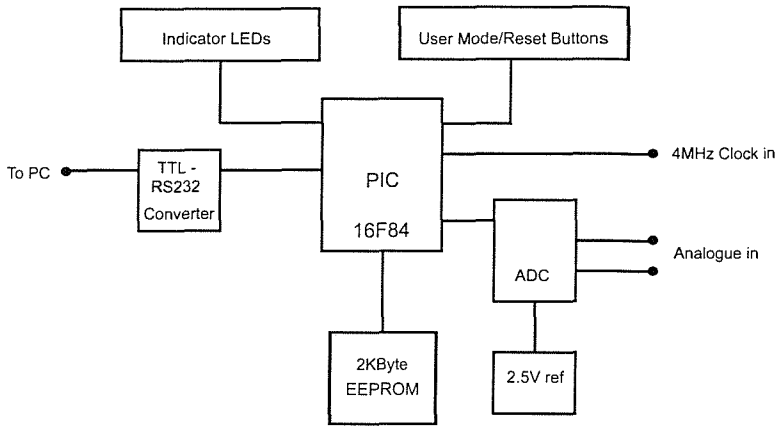


Figure 5.1: Data Acquisition Circuit Overview: Appendix D, Figure D1 provides the circuit diagram.

An 8-bit 16F84 Peripheral Interface Controller (PIC) from Microchip controls the logging device. Analogue to digital conversion is performed by a 12-bit successive approximation ADC (Linear Technology[5] LTC1285), offering one differential input, sampled at up to 7.5kHz. Voltage reference to the ADC was provided by a precision 2.5V bandgap voltage reference (MC1403) from STMicroelectronics[6]. According to data sheets, this device is stable to $\pm 0.1\%$. Linear Technology's data sheets for the ADC states a maximum conversion error of $\pm 0.2\%$ of full-scale (± 8 LSB).

The communication protocol between PIC and ADC occurs across three wires and is implemented in software. Data from the ADC is temporarily held in the PICs data memory, from whence it will be transmitted across a serial link to PC.

PIC to PC serial communication is implemented in software. Signal voltages, however must be converted from TTL levels to RS-232 levels to and from the PC. This function is performed by a MAX-232 IC from MAXIM[7].

LEDs are provided to indicate the logger's current status, these are driven directly

[4]Microsoft Corporation. Redmond, WA USA. Web: `http://www.microsoft.com`
[5]Linear Technology Corporation, , Milpitas, CA, USA. Web: `http://www.linear-tech.com`
[6]STMicroelectronics, Saint Genis, France. Web: `http://www.st.com`
[7]Maxim Integrated Products, Inc. Sunnyvale, CA, USA. Web: `http://www.maxim-ic.com`

from the PIC, whose output pins are capable of high current sourcing. Input buttons enable the selection of operational mode, or to reset the device.

Power is supplied to the circuit by a 5V regulator (LM2930) from National Semiconductor[8] and a standard 4MHz crystal provides the clock signal.

Appendix D provides a detailed description of the data acquisition circuit hardware and software.

5.4.2 Experimental Method

Figure 5.2 gives an overview of the experimental set-up.

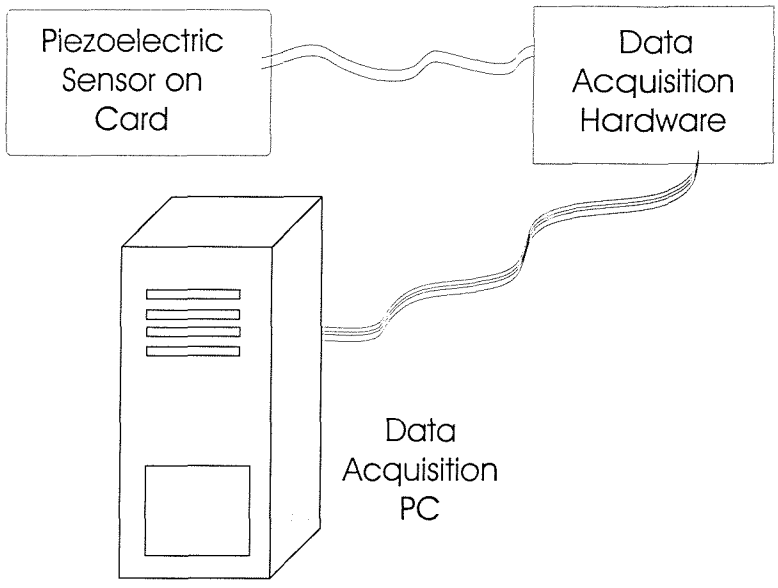
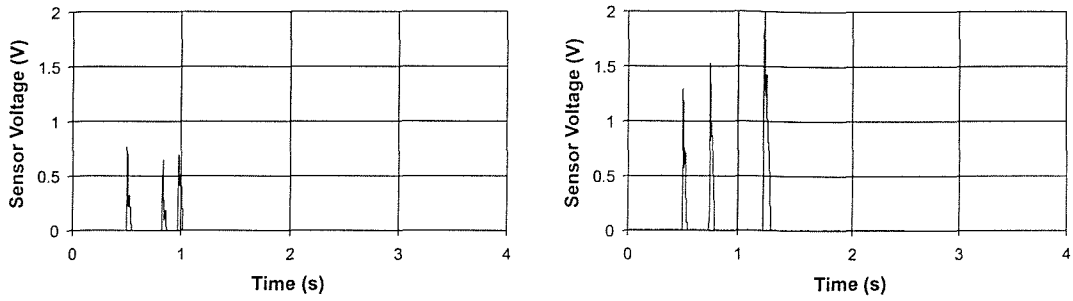


Figure 5.2: Experimental Set-up

To validate this approach to identity verification, students and staff from the Electronics & Computer Science department were invited to participate in a trial, and 34 people volunteered. The population is therefore self-selected, rather than randomly chosen. Within the experimental population there exists a reasonable variability in age and sex. Although a larger group of people would have been desirable, its size compares favourably with a number of the studies outlined above.

[8]National Semiconductor Corporation, Santa Clara, CA, USA. Web: `http://www.national.com`



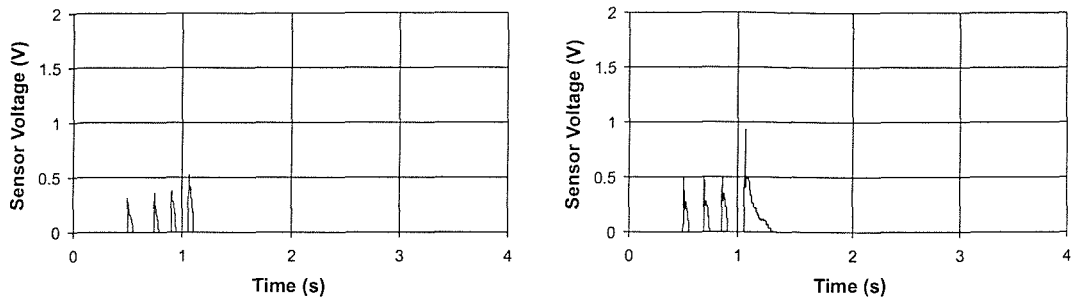
(a) Participant *A* (b) Participant *B*

Figure 5.3: 3-Pulse Pressure Sequences

Each volunteer was asked to choose a short tapping sequence (typically lasting between 2 and 4 seconds), and to tap that rhythm 30 times, in three sets of 10 sequences. In the first set the card was held in one hand whilst tapping with the other. In the second sequence, the card was placed upon a table, and in the third set, a mouse mat was placed between card and table. These three scenarios were thought to be representative of the way in which this system would be used in the real world, and remove some experimental bias. The volunteers were not given immediate feedback on how they were doing. Instead, they were asked simply to concentrate on tapping the rhythm.

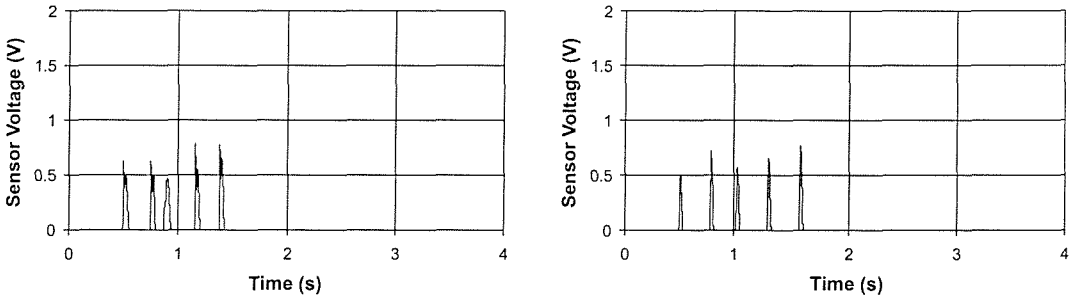
Figures 5.3 through 5.10 show examples of unprocessed pressure sequences with between 3 to 10 taps (or pulses). Each figure comprises of sequences from two participants, to help illustrate some differences between sequences. Unlike previous analysis of piezoelectric sensor response, the sensor outputs are given in terms of voltage rather than charge. This is simply because in the latter stages of this chapter, the unprocessed digital voltage levels (from the acquisition hardware) will be used to construct feature vectors representing each pressure sequence. Since the voltage output from the sensor signal conditioning circuit (see section B.2) is proportional to the charge generated, the conversion between voltage and charge is felt to be an unnecessary step.

As can be seen from figures 5.3–5.10, there exist obvious differences in the number of pulses, the pulse heights, pulse widths and inter-pulse (interval) lengths, both within and between sequences. It is with these simple easily extractable charac-



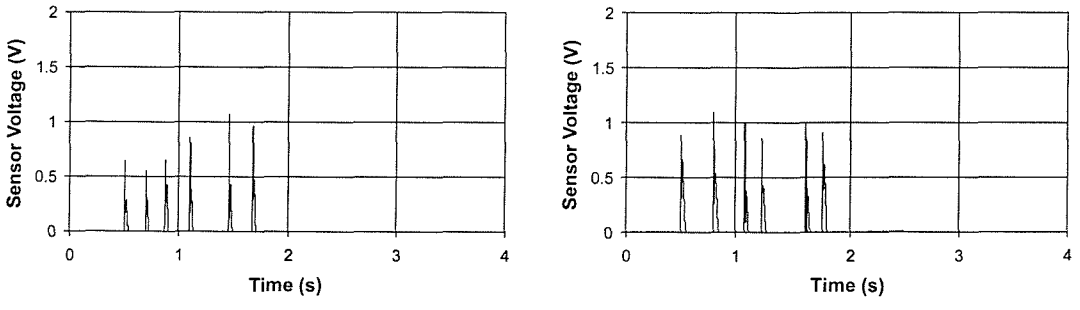
(a) Participant C (b) Participant D

Figure 5.4: 4-Pulse Pressure Sequences



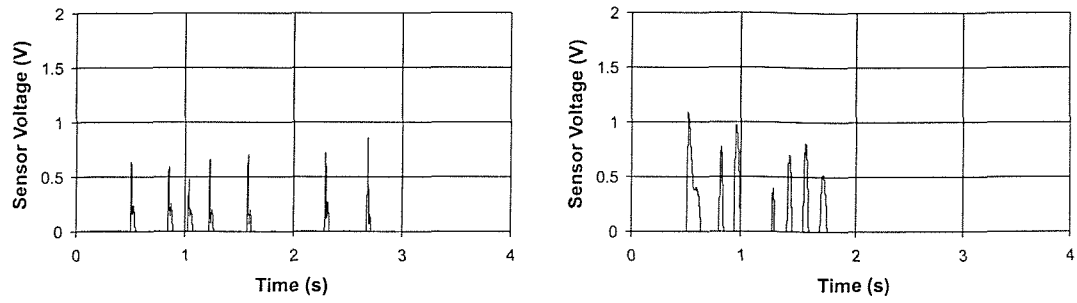
(a) Participant E (b) Participant F

Figure 5.5: 5-Pulse Pressure Sequences



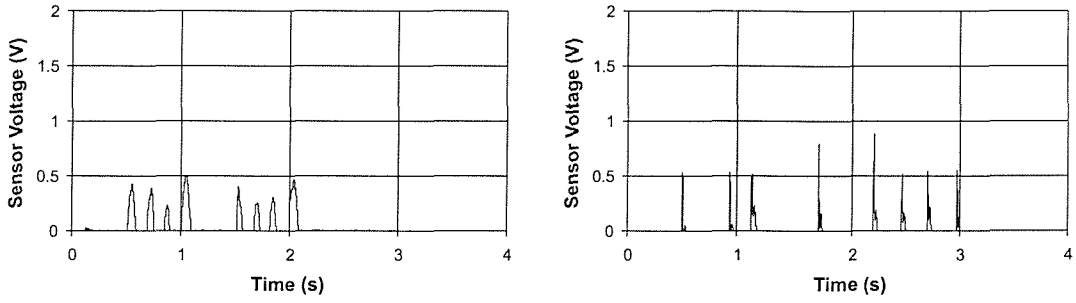
(a) Participant G (b) Participant H

Figure 5.6: 6-Pulse Pressure Sequences



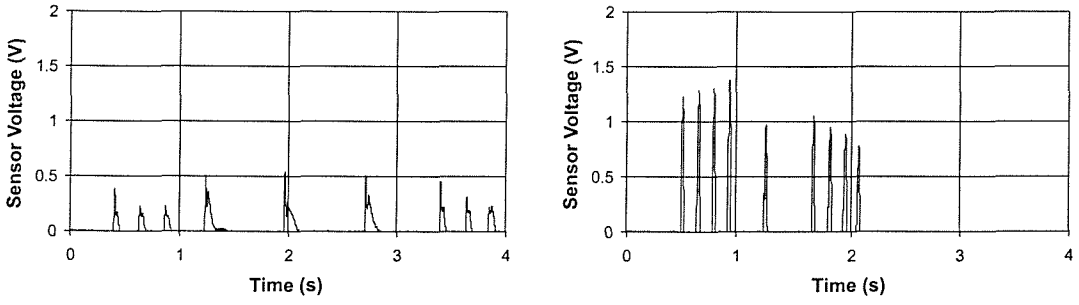
(a) Participant I (b) Participant J

Figure 5.7: 7-Pulse Pressure Sequences



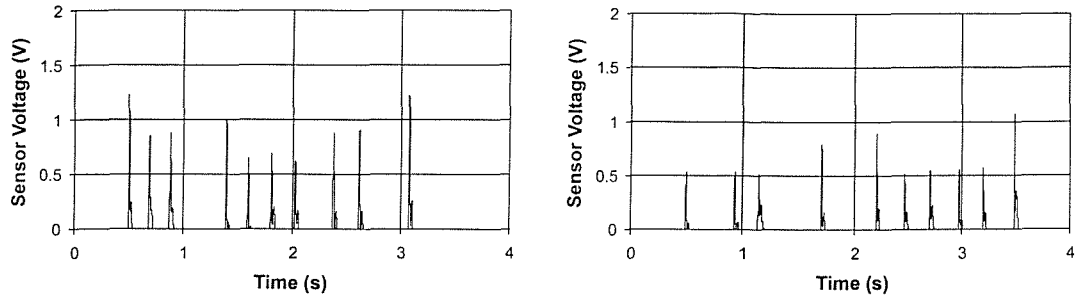
(a) Participant K (b) Participant L

Figure 5.8: 8-Pulse Pressure Sequences



(a) Participant M (b) Participant N

Figure 5.9: 9-Pulse Pressure Sequences



(a) Participant *O* (b) Participant *P*

Figure 5.10: 10-Pulse Pressure Sequences

teristics that the discrimination methods, considered in this chapter, operate, and with which the pressure sequence method aims to discriminate between a valid and an invalid user.

5.4.3 Feature Extraction

Before comparing two pressure sequences, the key characteristics; pulse height, pulse width and interval duration must be extracted from each sequence of raw, unprocessed data. This was performed using a simple Visual Basic program (See Appendix E for details). It seems that sensor responses do not have a standard and uniform shape, rather the shape varies between participants. This is illustrated, by considering individual pulses from the above raw data sequences. Figure 5.11 shows the third pulse of Figure 5.7(a) (generated by participant I), alongside the fourth pulse of Figure 5.8(a) (generated by participant K). These specific pulses were chosen simply because they are of comparable magnitude. They should be considered as illustration, only. Chapter 7 considers pulse shape in greater detail.

Pulses from participant 'I' rise and fall quickly, before rising again for a final time. Pulses from participant 'K', on the other hand, typically spend a longer period of time reaching their peak level, before falling at a lesser rate.

Considering all pulses collected during this experiment, Figure 5.12 shows the height for each pulse plotted against its width. It can be seen (from Figure 5.12

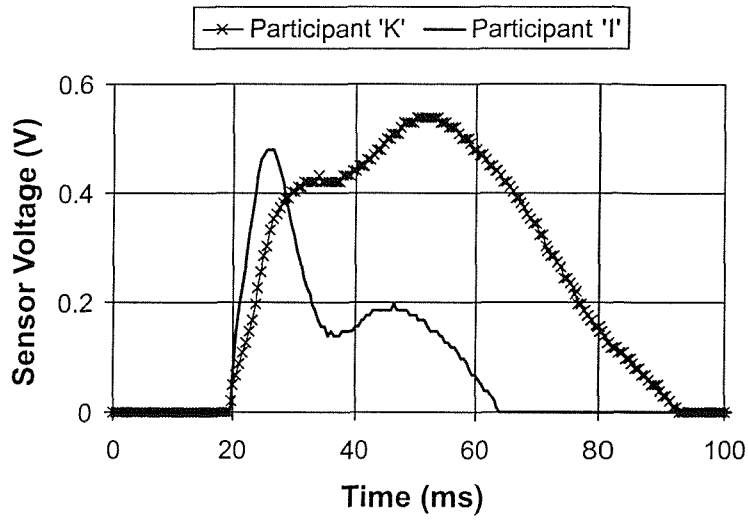


Figure 5.11: Pulses from two Participants

that although there is no dominant correlation between pulse height and pulse width, most pulses have a duration of less than 100ms and a height of less than 1.5V. Beyond a peak of 1.5V, pulses tend to have a duration of around 50ms, implying very short sharp taps upon the sensor. Pulse characteristics appear to be function of user interaction with the sensor and may offer additional discrimination characteristics between participants. However in this work, pulses are quantified only in terms of their height and width.

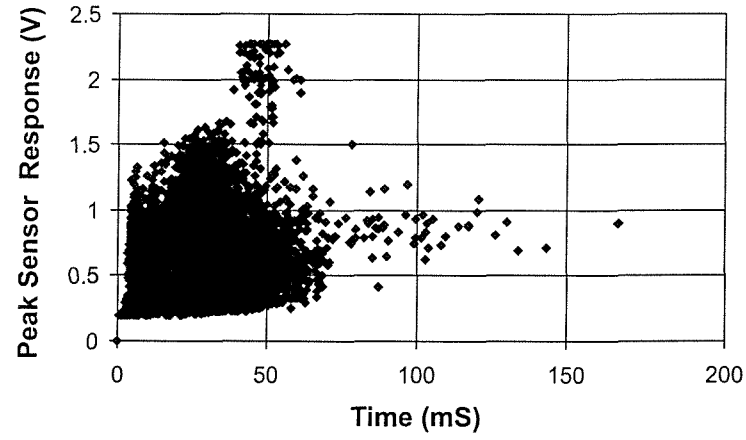


Figure 5.12: Pulse Width Vs Pulse Height (All Pulses)

Extracted pulse heights, pulse widths and interval durations, were then packaged

into a single column vector, of $(3n - 1)$ dimensions, where n is the number of pulses in a sequence. Appendix E provides more detail of the feature extraction algorithm used.

5.5 Capturing Impostor Sequences

Ten volunteers were asked to provide impostor data for this experiment. They had no direct experience of the sequences entered by the other users, and were merely guided in their masquerading. Since a pressure sequence consists of a distinct number of taps which could be sequentially guessed, the impostors were encouraged to begin by providing a number of sequences with two taps, then three, working up to sequences of 14 taps. It was pointed out to the impostor volunteers that a verifier would use pulse height, pulse width and interval duration as its recognition features. They were thus encouraged to generate sequences with variation in these aspects, to try and generate sequences that would be accepted as originating from an enrolled user. Each impostor provided between five and ten sequences per pulse number, giving a total of 992 impostor sequences. The distribution of impostor sequences with pulse number is shown in Figure 5.13.

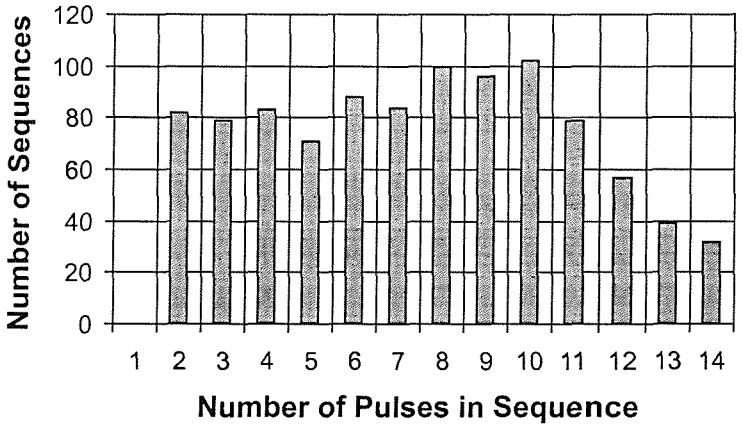


Figure 5.13: Impostor Sequence Distribution

The impostors were not given any feedback about their progress, nor their sequences' proximity to those of enrolled users. Impostor sequences were merely

captured and stored for future use. This approach misses out on an important aspect of real-world use, namely that the real-world impostor has something tangible to gain from his masquerading – these impostors had no such incentive. Indeed some of the volunteers began to show signs of boredom and frustration towards the end of their session.

The gaming aspect of breaking the sequences of colleagues and friends, as pointed out by Bleha & Obaidat (1993), may provide an impetus for impostor perseverance. This is highlighted as appropriate for future work.

5.6 Considering Reference Vectors

5.6.1 The Effect of Generating Sequential Sequences

Before considering how to compute reference vectors, this small experiment investigates the properties of sequences which were generated in rapid, back-to-back, succession. The experiment involved eight volunteers, each of whom were asked to tap out their pressure sequence 30 times one after the other. As before, pressure sequences will be quantified in terms of their average pulse height and average total duration. Figures 5.14 and 5.15 show average pulse heights for each of the 30 consecutive sequences normalised to the geometrical mean of all sequences, and average sequence duration, normalised to the geometrical mean for all sequences.

As seen in Figure 5.14, pulse heights demonstrate a convincing upwards trend with sequence number. The first two pulse height averages are 24% and 16% below the mean for all sequences, and the last two heights are 15% and 20% above the mean. Sequence length, shown in Figure 5.15 demonstrates a very gradual decrease with sequence number. The linear trendline begins 1.5% above the mean, and finishes 1.5% below the mean. This is far from conclusive as the standard deviation of sequence lengths was calculated to be approximately 4%, and could easily mask any effect of producing consecutive sequences.

The 15% increase in pulse heights observed within the main experimental data set can readily be explained by the effect of users tapping a number of sequences in quick succession. This is probably due to users becoming more familiar with

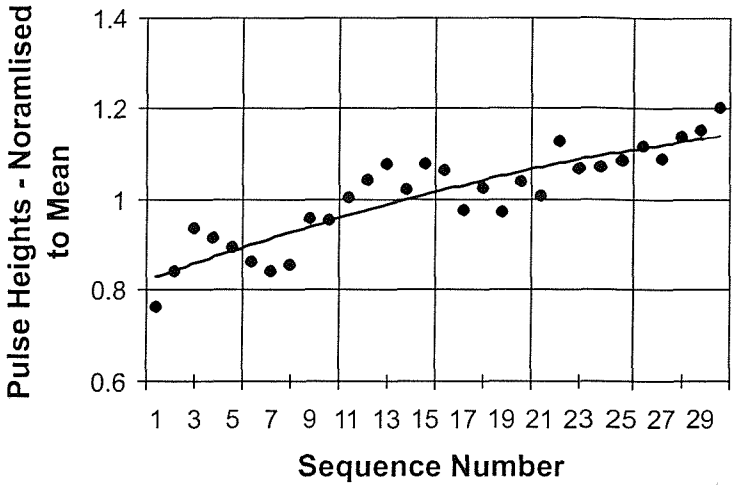


Figure 5.14: Mean Pulse Heights – Normalised to Mean of Sequences

the experimental apparatus and becoming less concerned with its fragility. It is possible that splitting the main capture session into three distinct regimes, lowered this effect from the 40% range observed during the capture of 30 consecutive sequences. Average sequence lengths appear to decrease with sequence number in the main experimental data set, which superficially may be due to user boredom, or again an increased familiarity with the apparatus. The consecutive sequences data set, however, does not support this.

5.6.2 User Consistency

The most obvious layer of discrimination is the number of pulses in a sequence. Testing the consistency with which a user taps sequences is an indication of how well suited the pressure sequence method is to that user. Initial analysis of data entered from the 34 participants is to compute the modal number of pulses within each person's sequences (defined as the *modal pulse number* from here on). It is assumed that the most common number of pulses in sequences from each user is their intended sequence. Figure 5.16 shows the proportion of user sequences entered with their modal number of pulses.

This shows that approximately 85% of people entered their modal number of pulses, more than 80% of the time, with all but one person entering sequences

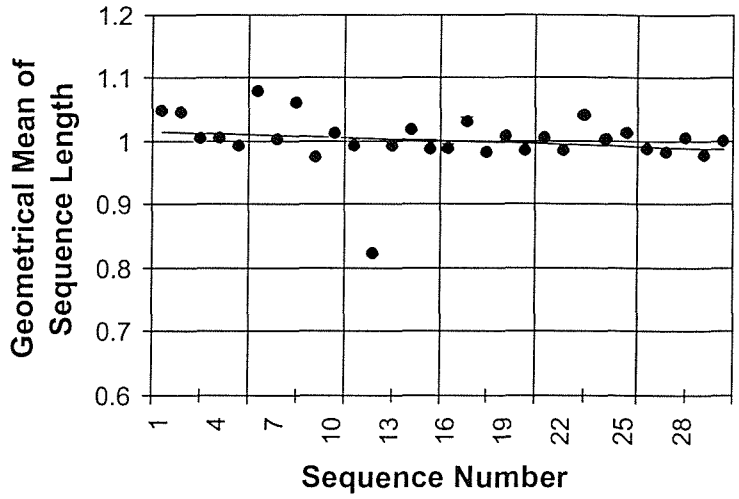


Figure 5.15: Mean Sequence Lengths – Normalised to Mean of Sequences

with modal pulse numbers greater than 60% of the time. The one person who failed to enter sequences consistently, performed particularly badly, entering his most common number of pulses only 17% of the time. Consistency at this stage is a good test of how well suited this verification method is to each person. Clearly, the user for whom entering sequences consistently was a problem would, given a choice, not choose this approach to verification. Figure 5.16 demonstrates that the majority of users were able to enter their sequences with reasonable consistency.

Figure 5.17 shows the distribution of modal pulse numbers. The most frequent modal number of pulses is 7, and the inconsistent participant being the only person entering sequences with 16 pulses. Sixteen is the highest modal pulse number from all participants, and it is believed that this user intentionally generated complex, copy-resistant sequences, making replication difficult even for himself. The next highest number of pulses, 14, was entered by two other users, both tapping 14 pulse sequences 87% of the time. Figure 5.18 shows consistency plotted against modal pulse number. From this it can be seen that the inconsistent user suffered atypical problems, and it is clear that that this participant is not well suited to the pressure sequence approach.

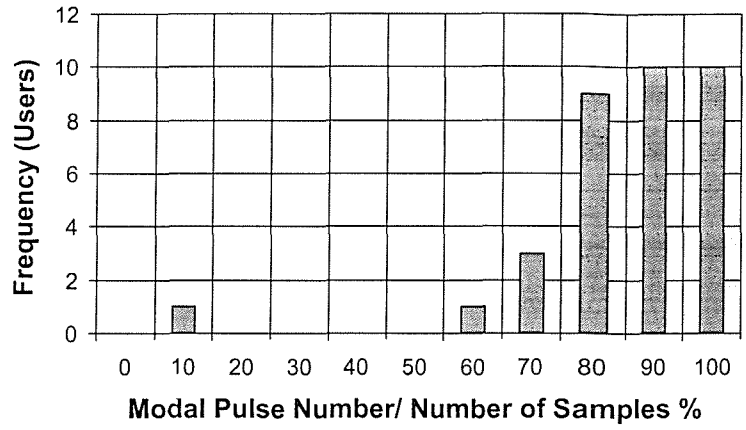


Figure 5.16: Participant Consistency: Histogram shows the distribution with which participants enter sequences containing their modal number of pulses.

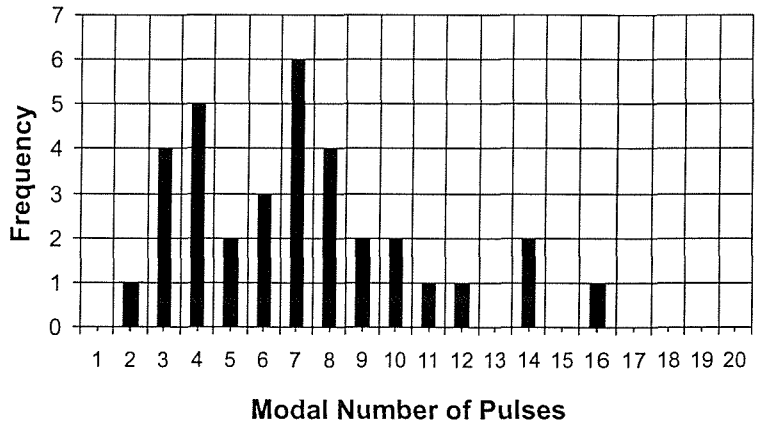


Figure 5.17: Distribution of Modal Pulse Numbers

5.6.3 Number of Enrolment Sequences

The trends in pulse height and sequence length with sequence number, described in Section 5.6.1 suggest that as few sequences as possible should be used in the generation of a reference profile. However, a sufficient number are required to confidently generate a mean and standard deviation for each component. It was decided that, in common with Joyce & Gupta (1990), eight enrolment sequences would suffice. Since all data was collected in one single session, and there is a requirement for two sets of distinct training and testing data, captured sequences from each user were alternately placed into training then testing directories. Ref-

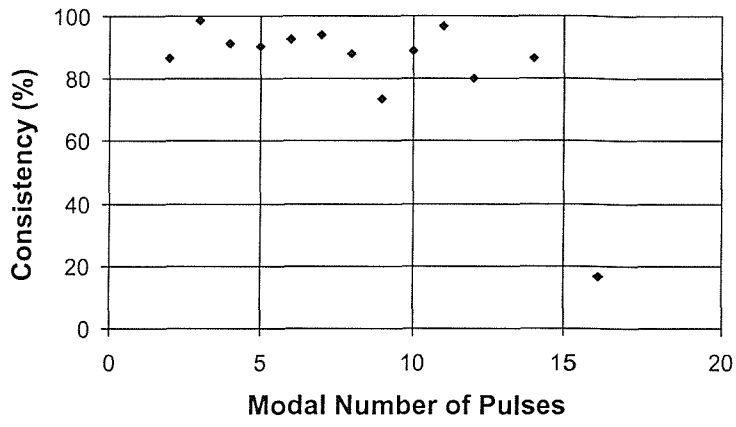


Figure 5.18: Consistency with Modal Pulse Number

erence profiles were generated entirely from the training set.

5.6.4 Outliers

A number of authors (Joyce & Gupta 1990), (Bleha et al. 1990) advocate the removal of outlier components. Upon inspection of captured sequences it has been decided not to remove outlying data components; it appears that users with outlying components in their sequences express a naturally high degree of variance. Outliers, therefore form part of their sequence characteristics, and will be retained.

5.7 Verification Functions

The structural pattern recognition methods outlined in section 5.3 will be investigated in this section. The methods to be used here are the ℓ_1 norm, the ℓ_2 norm, a minimum inter-class distance classifier and a classifier based around the Mahalanobis distance. In addition, two component-wise linear and non-linear methods are considered.

Since the aim of this project is verification, rather than identification, the reference vector from each participant is subjected to all impostor sequences, through a sweep of acceptance thresholds, rather than comparing each test sequence to all

user reference vectors. Test sequences are deemed to have originated from the user under test if they fall within certain threshold limits. Since the pressure sequence method requires that each user enters a certain number of taps for their sequence, and the number of taps in a sequence can be sequentially guessed, impostor vectors will only be applied to those user vectors with the same number of taps. To do otherwise would artificially lower false acceptance rates.

As indicated in Section 5.3, there are two possibilities for determining thresholds: Either setting a fixed acceptance threshold for all participants, irrespective of their natural variance, or to take account of each participant's variations in entering sequences. This can be performed using the approach of Joyce & Gupta (1990), in which deviations between enrolment signatures and each person's reference vector are used to calculate a mean difference and standard deviation, on a per participant basis. This approach is covered in detail in Section 5.3.

Although the discriminating power of each verifier is important, it is crucial to compare verifiers not only in terms of their error rates, but also in terms of processing requirements. These verifiers have been implemented in MatLab (V5.3)[9] which offers the facility to count the number of floating point operations (FLOPS) between two points in an algorithm. Whilst the task of processing is affected by a number of factors, such as memory access, processing architecture, instruction set, for example, it is felt that by comparing the number of operations for each verifier on a single PC platform, an early indication of the comparative requirements on a smartcard can be inferred. Smartcard computational constraints are more fully considered in Chapter 6.

A skeletal MatLab program was written to take care of common tasks such as file handling and the calculation of false rejection and false acceptance rates – Appendix F provides the MatLab code. Appropriate enrolment and verification functions are then called as required.

5.7.1 The ℓ_1 norm Verifier

As stated in Section 5.3, the ℓ_1 norm was used by Joyce & Gupta (1990) to verify identities based on the inter-key times of a user's first and last names, their

[9]MatLab from The MathWorks, Inc. Natick, MA, USA. Web: `http://www.mathworks.com`

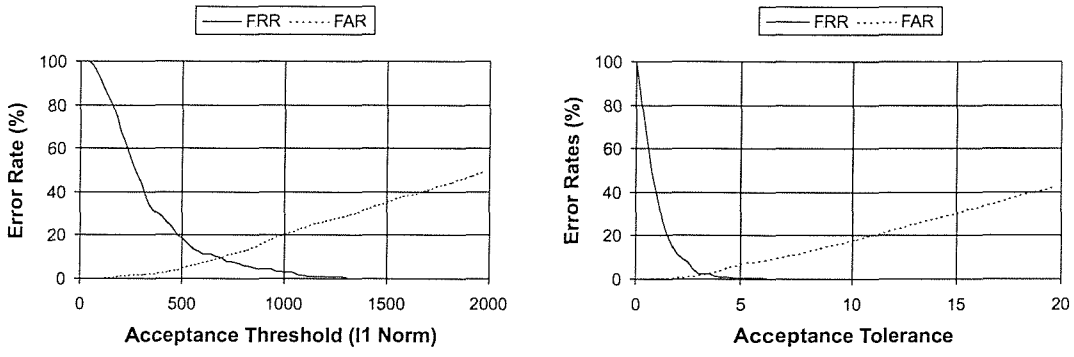
username and password. Joyce reports an equal error rate of approximately 3% (by visual inspection of his published results). Equation (5.4) is the implementation used to quantify the similarity between a test vector, T and a reference vector, R.

Using the same fixed acceptance thresholds for all users, a test vector will be accepted if:

$$\sum_{i=1}^{d} |r_i - t_i| < \theta \tag{5.24}$$

where θ is the acceptance threshold, and all other variables are as defined in Section 5.3.

Figure 5.19(a) shows the false acceptance and false rejection rates, calculated for a sweep of fixed thresholds. The equal error rate using this method was calculated to



(a) Fixed Acceptance Threshold

(b) User-Specific Threshold

Figure 5.19: Error Rates using the ℓ_1 norm Verifier

be 9.7% at an acceptance threshold of 685. The number of FLOPS for enrolment and verification were measured to be 175 and 24 FLOPS, respectively.

Setting the acceptance threshold on a per user basis, as outlined in Section 5.3, a test vector, T, will be accepted if it is sufficiently close to a reference, R, such that

$$\sum_{i=1}^{d} |r_i - t_i| < |\overline{D}_{Enrolment} - \tau\sigma_{Enrolment}| \tag{5.25}$$

where $\overline{D}_{Enrolment}$ is the mean distance of the enrolment vectors from the reference vector, $\sigma_{Enrolment}$ is the standard deviation of the enrolment vector distance from

the reference vector, and τ is defined as the acceptance tolerance. Figure 5.19(b) shows the resulting error characteristics.

Using the mean enrolment distance and standard deviation to determine acceptance thresholds considerably reduces the equal error rate for this verifier, with a rate of 2.3% compared with 9.7% for the ℓ_1 norm using fixed thresholds. As expected the processing requirements have increased, with the enrolment process requiring an average of 378 FLOPS against 175 for the fixed threshold method. Verification requirements have increased marginally to 29 FLOPS compared with 24 for the fixed threshold method.

Figure 5.20 shows the receiver operating characteristics (ROC) for both fixed and user dependent threshold verifiers. The user dependant curve offers a reduction in false acceptance rates, without having to tolerate a great increase in false rejections. For example, if the rate of false acceptance were reduced to around 1%, the user defined acceptance threshold method would suffer an increase of user rejections to around 10%. The fixed threshold method, however, would have to tolerate an increase in false rejections to around 80%, for a reduced false acceptance of 1%.

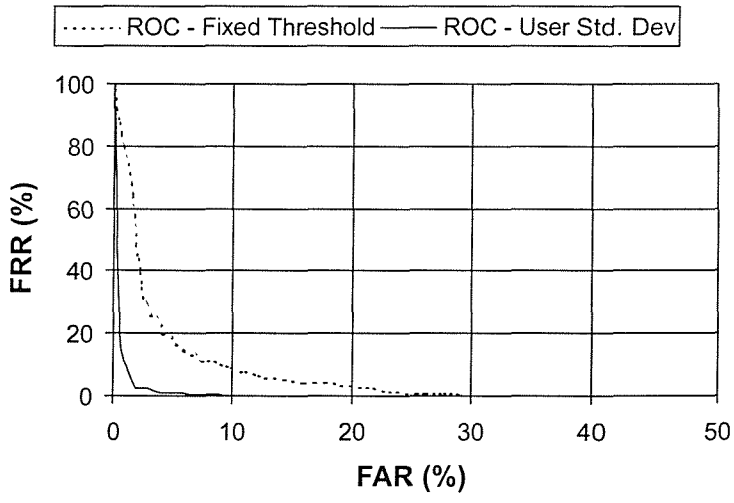


Figure 5.20: ROC curves for the ℓ_1 norm Verifiers

5.7.2 The ℓ_2 norm Verifier

From the literature survey at the beginning of this chapter, it was noted that the ℓ_2 norm (as stated in equation (5.2)), or Euclidean distance has been used, or proposed by a number of authors. For example Young & Hammond (1989) proposed using this method on a plurality of features although he reports no results. Brown & Rogers (1993) reports false rejection rates of 20% with 0 false acceptances, using the Euclidean distance measure on keystroke latency and key hold features extracted from user's names. Whilst Monrose & Rubin (1997) obtained a correct identification rate of 83% using a Euclidean distance measure upon keystroke durations and digraph times from a few sentences of text.

Under the fixed threshold implementation of the ℓ_2 norm verifier a test vector, $\mathbf{T}$ if it is sufficiently close to a participant's reference vector, $\mathbf{R}$, such that

$$\sqrt{\sum_{i=1}^{d}(r_i - t_i)^2} < \theta \tag{5.26}$$

Setting an acceptance threshold on a per-user basis, in the manner of (5.25), a test vector will be accepted as having originated from same participant with whom reference vector $\mathbf{R}$ is associated if

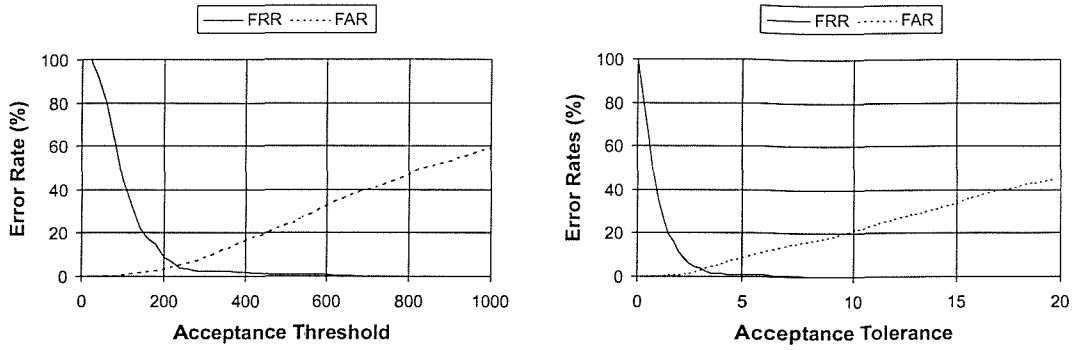
$$\sqrt{\sum_{i=1}^{d}(r_i - t_i)^2} < |\overline{D}_{Enrolment} - \tau\sigma_{Enrolment}| \tag{5.27}$$

where nomenclature is consistent with that defined in (5.25).

Figures 5.21(a) and 5.21(b) show the error characteristics of both methods with respect to threshold and tolerance.

Equal error rates for the fixed acceptance threshold and user specific threshold methods are 5% and 3%, respectively. Enrolment and verification requirements are 175 and 99 FLOPS for the fixed threshold method and 1000 and 104 FLOPS, respectively, for the user specific threshold regime. Figure 5.22 shows the ROC curves for both methods.

Setting both verifiers to have false acceptance rates of 1% then the false rejection



(a) Fixed Acceptance Threshold (b) User-Specific Threshold

Figure 5.21: Error Rates using the ℓ_2 norm Verifier

rates of the fixed threshold and user specific threshold methods would be 41% and 18%, respectively. This demonstrates a tightening of the spread exhibited by both threshold methods using the ℓ_1 norm as verifier.

Bleha & Obaidat (1993) uses a derivative of the ℓ_2 norm in his minimum distance classifier, which is essentially the square of the ℓ_2 norm. By squaring the norm, the computational process avoids the square root requirement and thus may be expected to furnish similar discrimination whilst perhaps reducing computational requirements. Using keystroke latencies within user's names, Bleha reports a false acceptance rate of 2.8% and false rejection rate of 8.1%. Although Bleha normalises his classifier, allowing name samples of differing lengths to be compared, filtering pressure sequences for number of pulses eliminates this requirement here. The verification implementations are given in (5.28) and (5.29).

$$(\mathbf{T}-\mathbf{R})^*\,(\mathbf{T}-\mathbf{R}) < \theta \tag{5.28}$$

$$(\mathbf{T}-\mathbf{R})^*\,(\mathbf{T}-\mathbf{R}) < |\overline{D}_{Enrolment} - \tau\sigma_{Enrolment}|. \tag{5.29}$$

The receiver operating characteristics for (5.28) and (5.29) are presented in Figure 5.23.

Equal Error rates are computed as 5.2% and 2.8% for the fixed user specific threshold regimes, respectively. Enrolment requirements are recorded as 175 and 844 FLOPS for fixed and user specific thresholds, whilst verification requirements are

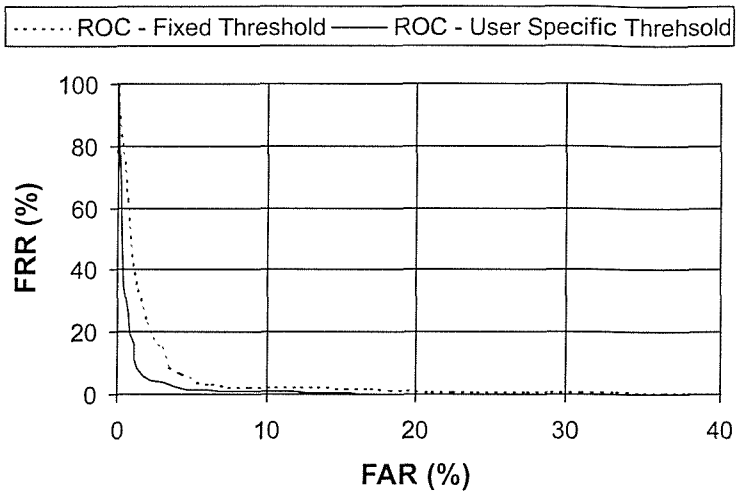


Figure 5.22: ROC curves for the ℓ_2 norm Verifiers

80 and 85 FLOPS, respectively. Error rates for the fixed threshold regime are comparable with the explicit ℓ_2 norm verifier – 5.2% against 5%, with this method offering marginally lower verification demands – 80 against 99 FLOPS. The user specific regime offers both improved error rates (2.8% against 3.5%) and processing demands (844 Enrolment FLOPS against 1000 and 85 verification FLOPS against 104), compared with the explicit ℓ_2 norm method.

At 1% false acceptance rate the fixed threshold regime offers 37% false rejection rate, whilst the user specific threshold method provides 11% false rejection rate.

5.7.3 The Mahalanobis Distance Verifier

Garcia (1986) reports upon the performance of a Mahalanobis verifier applied to keystroke latencies within a person's typed name. Capturing each person's name a number of times, Garcia biases his verifier against false acceptances, and quotes a false acceptance rate of 0.01%. The penalty for such a low rate of false acceptance is a higher false rejection, which he quotes as 50%. Robinson et al. (1998) use the Mahalanobis distance to discriminate between users based upon keystroke latencies and hold times contained within an individual's username. A false acceptance rate of 24% and a false rejection rate of 23% are reported from several hundred login attempts.

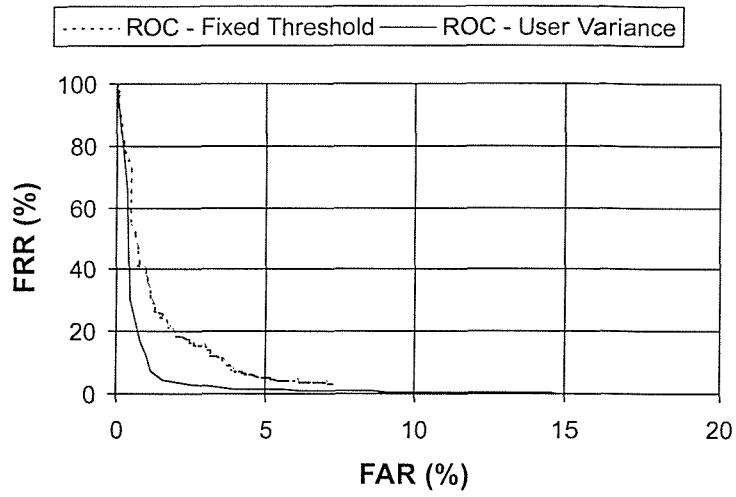


Figure 5.23: ROC curves for the ℓ_2 norm Squared Verifiers

The Mahalanobis Distance is defined in equation (5.1). Simple linear classifiers, such as the ℓ_1 and ℓ_2 norms, dealt with above, may suffer from limitations in the way in which they generate decision boundaries. They are known to be inaccurate when dealing with features which are highly correlated or badly scaled (Schalkoff 1992). The Mahalanobis distance provides a way around some of the limitations of simple classifiers. By taking the covariance matrix into account, the classifier can be made to account for poor scaling and deal with correlations between features. However, calculating the covariance matrix requires that a *relatively* large number of example vectors is available (generally d+1 example vectors for a d-dimensional feature vector). Robinson et al. (1998) had *several hundred* example vectors available per individual, whilst Garcia (1986) states that each person entered their name a *number of times*. In this experiment only eight enrolment vectors are available to generate a reference vector. Even for small feature vectors, generated from sequences containing only a few pulses, the number of features will be $(3n - 1)$, where n is the number of pulses, very quickly the number of features rises above eight, the number of example vectors available.

Indeed, this is demonstrated when one calculates a covariance matrix using sequence vectors generated from sequences containing only a modest number of pulses; the determinant of the resulting covariance matrix tends towards zero, indicating that the matrix is close to being singular.

Rather than use a badly scaled covariance matrix to compute the Mahalanobis distance, the leading diagonal which contains only the variances of each component, can be used. Whilst the resulting metric will not be rotationally invariant, it will exhibit scale invariance.

This method forms the subject of the next verifier. A test vector, $\mathbf{T}$ will be accepted under the fixed acceptance threshold regime, if:

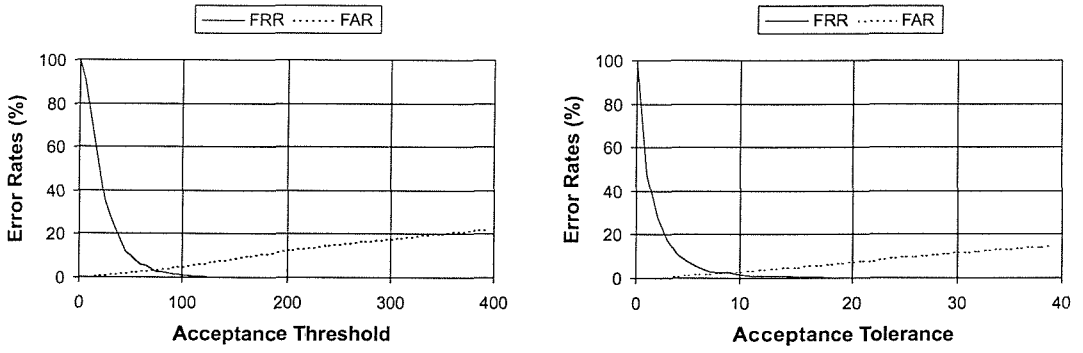
$$(\mathbf{T}-\mathbf{R})^{*}\,\mathbf{V}^{-1}\,(\mathbf{T}-\mathbf{R}) < \theta \tag{5.30}$$

where $\mathbf{V}$ is the leading diagonal of the covariance matrix generated by the enrolment vectors, whose mean is the reference vector, $\mathbf{R}$. Explicitly, $\mathbf{V}^{-1}$ is the inverse of a $d \times d$ matrix, whose non-zero components are on the leading diagonal and comprise the variances of each of the d-dimensional enrolment vectors' components.

Similarly, $\mathbf{T}$ will be accepted under the user specific acceptance threshold regime, if:

$$(\mathbf{T}-\mathbf{R})^{*}\,\mathbf{V}^{-1}\,(\mathbf{T}-\mathbf{R}) < |\overline{D}_{Enrolment} - \tau\sigma_{Enrolment}| \tag{5.31}$$

Figures 5.24(a) and 5.24(b) show the false acceptance and false rejection curves for fixed and user specific threshold methods, respectively.



(a) Fixed Acceptance Threshold (b) User-Specific Threshold

Figure 5.24: Error Rates using Modified Mahalanobis Distance

Equal error rates for this modified Mahalanobis measure were found to be 3.4% and 2.4% for the fixed and user-specific acceptance threshold approaches. There

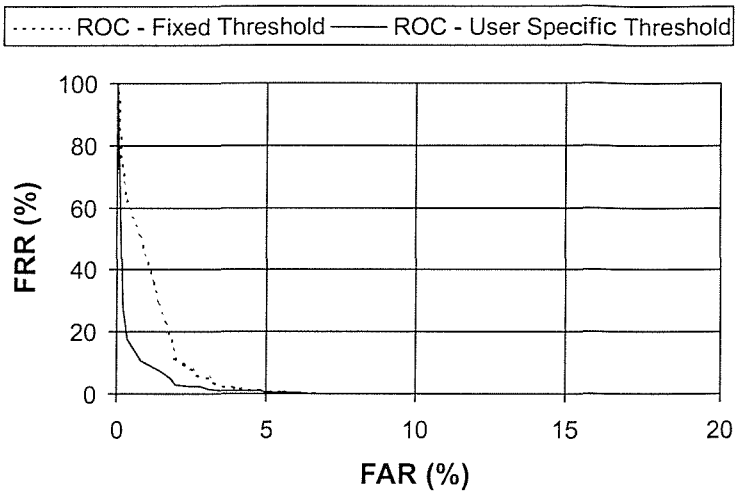


Figure 5.25: ROC Curves for Modified Mahalanobis Verifier

were two methods of calculating the variance matrix, $\mathbf{V}$, either to firstly compute the covariance matrix from the eight enrolment vectors then extract its leading diagonal, or to simply construct a matrix whose leading diagonal is composed of variances from each dimension of the enrolment vectors. Under the fixed threshold regime, the first approach recorded an average of 11108 enrolment FLOPS and 927 verification FLOPS, whilst the second recorded an average of 5179 enrolment FLOPS and (again) 927 FLOPS for the verification process. The user-specific threshold method required an average of 19198 FLOPS during enrolment and 932 FLOPS for verification, using the covariance approach. Directly generating a variance matrix required an average of 11779 enrolment FLOPS and 932 verification FLOPS.

Figure 5.25 shows ROC curves for both threshold approaches. Tightening the false acceptance rate to 1% will result in false rejection rates of 42% and 10% for the fixed threshold and user specific threshold approaches.

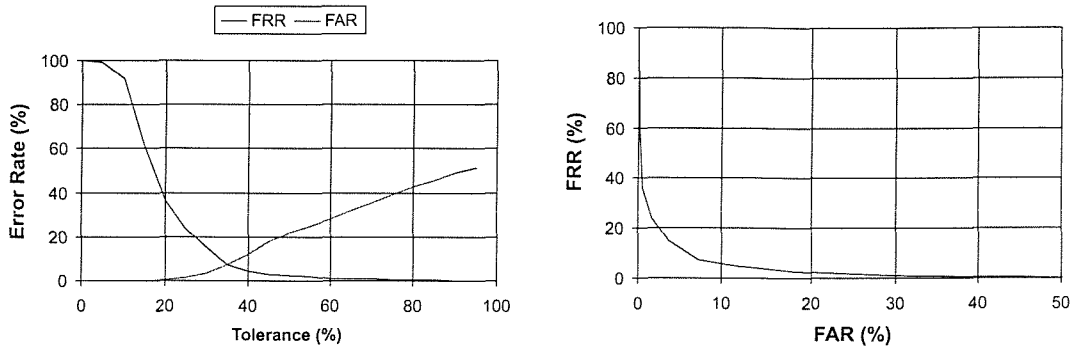
5.7.4 Component-Wise Linear Verifier

This method embodies perhaps one of the simplest verifiers. Its aim is to explicitly compare each component, t_i, of a test vector, $\mathbf{T}$, to the respective component, r_i,

of the mean reference, **R**. **T** will be accepted only if all components satisfy

$$|r_i - t_i| < \theta r_i \quad \forall i \tag{5.32}$$

Figure 5.26(a) shows the error rates for this verifier, whilst Figure 5.26(b) shows its ROC curve.



(a) FAR & FRR (%) (b) ROC

Figure 5.26: Error Characteristics for Component-Wise Linear Verifier

The equal error rate for this approach is found to be 7.2%, requiring an average of 175 FLOPS during enrolment and 157 for verification. The ROC curve suggests that by tightening the rate of false acceptance to 1% the verifier would exhibit a false rejection rate of around 32%.

The condition *for all i*, can be relaxed in the manner of Umphress & Williams (1985). Although Umphress made use of digraph latency times as features in his verifier, his approach of accepting sequence vectors if a fixed proportion of their components fell within set bounds is applicable here. The approach of accepting a test sequence if a preset proportion of components pass some comparison test, lends itself well to this component-wise verifier. By sweeping through a range of acceptance thresholds, as above, but relaxing the condition that all components in a test vector must pass the component-wise comparison to a proportion of components in a test sequence must pass the component-wise comparison, it may be possible to find low error regions within this two-dimensional threshold space.

Defining a component acceptance function $c(t_i)$ as

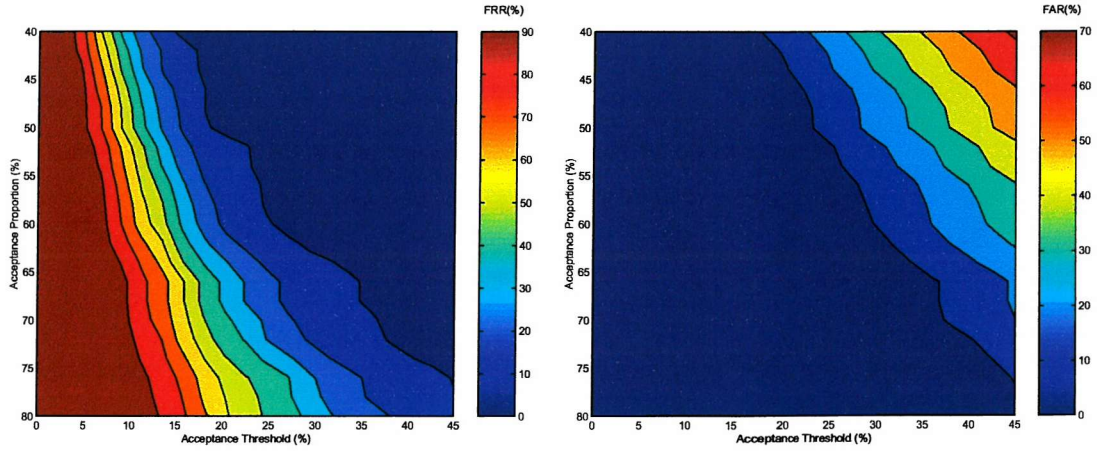
$$\begin{aligned} c(t_i) &= 1, \; if \;\; |r_i - t_i| < \theta r_i \\ &= 0, \; otherwise. \end{aligned} \tag{5.33}$$

A test vector, $\mathbf{T}$ will be accepted if a proportion, p, of its components satisfy

$$\sum_{i=1}^{d} c(t_i) > p\, d \tag{5.34}$$

where d is the number of components of $\mathbf{T}$.

Error rates were calculated for a sweep of component threshold under a range of acceptance proportions. Figures 5.27(a) and 5.27(b) show the false rejection and false acceptance rates under a plane of acceptance thresholds and proportions.



(a) FRR (%) (b) FAR (%)

Figure 5.27: Component-Wise Linear Verifier with Proportional Acceptance

False rejection rates can clearly be seen to decrease with increasing component acceptance threshold and decreasing proportional acceptance. False Rejection rates, meanwhile can be seen to increase with acceptance threshold and proportional acceptance. It was hoped that a region of low errors could be identified in this way. However from Figure 5.28, which shows the sum of false acceptance and false rejection errors, it can be seen that whilst such regions exist they are small and

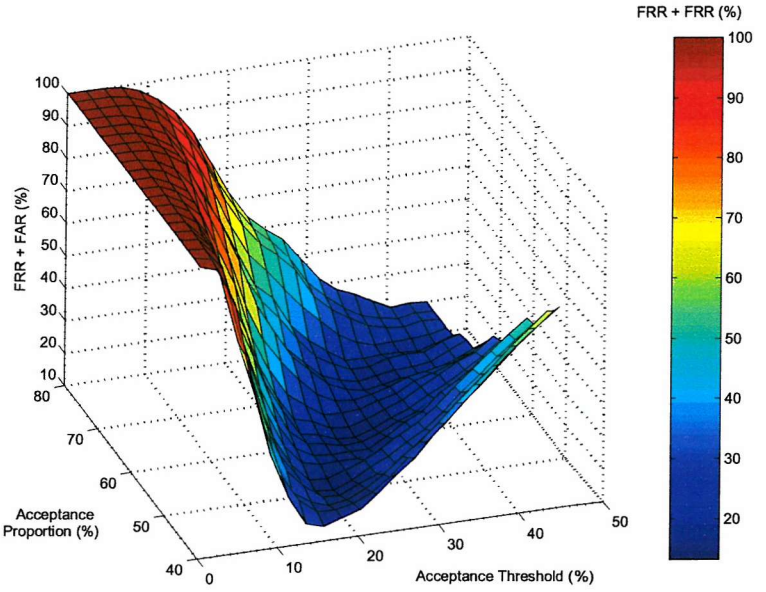


Figure 5.28: Combined Error Rates with Proportional Component Acceptance

likely a function of this particular data set.

5.7.5 Component-Wise Non-Linear Verifier

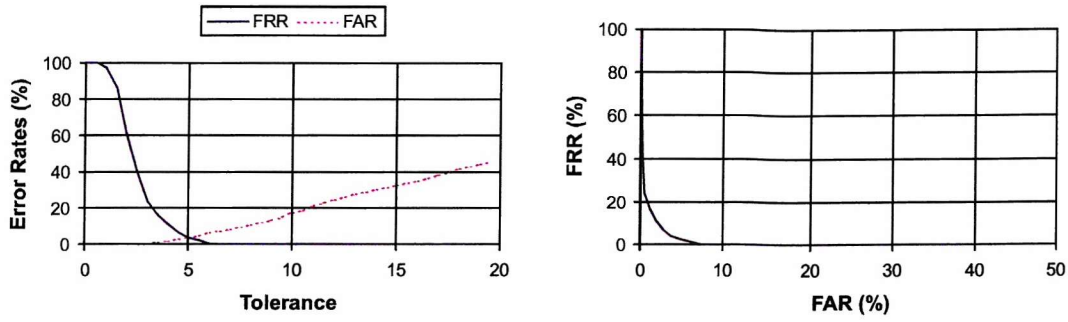
Robinson et al. (1998) reports upon his use of a component-wise non-linear verifier using keystroke latency and hold times. He achieved an equal error rate of 31% using the information contained within usernames, typed several hundred times. This approach is similar to that of the component-wise linear verifier, differing only in that the component acceptance threshold is based upon the component-wise standard deviation calculated from a user's enrolment vectors.

A test vector, $\mathbf{T}$, will be accepted if all of its components satisfy:

$$|r_i - t_i| < \tau \sigma_i \quad \forall i \tag{5.35}$$

Where σ_i is the standard deviation of the i^{th} component of $\mathbf{T}$, calculated from the enrolment vectors, and τ is the acceptance tolerance.

Figures 5.29(a) and 5.29(b) show error characteristics and ROC curve, respectively.



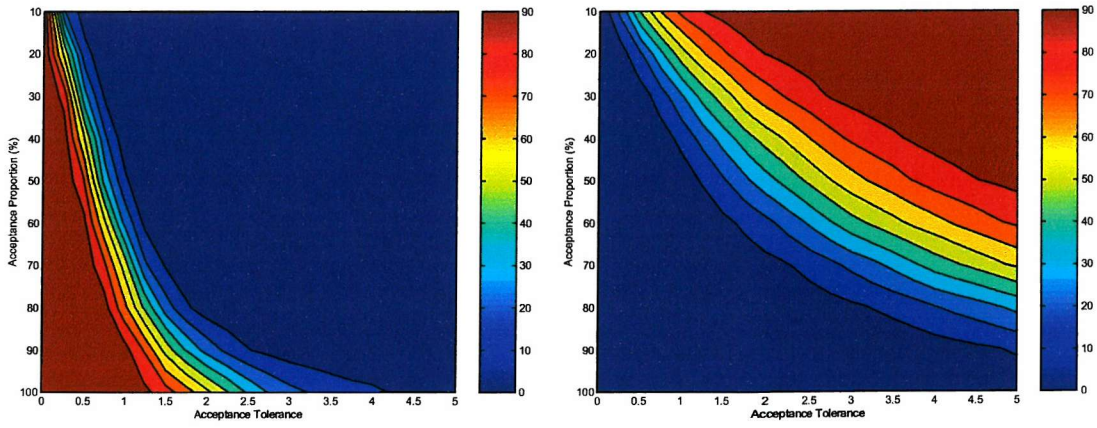
(a) FAR & FRR (%) (b) ROC

Figure 5.29: Error Characteristics for Component-Wise Non-Linear Verifier

In the same manner as the simple component-wise linear verifier, the condition *for all* i can be relaxed to a proportion of the number of components. Defining a component acceptance function $c(t_i)$ as

$$\begin{aligned} c(t_i) &= 1, \; if \;\; |r_i - t_i| < \tau\sigma_i \\ &= 0 \;\; otherwise. \end{aligned} \tag{5.36}$$

In common with the component-wise linear verifier, a test vector will be accepted if (5.34) is satisfied.



(a) FRR (%) (b) FAR (%)

Figure 5.30: Component-Wise Non-Linear Verifier with Proportional Acceptance

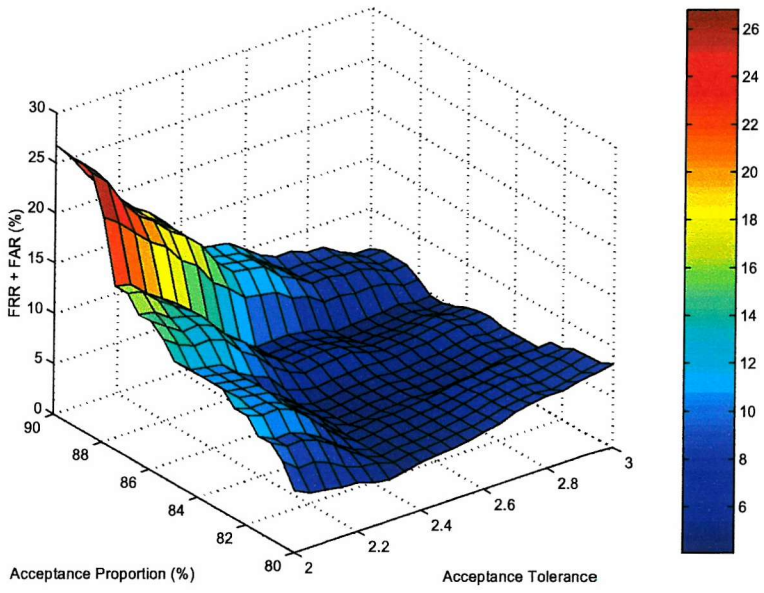


Figure 5.31: Combined Error Rates with Proportional Component Acceptance

5.7.6 Discussion

Figures 5.32 and 5.33 provide a comparison of the error rates and processing intensities between each of the verifiers thus far considered. It is observed that the ℓ_1 and ℓ_2 norms (with acceptance threshold set on a per-user basis), and both threshold approaches of the Mahalanobis distance verifier offer relatively low equal error rates. When the verifiers are biased against false acceptances (as shown in the secondary columns of Figure 5.32) the user-specific ℓ_1 norm and Mahalanobis distance verifiers perform best, offering equal error rates of 2.3% and 2.4%, respectively, whilst both exhibit 10% false rejection rates with 1% false acceptance.

A measure of the processing intensity of all verifiers (shown in Figure 5.33) indicates that computation of the ℓ_1 verifier requires significantly fewer floating point operations than the Mahalanobis distance method.

Equal error rates of 2.3% have been achieved using all features; pulse height, pulse width and interval duration. This figure, nor indeed any of the other error rates, include non-modal pulse sequences. Sequences with non-modal pulses must be considered errors, however, in common with Umphress & Williams (1985), Leggett & Williams (1988), Joyce & Gupta (1990), Brown & Rogers (1993) and Robinson et al. (1998), who all discarded incorrectly spelled samples from valid users, these

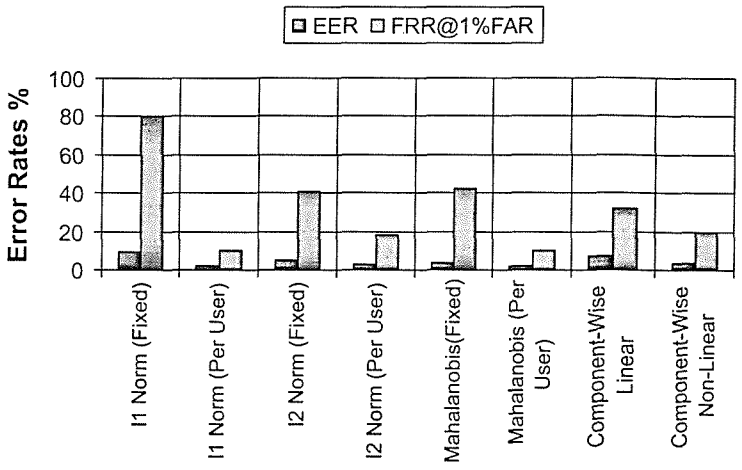


Figure 5.32: Error Rate Comparison with Verifier

non-modal sequences are also discarded. The distribution of modal-pulses has been calculated and the results discussed in Section 5.4.

Furthermore, it must also be stated that these error rates do not include the inconsistent user – who was unable to provide sufficient samples with the same number of pulses for enrolment and testing. Whilst it may be argued that this omission artificially lowers the reported error rates, it is felt that the situation whereby a user is not comfortable with the pressure sequence method and is (for whatever reason) unable to provide satisfactory sequences, is directly analogous to excluding people from using hand geometry biometrics, for example, when they physically have no hands. Of course, taking the populace as a whole, the person who can neither be accepted nor rejected, must be counted as a failure of the system. On the other hand, his unsuitability to this approach is clearly apparent, and he would not be included in trials.

Trialing of a verification scheme must include some measure of the appropriateness of the method, whether this is fingerprints, hand geometry, voice, or whatever, the proportion of people unable or unsuitable to the method should be assessed. No method of identity verification will ever suit all people for all purposes. It seems in this case, one user from 34 has demonstrated unsuitability, representing around 3% of the sample population.

Participants were generally self selected, or coerced with (only) a little persuasion, and (perhaps as a result) reactions have been predominantly positive – with

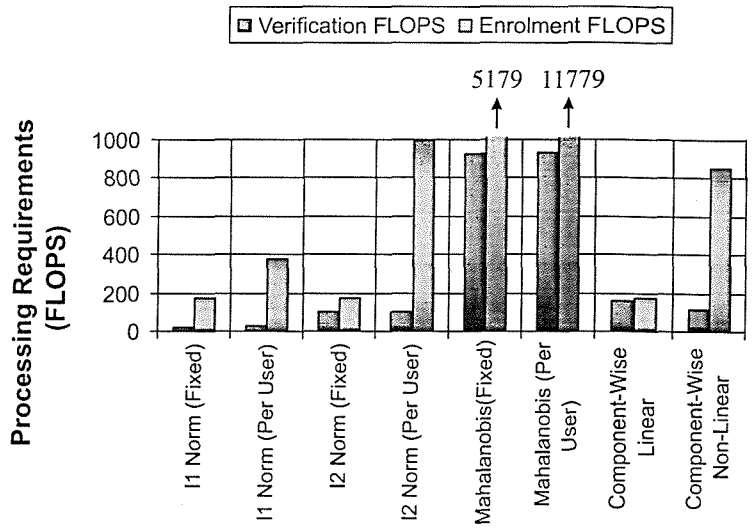


Figure 5.33: Floating Point Comparison with Verifier

comments from the participants including: "fun to use", "interesting", "novel".

It should be further noted that error rates presented thus far are global averages, calculated across the results from all participants within the experimental population. It follows that individual detail is missing. Table 5.1 provides specific performance characteristics for each participant, under the per-user ℓ_1 norm verifier, from which it can be seen that the majority of participant characteristics exhibit equal error rates of 0%. However, this is most likely a function of the limited size of the impostor data set, and although encouraging, emphasizes the need for larger numbers of collected samples. This is highlighted as an important aspect of future work.

It is observed that the mean false rejection rate, at the global crossover point, is due entirely to three participants, each of whom exhibit equal error rates of 0% at slightly higher values of acceptance tolerance.

5.8 Pressure Sequence – Conclusions

Requiring only one single pressure sensor and elementary signal conditioning, this simple idea of identity verification based on inherent rhythm has demonstrated remarkable discrimination. In a laboratory based experiment which involved 34

Participant (Number of Pulses)	Individual Results		Results at Global Crossover (τ=3.25)	
	EER (%)	τ	FRR (%)	FAR (%)
a (2)	8.4	1.125	0	22.8
b (3)	0	2.0–4.25	0	0
c (3)	2.56	2.0	0	3.7
d (3)	2.56	2.75	0	5.1
e (3)	0	1.75–2.5	0	2.4
f (4)	0	2.0–6.75	0	0
g (4)	0	1.25–1.75	0	1.1
h (4)	0	1.75–6.0	0	0
i (4)	0	2.5–6.0	0	0
j (4)	0	2.5–6.25	0	0
k (5)	0	2.75–4.5	0	0
l (5)	1.4	1.75	0	8.4
m (6)	0	1.75–11.5	0	0
n (6)	0	3.0–4.25	0	0
o (6)	0	1.25–14.5	0	0
p (7)	0	1.75–9.5	0	0
q (7)	0	4.75–19.75	37.5	0
r (7)	0	2.75–10.75	0	0
s (7)	0	1.25–11.25	0	0
t (7)	0	1.25–2.75	0	1.2
u (7)	0	1.5–1.75	0	3.6
v (8)	0	1.75–16.5	0	0
w (8)	0	1.75–11.75	0	0
x (8)	0	3.75–19.75	12.5	0
y (8)	5.4	2.125	0	13.6
z (9)	0	3.25–16.5	0	0
aa (9)	0	3.0–12.75	0	0
ab (10)	0	1.0–2.25	0	9.1
ac (10)	0	3.0–19.75	0	0
ad (11)	0	1.25–12.25	0	0
ae (12)	0	6.25–7.25	25.0	0
af (14)	0	1.25–3.0	0	3.1
ag (14)	0	2.25–3.0	0	3.1
		Mean:	2.3	2.3

Table 5.1: Specific Performance Details (User Specific ℓ_1 Norm)

volunteer participants, an equal error rate of 2.3% has been demonstrated.

Pressure responses of taps were crudely represented by pulse-amplitude and pulse-duration, and in conjunction with inter-pulse times form the representative features of a raw analogue rhythm sequence. The body of work on keystroke dynamics is used as a source of potential verification functions, and from an investigation of verification functions the ℓ_1 norm is found to be the most discriminating.

Whilst good discrimination between individuals is shown, there exist a number of uncertainties. For logistical reasons, training and testing samples were captured during the same session, and this does not account for any differences which may be exhibited on a day-to-day basis. Indeed the variability in sequences over time has not been assessed, and this is a requirement for any practical approach to identity verification.

Impostor data was captured in a manner believed to best represent a person 'finding' a smartcard, and using a brute-force method of attack. However, the motivation for impostor perseverance in this experiment is questionable, and must

be strengthened – impostors must have a strong desire to break an individual's sequence. This is an area strongly highlighted for future work. Another critical uncertainty is the effect of an impostor overhearing the input of someone's sequence. Although preliminary experiments suggest that it can be difficult to replicate a sequence to the satisfaction of its originator, the objective effect of overhearing a sequence must be fully assessed.

This work uses pulse-amplitude and pulse-duration to represent pressure responses of finger taps. Clearly from the data presented in this chapter, this is an over simplification, and from Figure 5.11 differences between the responses of two participants are observed. This offers further scope for discrimination, and may result in a sequence-independent method of verification. Such an approach moves towards a true biometric, rather than the current implementation which is that of identity verification.

Chapter 6

Smartcard Processing Considerations

6.1 Introduction

The new identity verification method presented and demonstrated in Chapter 5 complies physically with smartcards, in terms of the manner of demonstrating identity and in the sensor structure used to capture discriminatory characteristics. This chapter considers the algorithmic requirements for executing enrolment and verification functions on-card. Algorithmic demands are twofold, in that the smartcard platform must support the necessary mathematical mechanics, such as floating-point arithmetic, and the smartcard processor must be sufficiently powerful to execute algorithms in a timely fashion. It seems reasonable that verification, executing on a per-transaction basis, should complete within around one second, and the one-off enrolment function within a few seconds.

As stated, the Java Card platform supports applications (or *applets*) written in a subset of the Java language. Whilst this reduced form of Java may be suitable for a great number of smartcard applications, it offers no support for floating-point operations nor basic mathematical functions, such as square roots. This is unfortunate, since the discrimination functions of Chapter 5 rely upon native scientific capabilities of the MatLab numerical analysis package. Hence the first task of this work is to devise suitable integer-based alternatives, and explicit means of im-

plementing the required mathematical functions. The enrolment and verification functions, detailed in the last chapter, are then translated to Java and executed on a typical Java Card platform, allowing time requirements to be recorded and compared. This work forms the basis of Henderson, White & Hartel (2001).

The chapter begins with a brief overview of programming the Java Card smartcard platform.

6.2 The Java Card Platform

Java Card technology enables applications written in the Java language to run on smartcards and other embedded systems. As indicated in Chapter 1, this offers developers the advantages of a structured, type-safe high-level development language, whilst complying with the security and trust philosophy of the smartcard. Java Card effectively insulates the programmer from underlying hardware and protocol layers, and as a result programs may be rapidly developed, and written by independent third parties for any compliant device. For more details on Java Card, see Chen (2000).

The Java Card platform used in this work was implemented on a device called an *iButton* from Dallas Semiconductor[1]. The iButton is a 32-bit smartcard RISC processor, housed in a small (16mm ⌀× 6mm) stainless steel canister rather than embedded in the plastic substrate of a smartcard. The steel canister also houses a battery and real-time clock, but these differences aside, the processor is comparable to other modern Java Card smartcards, with 32kBytes of EEPROM for application storage and 1kByte of volatile RAM for run-time processing. The iButton was chosen as a test-platform because of its availability and its good simulation and debug support.

Applets were compiled using Sun Microsystem's Java Card 2.1.1 Development Kit, under version 1.10 of Dallas Semiconductor's iButton Development environment (iB-IDE). The resulting Java-Code was run on the Java Card 2.0 compliant Java-Virtual-Machine (JVM) of the iButton.

[1]Dallas Semiconductor, Austin, Texas, USA. Web`http://www.iButton.com`

6.3 Implementing Verification Functions

To test the suitability of the pressure sequence verifiers, the enrolment and verification functions of each verifier were written as stand-alone Java functions and called from a skeletal applet, containing the essential communication and timing routines. The applet code can be found in Appendix G, and its structure is as follows:

- Retrieve and store pre-processed enrolment vectors from PC
- Record start time
- Perform enrolment (a number of times)
- Record finish time
- Send total enrolment time and reference vector to PC.

The feature vectors used were obtained from the data collection exercise of Section 5.4.2. Once the iButton had completed vector processing, reference vectors and processing times were returned to the host PC, whereupon the reference vectors were checked for consistency with the MatLab results (described in Section 5.7), and mean processing times were calculated. Verification times were measured in the same way as enrolment times, with the verification function called in place of enrolment.

The applet executes by firstly loading a user's enrolment feature vectors, pre-processed from their enrolment sequences, to the iButton. Time is recorded using the iButton's real-time clock, then the enrolment function is called a number of times. Upon completion, the finish time is recorded and the process duration is calculated. This is then transmitted to the host PC, and a mean execution time is calculated.

Implementing the process in this way avoids lengthy PC to iButton communications which are not involved in the enrolment (or verification process). This process assumes that the enrolment samples have already been captured, pre-processed, and are now available to the processor. This will be considered further in section 6.4.

6.3.1 Specific Implementation issues

There are two fundamental limitations to running the verifiers on the Java Card platform: the restriction to 32-bit integers and hence the lack of floating-point data-types; and the lack of elementary mathematical functions, such as square-roots.

The fixed threshold version of one of the most successful verification schemes, the ℓ_1 norm is defined in equation (5.4) and is repeated for the convenience of the forthcoming discussion:

$$\|\mathbf{R} - \mathbf{T}\|_1 = \sum_{i=1}^{d} |r_i - t_i|$$

where $\mathbf{R}$ is the d dimensional reference-vector, generated from the mean of each of the components in the user's enrolment vectors, $\mathbf{T}$ is a d dimensional test vector, and r and t are the components of $\mathbf{R}$ and $\mathbf{T}$ respectively.

The identity of a user will be accepted if equation (5.24) is satisfied, that is

$$\|\mathbf{R} - \mathbf{T}\|_1 \leq \theta$$

where θ is the acceptance threshold.

Calculation of the ℓ_1 norm is achievable with no loss of precision under the restriction of the 32-bit integer data type. Using a fixed acceptance threshold, however, resulted in a rather uninspiring equal error rate (EER) of 9.7% (see Figure 5.32), since the same acceptance threshold is used for all users irrespective of their natural variance.

To improve upon this rate, Section 5.7.1 considered taking the variation of a user's enrolment vectors from their reference vector into account, and equation (5.25) was the result. This is repeated (again for the sake of convenience) as

$$\|\mathbf{R} - \mathbf{T}\|_1 \leq |\overline{D}_{Enrolment} - \tau \cdot \sigma_{Enrolment}|$$

where τ is the globally-set acceptance tolerance, $\overline{D}_{Enrolment}$ is the mean distance of the enrolment vectors to the reference vector, and $\sigma_{Enrolment}$ is the standard deviation of these distances.

The problem for the integer-only Java Card system is that $\sigma_{Enrolment}$ requires the calculation of a square root:

$$\sigma_{Enrolment} = \sqrt{\frac{\sum_{j=1}^{n}(\mathbf{S}_j - \mathbf{R})^2}{(n-1)}} \tag{6.4}$$

where n is the number of enrolment vectors, and $\mathbf{S}_j$ is the j^{th} enrolment vector. Whilst the immediately obvious method for calculation of square roots, the Newton-Raphson method (Press, Teukolsky, Vetterling & Flannery 1993), is well known and used, it suffers from a number of problems. Firstly, it is an iterative method, whose efficiency depends upon the quality of the initial guess. Secondly, applied to integers in its native form, the Newton-Raphson method will infinitely oscillate between two integers, above and below a real non-integer root (Crenshaw 1998). Crenshaw has devised a simple algorithm for the calculation of square roots. His method is essentially a search through all possible integers until the integer part of a non-integer root is found. The simplest form of this is to search all possible integer square roots, x, for $\sqrt{N}$ until $x^2 \geq N$. The integer root of N is then the exit value of x, minus one. However, Crenshaw provides a more efficient implementation arising from the observation that

$$(x+1)^2 = x^2 + (2x+1). \tag{6.5}$$

This means that to check each new possible square root, $(2x+1)$ merely has to be added to the previous square. The Java code implementation for this is presented as follows (listing 6.1):

Listing 6.1: Java Code for Integer Square Root Function

```
public static int SQRT(int a)
{
    int square = 1;             // x=1: 1st Integer Square Root
    int delta = 3;              // (2x+1), for x=1

    while(square<=a) {
        square+=delta;          // (x+1)^2 = x^2 + (2x+1)
        delta +=2;              // Next value for (2x+1)
    }
    return (delta/2 - 1);       // square is now > a, so find previous value of x
}
```

Although faster fixed cycle-length square root functions exist (Crenshaw 1998),

...	Mean Distance	Standard Deviation
Floating Point Result	221	49.29
iButton's Integer Result	222	48

Table 6.1: Comparison of Floating Point and Integer Results

the time required for execution of the enrolment process using this implementation was measured to be 3.1 seconds – well within the bounds of a *reasonable* duration. 20 square roots were required for the enrolment process, each iterating 49 times. Verification required only 0.12 seconds for completion.

Although the final values for the mean distance and its standard deviation between enrolment vectors and the user's reference vector will be truncated to integer levels, the fractional loss is small relative to the distances involved and is not expected to cause any significant reduction in the verifier's accuracy. Table 6.1, presenting the floating point and integer results from one user, illustrates this point.

The second most discriminating method; the Mahalanobis distance verifier is defined in equation (5.31) as follows:

$$(\mathbf{R}-\mathbf{T})\cdot\mathbf{V}^{-1}\cdot(\mathbf{R}-\mathbf{T}) \leq |\overline{D}_{Enrolment} - \tau\cdot\sigma_{Enrolment}|$$

where $\mathbf{V}^{-1}$ is defined as the inverse of a square $d\times d$ matrix whose leading diagonal is composed of the variances from each dimension of the enrolment vectors and all other elements are zero.

This leads us to the problem that multi-dimensional arrays are not supported under Java Card 2.0, and as a result matrix manipulation will be both convoluted and time consuming. The solution rests with the analytical reduction of the left-hand side of (5.31) to give:

$$(\mathbf{R}-\mathbf{T})\cdot\mathbf{V}^{-1}\cdot(\mathbf{R}-\mathbf{T}) = \sum_{i=1}^{d}\frac{(r_i-t_i)^2}{v_i} \tag{6.7}$$

where v_i is the variance of the i^{th} component of the enrolment vectors. Equation (6.7) now offers a route to the direct computation of the Mahalanobis distance verifier.

There is one further catch, however. v_i as the variance of each vector component,

...	Mean Distance	Standard Deviation
Floating Point Result	17.5	5.56
iButton's Integer Result	10	4
iButton's Pre-Multiplied Integer Result	17	5

Table 6.2: Comparison of Floating Point, Integer and Pre-Multiplied Integer Results

is the square of the standard deviation for that component. It is hence likely to be of comparable magnitude to that of the component values. The integer division, therefore, of $(r_i - t_i)^2$ by v_i is extremely likely to result in the loss of a significant fractional part of the result, further compounded by the sum across all components. Pre-multiplying each numerator by $10^{RequiredPrecision}$ and post-dividing the sum by the same enables retention of the fractional information. Table (6.2) gives the floating point, integer and pre-multiplied integer results.

Execution of the Mahalanobis enrolment process required 4.5 seconds, whilst verification completed within 0.16 seconds.

6.3.2 Further Results

The enrolment and verification times for the other verification schemes considered in Chapter 5 were also measured on the iButton. Additionally, the combined program size for enrolment and verification functions was recorded. Table (6.3) presents a comparison of these quantities, along with a repetition of the Equal Error Rates, for convenience.

6.4 Proposed System Architecture

The above analysis assumes that the smartcard processor has access to feature vectors of live pressure sequences. This is assumed, and is desirable, in that the smartcard processor is not required to continuously monitor the live sensor output. Since the output signal from the sensor is an analog quantity and requires to be digitised before any processing can be performed, an on-card biometric system must have analog to digital conversion circuitry. There are hence, two options.

Verification Scheme	Enrolment Time (Seconds)	Verification Time (Seconds)	Program Size (bytes)	EER%
ℓ_1 norm - Fixed Threshold	1.2	0.12	296	9.7
ℓ_2 norm - Fixed Threshold	1.2	0.26	356	5
Component-Wise Linear	1.2	0.19	285	7.2
MICD - Fixed Threshold	1.2	0.15	319	5.2
Component-Wise Non-Linear	2.9	0.29	524	3.7
ℓ_1 norm - User Specific	3.1	0.12	684	2.3
ℓ_2 norm - User Specific	4.2	0.25	707	3
Mahalanobis - Fixed Threshold	2.4	0.16	550	3.4
Mahalanobis User Specific	4.5	0.16	752	2.4
MICD - User Specific	8.5	0.16	704	2.8

Table 6.3: Results for all Verifiers

Either the architecture of the smartcard's processor hardware can be modified to include such circuitry, or more generally, an additional signal conditioning IC can be included on-card. In the former case the processor becomes specific to one, and only one form of identity verification, whereas in the latter, smartcard processors remain standard and an appropriate conditioning IC is included with the sensor.

Since the Java Card platform offers unprecedented flexibility to the programming of smartcards, the inclusion of biometric specific circuitry moves counter to the generalist philosophy of Java Card. The latter option of employing a separate conditioning IC module is therefore favoured.

It is proposed that the conditioning module contains all the required signal conditioning (for example charge amplifiers), analog to digital converters and any necessary control circuitry. This can be extended to include some local processing functionality, such that the feature extraction process can occur autonomously and free from smartcard processor intervention. An approach of this manner is directly analagous to the fingerprint sensors of Jung et al. (1999), in which embedded logic identifies and extracts minutiae locations. It is suggested that this biometric module could be based around one of the many available microcontroller ICs, for example, the 16F73 peripheral interface controller (PIC) from Microchip[2], which includes a number of on-board ADC channels and communication ports.

Figure 6.1 outlines one possible architecture.

[2]Microchip Technology Inc. Chandler, AZ, USA. Web:`http://www.microchip.com`

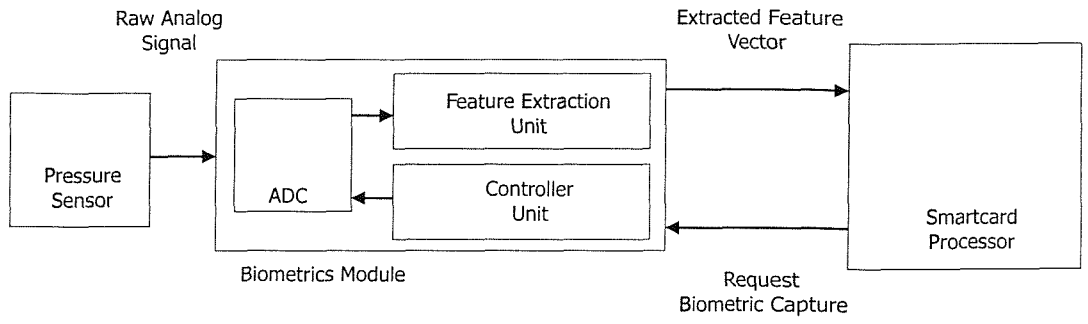


Figure 6.1: Biometric System Schematic

The smartcard processor issues a request to capture a biometric sample, this is recieved by the biometric module, and signal capture is initiated. Upon completion, a feature extraction component analyses the captured signal, performs the necessary preprocessing and transmits this information back to the smartcard processor. It should however be emphasised, that enrolment and verification occur on the tamper-resistant smartcard processor, and not within this biometric module.

6.5 Conclusions

In this chapter the enrolment and verification functions of the verifiers developed in Chapter 5, were implemented on a Java Card platform. Necessary mathematical resources, such as square roots and matrix division, are not available to the native Java Card platform, and as a result some effort was invested in overcoming these limitations.

The execution times of enrolment and verification processes were measured and found to be well within reasonable bounds, and the most discriminating verifier (the user specific ℓ_1 norm) required 3.1 seconds to perform enrolment and 0.12 seconds for verification. Furthermore, both enrolment and verification functions of this verifier, taken together, were found to use < 700 Bytes of EEPROM resources. This combination of processing time and applet size, demonstrates that the computational requirements of the pressure sequence method are well within reach for a typical Java Card platform.

Chapter 7

Conclusions and Further Work

7.1 Conclusions

Modern smartcards are fully programmable, open and trusted computational platforms, and are used for a wide range of functions. At the simplest level, they can be involved in the storage of electronic data and for the prepayment of goods and services. With the cryptographic capabilities of high-end smartcards, applications include electronic authentication of an individual's credentials, and the signing of documents and transactions. As a result, smartcards represent an important tool of e-commerce and information security.

Unfortunately, the fundamental association between an individual and their smartcard is based upon possession or knowledge, both of which are widely considered to be weak demonstrations of identity, since cards can be lost, stolen or 'borrowed', and secret knowledge can be discovered by third-parties. Whilst a smartcard can robustly demonstrate its involvement during a transaction, or its presence during remote access of resources, the presence of its rightful holder can not be assured. There is hence a strong motivation to tighten the association between a smartcard and its holder.

Currently, there is growing interest in using biometric characteristics to strengthen this association, and a number of schemes have been commercially developed, and proposed in the open literature. However, these schemes predominantly rely upon components, such as sensors or processing elements, which are external to the card.

Whilst offering improved binding of the holder's identity to the card, the reliance upon external components restricts where the card can be used, and moreover, exposes the possibility of eavesdrop and replay attacks.

To overcome these concerns, this thesis proposes integrating an identity verification mechanism on-card. However, the physical size and properties of a smartcard introduces a number of challenges. The size of a smartcard restricts the manner in which identity can be demonstrated, its mechanical flexibility imposes the use of compliant sensing mechanisms, and its constrained computational resources limit the complexity of the necessary enrolment and verification algorithms. Each of these areas is considered in the work of this thesis.

In Chapter 2 a comprehensive review of biometrics is provided, from which it is apparent that demonstration of a smartcard holder's identity can be based upon aspects of the hand and fingers. Possible characteristics include: fingerprints; finger-geometry; finger-crease pattern; grasping pressures; and palm-prints. The measurands of pressure, temperature and capacitance are identified as means of capturing discriminating characteristics.

That the smartcard imposes particular restrictions on sensor properties has been indicated, and it is apparent that a sensor requires to be geometrically planar, mechanically robust, flexible and economically compliant with smartcards. Such restrictions can be satisfied by a class of sensors, known as polymer thick film sensors, and a review of the field is presented in Chapter 3. Polymer thick film offer the advantage of low processing temperatures, and as a result can be printed onto a wide range of substrates, including flexible polymers. This property of flexibility, and in conjunction with those of planar geometry, mechanical robustness and low cost, make PTF sensors inherently suitable for integration with smartcards.

Piezoresistive and piezoelectric pressure sensors can be realised using polymer thick films, and an array structure of such sensing elements are most likely to fully capture spatial human characteristics. It is found that the inherent resolution of polymer thick film sensing arrays is limited by the achievable thick film linewidths. Moreover, since polymer thick film is a passive technology, every charge-generating sensing element (such as a piezoelectric element) requires an explicit conductive path connecting it to the outside of an array. This severely restricts spatial resolution since conductive tracks must either pass between sensing elements, or be

buried beneath an insulating layer. It transpires that pixel pitch varies in proportion to the number of sensing elements per row of an array, and hence resolution must decrease with increasing numbers of sensing elements. The design of a charge-generating PTF sensor array then becomes a trade-off between the required number of sensing elements and the desired spatial resolution.

Piezoresistive elements, on the other hand, can be connected on a per row and per column basis, are be addressed individually by measuring the voltage drop across a specific intersection. Spatial resolution of such an array is limited by the linewidth and registration resolution of the printing process. It is estimated that a minimum pixel pitch of around 250μm is achievable, resulting in a spatial resolution of around 40 elements per centimetre.

However, detailed theoretical models of both sensors are derived, and it is apparent that both sensors will exhibit sensitivity to planar strains. Since the flex of a smartcard results in planar strains propagating across the card, planar sensitivity of the sensors, rather than array architecture, becomes the limiting factor to spatial resolution.

Chapter 4 makes use of first-order finite element analysis to assess the extent of strain propagation across a smartcard, and the extent to which an array of sensing elements respond to a single applied load, typical of human-finger interactions. It is found that sensors across the surface of the card will respond significantly to a singular load applied elsewhere. Whilst sensor responses are observed to diminish with distance from an applied load, they remain significant in proportion to the peak response. This effect becomes the limiting factor in bonding sensors to smartcards, and it is not practical to bond more than one sensor, without suffering cross-sensitivities between sensors.

Further, it is shown that the flex of a smartcard is an important agent in the sensitivity of both sensors, and constraining the vertical out-of-plane freedom of a smartcard significantly reduces sensitivity. A way around this has been discovered, and involves the formation of a dome structure underneath the sensor. In response to an applied load, the dome deforms, and the resulting strains are transferred to the sensor.

Both sensors are shown to capture aspects of the temporal (rather than spatial)

interaction between a finger and the card. Piezoresistive sensors can adequately capture both static and dynamic signals, whilst piezoelectric sensors are able to capture fast-changing dynamic signals rather than static or quasi-static signals. But, piezoresistive sensors suffer from hysteretic behaviour, and whilst this is not characterised, it is observed to distort fast changing, dynamic interactions.

Chapter 5 proposes and demonstrates a novel approach to identity verification based upon temporal interactions between a finger and pressure sensor. The pressure responses of an individual's finger are recorded and used as a basis for discrimination. Requiring only one single pressure sensor and elementary signal conditioning, this simple idea of identity verification based on inherent rhythm has demonstrated remarkable discrimination. In an experiment involving 34 volunteer participants, each using self-selected rhythms, an equal error rate of 2.3% has been demonstrated.

Pressure responses of taps were represented by pulse-amplitude and pulse-duration, only, and in conjunction with inter-pulse times form the representative features of a raw analog rhythm sequence. The body of work on keystroke dynamics is used as a source of potential verification functions, and from an investigation of verification functions the ℓ_1 norm is found to be the most discriminating.

For logistical reasons, training and testing samples were captured during the same session, and this does not account for any differences which may be exhibited on a day-to-day basis. Indeed the variability in sequences over time has not been assessed, and this is a requirement for any practical approach to identity verification. Impostor data was captured in a manner believed to best represent a person 'finding' a smartcard, and using a brute-force method of attack.

In Chapter 6 the enrolment and verification functions of the verifiers developed in Chapter 5, were implemented on a Java Card platform. Necessary mathematical resources, such as square roots and matrix division, are not available to the native Java Card platform, and as a result some effort was invested in overcoming these limitations.

The execution times of enrolment and verification processes were measured and found to be well within reasonable bounds, and the most discriminating verifier (the user specific ℓ_1 norm) required 3.1 seconds to perform enrolment and 0.12

seconds for verification. Furthermore, both enrolment and verification functions of this verifier, taken together, were found to use < 700 Bytes of EEPROM resources. This combination of processing time and applet size, demonstrates that the computational requirements of the pressure sequence method are well within reach for a typical Java Card platform.

7.2 Further Work

7.2.1 Pressure Sequence

Whilst discrimination between individuals has been demonstrated, there are a number of unresolved issues. Day to day variations in sequences has not been assessed, and this is clearly of practical importance for the method. A long term data collection exercise should be set up, perhaps with some incentives encouraging participants to enter their data on a frequent basis. This should be open to as many people as possible, most likely on a departmental basis.

One crucial experiment which must be performed, is to test the effect of impostors overhearing a user entering their rhythm. Preliminary experiments suggest that it can be significantly harder than one might think to replicate the sequence of another, although this must be rigourously verified. The sensitivity to eavesdropping in this way will depend upon the complexity and inherent variance of an individual's rhythm, and this should be investigated.

Section 5.4.3 illustrates some differences between singular tap-responses from two participants, and it is clear that response characteristics can be quite different between two people. As indicated in Section 5.7.6, very recent work at the University of Twente has explored the use of statistical representations of tap responses, and from this a verifier has been demonstrated with an equal error rate of around 7.7%. This work can be extended further by modelling the biomechanical interactions of a finger tapping upon a smartcard, from which discriminatory tap features may arise. If successful then this will guard against overhearing and repetition of a rhythm, indeed it may lead to an entirely sequence-independent verifier. In this instance the knowledge-based component of the sequence (the rhythm) is removed, and the approach becomes a true behavioural biometric.

The practical security of this method needs to be investigated, and impostors must be sufficiently motivated to sustain attempts of circumvention. A proposition to this effect could involve a number of participants (again, as many as possible), each given some protected file space on a network. The file space would be protected by the pressure sequence method, such that a user would login by stating their claimed identity, then demonstrate this using their self-selected rhythm. To provide an incentive for impostors, each person's file space would contain a one-time-usable electronic token, such as a £20 Amazon[1] voucher. Some mechanism would be devised that requires each authorised person to login to their file space regularly (so that impostors have a chance of overhearing), but prohibits the authorised person from spending their voucher until the end of the trial, which of course is only possible if it has not already been spent.

7.2.2 Alternative Applications

It is the flex of a smartcard which enhances the sensitivity of polymer thick film pressure sensors to an applied load, and constraining a sensor in the vertical direction significantly reduces its sensitivity. However, Section 4.9 describes a method of improving sensitivity of constrained sensors, which involves the fabrication of a dome structure underneath a PTF sensor. This will allow sensors to be incorporated onto rigid platforms such as Personal Digital Assistants (PDAs) or generic smart objects.

Further, it may be possible to incorporate pressure sensors into numeric keypads, as suggested by Spillane (1975), and to adapt the pressure sequence approach. This has the potential of providing a layer of security which is additional to PINs.

This is an area which should be investigated.

7.2.3 Alternative Spatial Sensing Mechanisms

The pressure sensors considered in this thesis are sensitive to planar strains, and since a smartcard flexes in response to an applied load, sensors will respond to strains applied elsewhere on the card. If spatial characteristics, such as finger

[1] `http://www.amazon.co.uk`

geometry, or finger crease pattern are to be captured then a sensing mechanism which is predominantly sensitive to normal presentation of characteristics must be found.

Chapter 3 proposed a new polymer thick film implementation of a capacitance sensor, which should be sensitive to proximity of an object, rather than load. In this sense, a capacitance sensor will be unaffected by flex of the card, and it may be possible to capture finger characteristics, as described above. Such sensors generate charge in proportion to the proximity of an object such as the finger, and hence require the type of array architecture considered for charge generating piezoelectric sensors (Section 3.4.1). From these considerations, and from equation (3.3), it is shown that an array of elements with a pitch of $2mm^2$ and a printing linewidth of $100\mu m$, can support 20 elements per row of an array, which would occupy 4cm across. This may be sufficient to capture finger characteristics, but detailed research is needed.

7.2.4 On-Card Speaker Recognition

As indicated in Chapter 1, on-card voice-pattern recognition has been demonstrated algorithmically (Phipps & King 1997, George 2000), although the system requires an external microphone for voice capture. In Section 2.4, an on-card speaker recognition system was identified as a practical proposition from the perspective of acquiring biometric characteristics. It was suggested that given a suitable microphone, an embedded on-card speaker recognition system could be practical.

Recently a number of authors have reported silicon micromachined microphones, which, due to their size are suitable for smartcard integration (Rombach, Müllenborn, Klein & Rasmussen 2002, Torkkeli, Rusanen, Saarilahti, Seppä, Sipola & Hietanen 2000, Bree, Leussink, Korthorst, Jansen, Lammerink & Elwenspoek 1996). For example Rombach et al. (2002) describes two micromachined capacitive microphones with planar dimensions of 1×1 and 2×2mm with a thickness of approximately $15\mu m$. These devices offer good linearity and a relatively flat frequency response to around 5kHz.

Using on-card displays (as detailed by Praca & Barral (2001)) or MEMS acoustic

actuators (as reported by Neumann & Gabriel (2002)), speaker recognition could be randomly prompted, as described in section 2.3.10. This would offer improved resilience to circumvention.

All the components of an on-card speaker recognition system have been demonstrated, and their complete integration with smartcards should be investigated.

7.2.5 Alternative Technologies

A novel method of implementing a low cost, flexible sensor may be through the use of semiconducting polymers. A class of plastics known as conjugated polymers can be made to conduct or semiconduct depending upon the level of doping (Garnier, Haklaouri, Yasser & Srivastava 1994) . Semiconducting polymers have been incorporated into Light Emitting Diodes (LEDs) (Burroughs, Bradley & Brown 1990) and Field Effect Transistors (FETs) (Bao, Feng & Dodabalapur 1997). Logic gates made from polymer transistors have been demonstrated, and all-polymer integrated circuits fabricated (Drury, Mutsaers & Hart 1998).

The use of all-polymer transistors offers the potential to create an array within which charge can be localised, and data extracted in the manner of (Young 1997). In conjunction with the capacitance sensor implementation of Chapter 3, all parts of this sensing array could be fabricated using flexible, low-cost polymers.

This is an important area of further work.

7.3 Concluding Remarks

The work of this thesis has proposed and demonstrated a novel approach to identity verification, which complies mechanically, economically, and computationally with smartcards. Although a number of ancilliary components, such as power source, display and input switches are required to complete an on-card verification system, there are no fundamental technological barriers to their incorporation on-card (Praca & Barral 2001). Indeed as a demonstration of feasibility, Gemplus[2] have reported the manufacture of a smartcard with on-board power, display and user

[2]Gemplus Research Labs, BP100, Gémenos Cedex, France. Web: `http://www.gemplus.com`

input components. The challenge remaining is to mass-manufacture such cards economically, which according to Praca & Barral (2001), requires that the cost of high-end cards be less than around $10.

Appendix A

Beam Theory

A.1 Background

Young's modulus of a smartcard can be measured using the cantilever deflection method. This involves applying load to one end of the card, whilst the other is rigidly clamped. The deflection of the unclamped edge is proportional to Young's modulus of the material.

This appendix outlines the relationship between deflection and Young's modulus.

A.2 Beam Bending Description

Considering a beam which is clamped at one end and has a known load, W, applied at a distance, L, from the clamp. A vertical deflection, Z, will result, this situation is described in Figure A.1.

The relationship between Young's Modulus, E, beam length, L, vertical displacement, Z and applied load, W is given as

$$Z = -\frac{1}{6}\frac{W}{EI_x}\left(y^3 - 3L^2y + 2L^3\right) \tag{A.1}$$

where y is the distance from the free end at which the deflection measurement is made, and I_x is the moment of inertia of the beam around the centroid axis (Roark

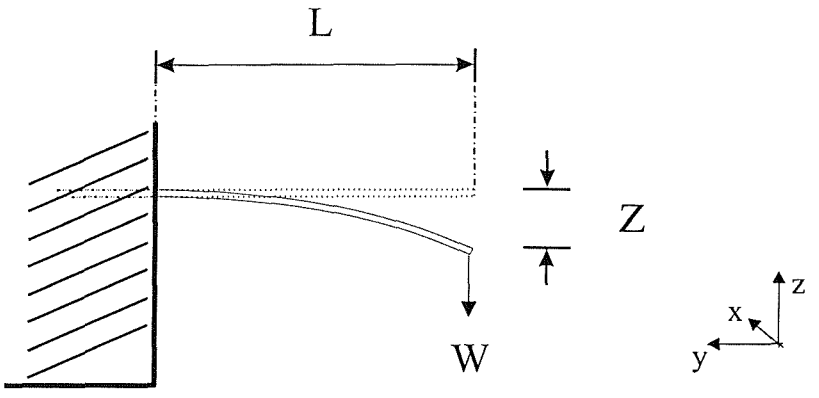


Figure A.1: Cantilever Deflection Schematic

& Young 1975). In the case of $y = 0$, (A.1) is reduced to

$$Z = -\frac{1}{3}\frac{WL^3}{EI_x} \tag{A.2}$$

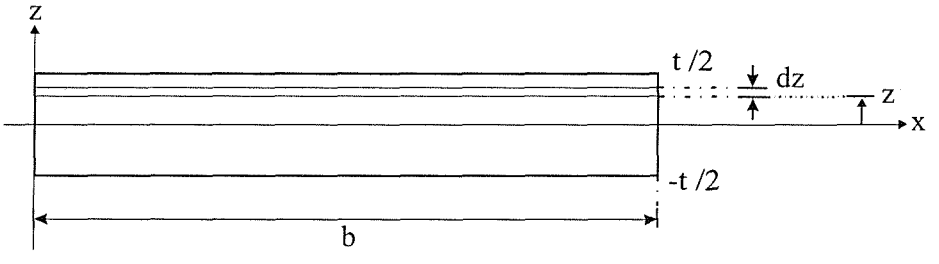


Figure A.2: Beam Cross Section

Considering a cross-section through the width of the beam (as shown in Figure A.2), and noting that the moment of inertia of a differential area, da, around the x-axis (*centroid axis*) is given as $I_{x\ da} = z^2 da$ then the moment of inertia of the beam is calculated as

$$I_x = \int z^2 da \tag{A.3}$$

$$= \int_{-\frac{t}{2}}^{\frac{t}{2}} bz^2 dz$$

$$\Longrightarrow I_x = \frac{bt^3}{12} \tag{A.4}$$

where $da = b\,dz$ and b is the width of the beam (see Figure A.2).

Combining (A.2) with (A.3) and rearranging, gives

$$E = \left[\frac{4}{bt^3}\right] \frac{LW}{(-Z)} \tag{A.5}$$

It is (A.5) which is used in section (4.3.1) to calculate Young's modulus of a smartcard, given the deflection arising from an applied load.

Appendix B

Signal Conditioning Electronics

This Appendix describes in detail the signal conditioning circuits used to capture the output responses from both the piezoresistive and piezoelectric sensors, described in Chapter 3, and used in Chapters 4 & 5. Piezoresistive responses are captured using a resistive bridge whose output is connected to an instrumentation amplifier, whilst piezoelectric signals are captured using a low-current charge amplifier circuit.

B.1 Piezoresistive Signal Conditioning

The signal conditioning circuit used to sense the resistance change of the piezoresistive sensor is a variation of the classic resistive (Wheatstone) bridge circuit. This implementation comprises of four resistors connected through two parallel branches, driven by a constant input voltage. The output voltage from the bridge is measured between points B and D as shown in Figure B.1. R_1 and R_2 are of fixed resistance, R_3 is a variable resistor, whilst the resistance of the sensor, R_{Sensor} will change in response to applied stress.

A full analysis of the bridge circuit is presented in appendix C. The resistance of the unstressed sensor, at 23°C, was measured[1] to be 87.15kΩ. Equation (C.8)

[1]Resistance measurements made using a Keithely 2000 series multimeter (to ±0.008% accuracy). Keithley Instruments, Inc. 28775 Aurora Road, Cleveland, Ohio 44139. USA. Web: http://www.keithley.com

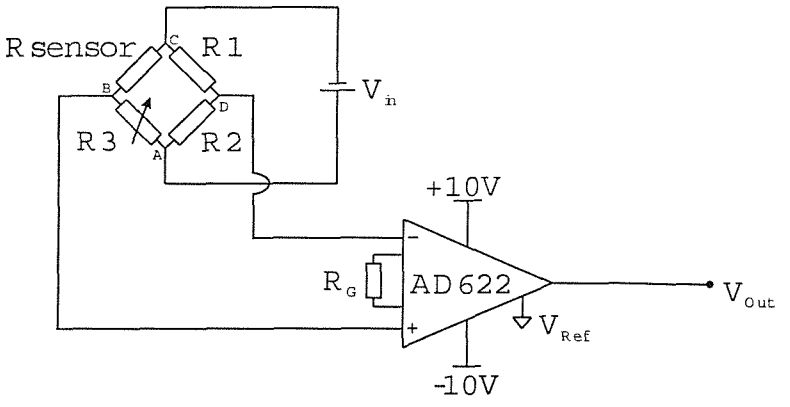


Figure B.1: Piezoresistor Signal Conditioning

states that the condition for the bridge to be balanced, that is $V_{BD} = 0$, the ratio of R_{Sensor} to R_3 and the ratio of R_1 to R_2 must be equal. For convenience this ratio was taken to be around 10, with R_1 measured to be 9853.7Ω and R_2 being 977.6Ω. The variable resistor, R_3, was used to balance the bridge. Figure C.2 clearly shows that for proportional changes in resistance of $\frac{\Delta R_{Sensor}}{R_{Sensor}} \lessapprox 10\%$, the proportional change in bridge output voltage, $\frac{\Delta V}{V_{in}}$, can be assumed to be linear. Rearranging (C.11), it is found

$$\frac{\Delta R_{Sensor}}{R_{Sensor}} = -\frac{(1+r)^2}{r}\frac{\Delta V}{V_{in}} \tag{B.1}$$

where r is the resistance ratio $\frac{R_{Sensor}}{R_3}$ and $\frac{R_1}{R_2}$.

Since the output voltage from the bridge is typically small (in the order of a few milivolts per percentile resistance change), amplification is required. The amplifier used here is an AD622 instrumentation amplifier from Analog Devices[2]. This amplifier is based around the classic three op-amp instrumentation amplifier (see for example Horowitz & Hill (1990)), and is of low-cost, moderate accuracy, offering good linearly, temperature stability and common-mode rejection characteristics. It has the further advantage of requiring only one external resistor to program its gain. This is selected according to (B.2)

$$R_G = \frac{50.5k\Omega}{G-1} \tag{B.2}$$

[2]Analog Devices, One Technology Way, P.O.Box 9106, Norwood, MA 02062-9106, USA. Web: `http://www.analog.com`

where R_G is the gain-set resistor and G is the gain of the amplifier.

The numerator resistance of $50.5k\Omega$ arises from the sum of two integrated $25.25k\Omega$ resistors, laser-trimmed for precision. Analog Devices' data sheet for the AD622 claims that (B.2) is accurate to 0.5%. The resistance of the external resistor, R_G, was measured to be 489.35Ω, resulting in a gain of $G = 104.2 \pm 0.52$.

The proportional resistance change, from the output of the amplifier stage is found by combining (B.1) and (B.2),

$$\frac{\Delta R_{Sensor}}{R_{Sensor}} = \frac{1}{G}\left[-\frac{(1+r)^2}{r}\frac{\Delta V}{V_{in}}\right]. \tag{B.3}$$

In all subsequent measurements of $\frac{\Delta R_{Sensor}}{R_{Sensor}}$, the voltage driving the bridge network, V_{in}, was supplied by battery and measured to be 8.8V. The instrumentation amplifier's reference voltage was set to ground.

B.2 Piezoelectric Signal Conditioning

The conditioning circuit used for the piezoelectric sensor signal is shown in Figure B.2. The op-amp used here is the LMC6001 Ultra-Low Input Current Amplifier

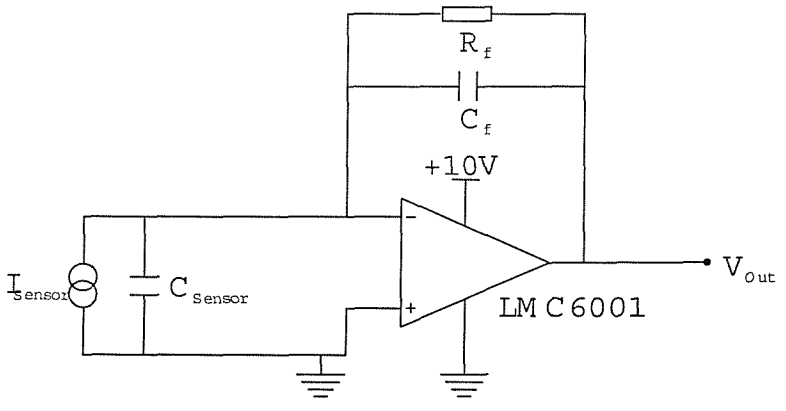


Figure B.2: Piezoelectric Sensor Signal Conditioning

from National Semiconductor[3], and is configured as an op-amp integrator. The piezoelectric sensor is modelled here as a current source with inherent capacitance,

[3]National Semiconductor, 2900 Semiconductor Drive P.O. Box 58090, Santa Clara, California USA. Web: `http://www.national.com`

as indicated by I_{Sensor} and C_{Sensor}. The virtual earth connected to the amplifier's non-inverting input, ensures that there is effectively no voltage drop across the sensor, hence the output voltage from the amplifier, V_o, is proportional to the charge developed by the sensor,

$$V_{out} = \frac{Q_{Sensor}}{C_f} \tag{B.4}$$

where Q_{Sensor} is the charge developed across the sensor and C_f is the capacitance of the feedback capacitor.

The feedback resistor, R_f, determines the rate of discharge of C_f, by providing a direct current path. R_f was measured to be 100.4MΩ whilst the capacitance of C_f was 470pF (to 5%).

B.3 Data Acquisition

Data from both sensors was captured using a Data Translation[4] DT9802 acquisition board. This board offers 12-bit sampling resolution of up to 16 single-ended (or 8 differential) channels at a maximum data throughput of 100k samples per second. The input voltage range across which the digitisation process occurs is user selectable from a number of predetermined ranges. In these experiments 12-bit digitisation occurred across 20V, between -10V and +10V. The conversion of digital values to voltage is performed simply, using

$$V_{Analogue} = \left[\frac{V_{Range}}{2^{BitResolution}}\right] V_{DigitalValue} - V_{Offset} \tag{B.5}$$

A Visual Basic program, employing Data Translation's ActiveX driver controls[5], was written to capture and store data.

[4]Data Translation, Inc. 100 Locke Drive, Marlboro, MA, USA. Web: http://www.datx.com
[5]DTx-EZ ActiveX controls for sampling and plotting, available from Data Translation

Appendix C

Piezoresistive Signal Conditioning Circuit

The following analysis describes the signal conditioning circuit used to measure resistance change of piezoresistive sensors. In this instance, the implementation is a variation of the classic resistive (Wheatstone) bridge circuit, comprising of four resistors connected through two parallel branches, driven by a constant input voltage. The output voltage from the bridge is measured between points B and D as shown in Figure C.1. R_1, R_2 and R_3 are of fixed resistance, whilst the resistance of the sensor, R_{Sensor} is allowed to change.

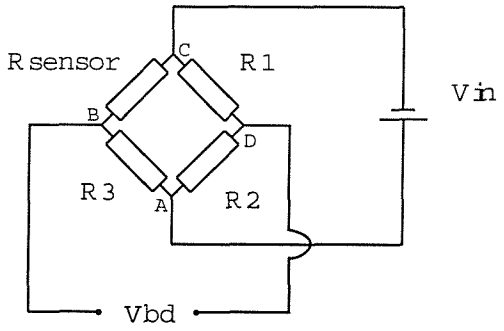


Figure C.1: Classic Resistive Bridge

Clearly the voltage difference between points A and C is independent of route,

hence V_{ABC} and V_{ADC} are the same, and equal to the input voltage, V_{in};

$$V_{in} = V_{ABC} = V_{ADC} \tag{C.1}$$

For a fixed input voltage, V_{in}, the branch currents through points ABC and ADC depend upon the resistance encountered:

$$I_{ABC} = \frac{V_{in}}{(R_3 + R_{Sensor})} \tag{C.2}$$

$$I_{ADC} = \frac{V_{in}}{(R_2 + R_1)} \tag{C.3}$$

It follows that the potential differences from point A to B and A to D are given by the potential divider equations;

$$V_{AB} = I_{ABC}\, R_3 = \frac{R_3}{(R_3 + R_{Sensor})}\, V_{in} \tag{C.4}$$

$$V_{AD} = I_{ADC}\, R_2 = \frac{R_2}{(R_2 + R_1)}\, V_{in} \tag{C.5}$$

The voltage from B to D is hence the difference between C.4 and C.5;

$$V_{BD} = V_{AB} - V_{AD}$$

$$\Longrightarrow \qquad V_{BD} = \frac{R_3\, R_1 - R_2\, R_{Sensor}}{(R_3 + R_{Sensor})(R_2 + R_1)}\, V_{in} \tag{C.6}$$

If the resistance of the piezoresistor is allowed to change from R_{Sensor} to $(R_{Sensor} + \Delta R_{Sensor})$, then the change in output voltage, ΔV is given by

$$\Delta V = \left[\frac{R_3\, R_1 - R_2\, (R_{Sensor} + \Delta R_{Sensor})}{(R_3 + R_{Sensor})(R_2 + R_1)} - \frac{R_3\, R_1 - R_2\, R_{Sensor}}{(R_3 + R_{Sensor})(R_2 + R_1)}\right] V_{in}$$

$$\Longrightarrow \Delta V = -\left[\frac{R_2\, \Delta R_{Sensor}}{(R_3 + R_{Sensor})(R_2 + R_1)}\right] V_{in} \tag{C.7}$$

If resistor values are chosen such that the bridge is initially in balance, ie. $V_{BD} = 0$ then from C.6, $R_3\, R_1 = R_2\, R_{Sensor}$ or

$$\frac{R_{Sensor}}{R_3} = \frac{R_1}{R_2} = r \tag{C.8}$$

Making use of C.8, C.7 reduces to

$$\Delta V = -\frac{r}{(1+r)^2}\left[\frac{\Delta R_{Sensor}}{R_{Sensor}}\right](1+\eta)\, V_{in} \tag{C.9}$$

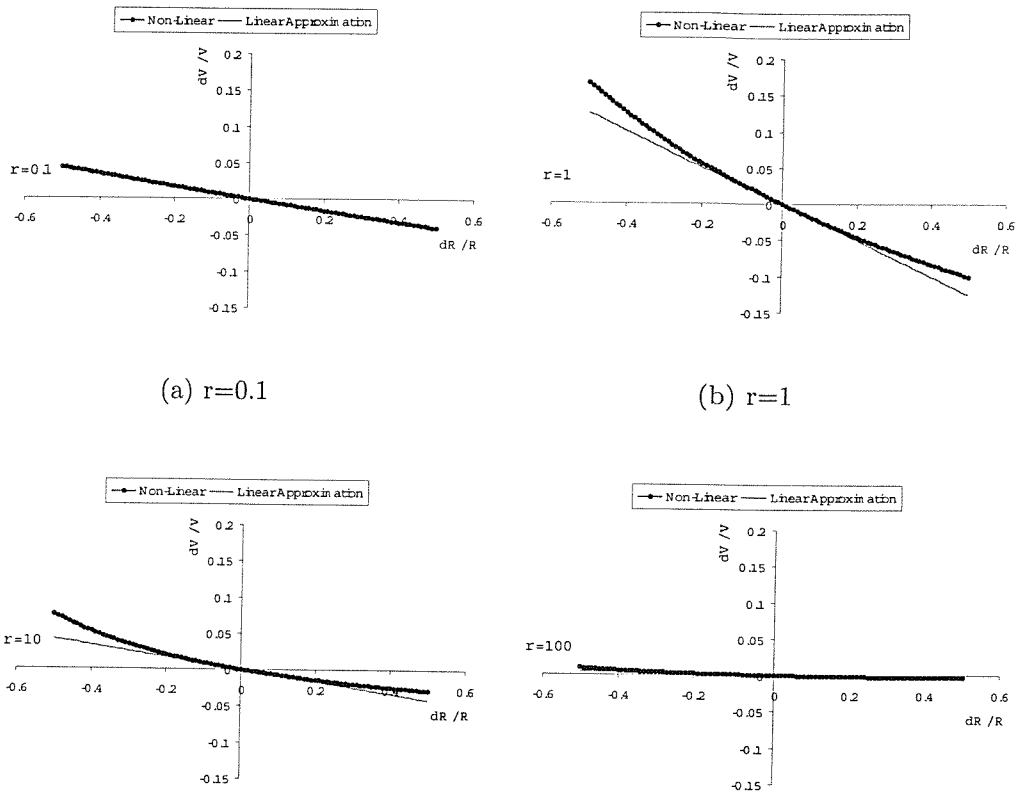
where

$$\eta = \frac{1}{1+a}, \quad \text{and } a = \frac{(1+r)}{r\left(\frac{\Delta R_{Sensor}}{R_{Sensor}}\right)} \tag{C.10}$$

If $\Delta R_{Sensor} \ll R_{Sensor}$, then $a \rightarrow \infty$ and $\eta \rightarrow 0$. Under such a condition C.9 collapses to

$$\Delta V \approx -\frac{r}{(1+r)^2}\left[\frac{\Delta R_{Sensor}}{R_{Sensor}}\right] V_{in} \tag{C.11}$$

Figure C.2 plots the proportional change in voltage, $\frac{\Delta V}{V_{in}}$, against proportional change in sensor resistance, $\frac{\Delta R_{Sensor}}{R_{Sensor}}$, for both the approximate linear and the exact non-linear descriptions. These curves are derived from equations C.11 and C.9 for four bridge resistance ratios, $r = 0.1$, 1, 10 and 100. The approximation C.11 is generally considered valid for resistance changes of $\lessapprox 5\%$ (See for example (Jones & Chin 1983)). This is illustrated in figures C.2(b) and C.2(c)showing clearly the non-linear deviation from C.11 as ΔR_{Sensor} increases, whilst figures C.2(a) and C.2(d) exhibit reduced sensitivity to change in sensor resistance.



(a) r=0.1

(b) r=1

(c) r=10

(d) r=100

Figure C.2: Linearity with Resistor Ratio

Appendix D

Data Acquisition Circuit

D.1 Overview

This appendix provides a detailed description of the data acquisition circuit used to capture the pressure sequences described in Chapter 5.

Figure D.1 provides the hardware detail for this circuit. As indicated in Chapter 5, the circuit is based around Microchip's 8-bit 16F84 Peripheral Interface Controller (PIC). The task of the PIC is the timely initiation of the analogue to digital conversion process, performed using Linear Technology's TLC1285 12-bit successive approximation ADC. Upon digitisation of the analogue signal, the PIC's next task is to transmit the data to a host PC. This is performed across a 2-wire serial link, whose protocol is implemented in software. The sampling through-rate of this circuit was set to 2kHz, enabling detailed capture of pressure response characteristics.I t is the detail of the timing, sampling and communication functions which form the subject of this appendix.

Before considering the software detail of the microcontroller's operation, some thought must be given to initialising the input and output ports of the PIC. The 16F84 has two bi-directional I/O ports; port A and port B offering 5 and 8-bit ports, respectively. Their direction is software selectable, depending upon the bit state of specific control registers. These registers are called *TRISA* and *TRISB*, respectively. Table D.1 shows the direction of the IO registers.

IO Register	Function	Direction
RA0	na	na
RA1	na	na
RA2	Indicator LED	Output
RA3	Indicator LED	Output
RA4	Indicator LED	Output
RB0	Interrupt Switch	Input
RB1	I^2C Communication Data	Input/Output
RB2	I^2C Communication Clock	Input/Output
RB3	Serial Communication Tx	Output
RB4	Serial Communication Rx	Input
RB5	ADC Clock	Output
RB6	ADC Data Out	Input
RB7	ADC $\overline{ChipSelect}$	Output

Table D.1: PIC IO Registers Summary

The physical pin numbers for each register can be found by reference to Figure D.1.

Registers RA2-RA4 supply the output indicator signals, these are used primarily for status indication and as an aid to debugging during software development. RB0 enables external signals to trigger software interrupts, allowing program flow to be modified during run-time. In this case RB0 is configured as an input and is connected to a push button switch, whose terminal is pulled-down by a 470Ω resistor, preventing the input from floating.

Bit registers RB1 and RB2 are used to implement an I^2C 2-wire serial interface bus[1] between the PIC and an EEPROM[2] memory module. The EEPROM is not used in this implementation, but is included to provide flexibility for future applications. No further details of the I^2C protocol will be given here, however, the interested reader is directed to the reference below.

RB3 and RB4 implement the serial output and input communication lines, respectively, whilst RB5-RB7 implement the 3-wire control of the ADC device. The detailed function of these bit registers will be described subsequently.

Listing D.1 gives the detail of the C-header file for the main program code, pro-

[1]Inter IC 2-wire communication protocol. Devised by Philips Inc. See for example, http://www.semiconductors.philips.com/i2c/facts/

[2]Electrically Erasable Programmable Read Only Memory, 24LC16B from Microchip Technology Inc. 2355 West Chandler Blvd. Chandler, Arizona, USA 85224-6199. Web: http://www.microchip.com

viding name and constant definitions used in subsequent code listings.

Listing D.1: Acquisition Code - Header File

```
//  'logger.h'
//
//  All definitions and constants for logger project.
//  Relevant parts extracted from function files.
//
//  Neil Henderson  7/6/'99

#define XTAL        4000000     // Xtal frequency (Hz)  (serial.c)
#define BRATE       38400       // Baud rate (Hz)       (serial.c)

#define EEPROM_Address  10      //24LC16B address.

#define Decision_Time       2000    //Time (mS) allowed for second RB0 press
#define Comms_Flash_Period  1000    //Time (mS) for LED flash cycle during comms_mode.
#define Comms_LED_ON        300     //Duration (mS) for LED on during comms flash cycle.
#define sp_addr             1       //The starting location of sample period (pic's eeprom)

#define ON      =1
#define OFF     =0

#define RB0InterruptLED     RA0         //Indicate RB0 interrupt
#define timingInterruptLED  RA1
#define decisionModeLED     RA2
#define decisionTimeLED     RA3
#define sampleLED           RA4

#define     Int_button      RB0     //Button press interrupt    (pin 6)
#define     SDA             RB1     //i2c serial data line      (pin 7)
#define     SCL             RB2     //i2c serial clock line     (pin 8)
#define     TxData          RB3     //Serial Tx Data            (pin 9)
#define     RxData          RB4     //Serial Rx Data            (pin 10)
#define     Clk             RB5     //ADC Bus Clock             (pin 11)
#define     Din             RB6     //ADC Data in               (pin 12)
#define     CS              RB7     //ADC Chip Select           (pin 13)

#define     IButton_Direction   TRISB0
#define     SDA_DIR             TRISB1
#define     SCL_DIR             TRISB2
#define     TxDirection         TRISB3          // Serial Tx Direction register
#define     RxDirection         TRISB4          // Serial Rx Direction register
#define     ADC_Clock_Dir       TRISB5
#define     ADC_Data_Dir        TRISB6
#define     ADC_CS_Dir          TRISB7
```

D.2 Microcontroller Timing

The 16F84 has built-in timing interrupt sources. The primary timer - an 8-bit register (called *TMR0*) - increments in step with the processor's instruction cycle, which in-turn is determined by the external clocking frequency. As a result of the PIC's architecture, the actual instruction cycle consumes four external clock

cycles, meaning that the instruction rate is one quarter of the clocking frequency, in this case 1MHz. The 16F84 offers the possibility of using either an oscillator crystal or a simple RC circuit to provide the external clocking signal. For stability, an oscillator crystal was used.

The main data acquisition program relies upon the PIC's timing interrupt handler, for the desired 2kHz sample rate. Whilst the instruction cycle at 1MHz rate is 1μs, the sampling interrupt is required to trip every 500μs. This is achieved by means of a timing prescaler, whose function is to delay the incrementing of TMR0 by a predefined number of instruction cycles. To arrive at an interrupt time of 500μs the prescaler was set to a ratio of 1:2, such that every second instruction cycle causes the TMR0 register to increment, and an initial value of -250 was loaded into TMR0. When enabled, the timing interrupt is initiated when TMR0 changes to 0.

Prescaler characteristics are determined by selective bit manipulation of the *OPTION* control register. For further details see the 16F84 data sheet from Microchip. Details of this specific implementation are given below, in the code listing for the main acquisition program (listing D.2).

Listing D.2: Acquisition Code - Main Program

```
//  'loggerv2.c'
//
//  Data Acquisition Development program.
//
//  Include file - 'logger.h' - works with;   delay_1.c   7/6/'99
//                                            serial_1.c  7/6/'99
//  This is Version 2.0   14/12/'99
//
//  Cut Down version of Loggerv1.0 - allowing only sample to PC
//  functionality at 500Hz sample rate
//
//  NOTE::: Serial PIC->PC comms set for 38400 Baud transmission.
//          Sample Rate Increased and Reliable to 2kHz.
//
//          Version 3.0 NH  11/1/2000

#include    <pic1684.h>     //Standard pic registers
#include    "logger.h"      //Definitions and constants for Acquisition program
#include    "serial_1.c"    //Serial comms functions - Modified
#include    "delay_1.c"     //Delay functions - Modified.
#include    "adcfs.c"

//**********Globals from here.................

    const sample_period = 1;            //Total Sample Period = sample_period x 500us
```

```
    unsigned long   event_counter;      //Tracks the number of timing interrupts
    unsigned char   sample_NOW;         //Flag - true when time to sample


//**********Interrupt handler from here.........

static void interrupt isr(void)
{
    if(INTCON,TOIF)                         //..then counter's ticking....
    {
        timingInterruptLED  ON;

        if(--event_counter==0)              //...then it's time to sample!
        {
            sample_NOW=1;                   // = TRUE - Go Sample!
            event_counter = sample_period;  //Reload event trigger...
        }

        //****Finish up gracefully..................

        TMR0 -= 250;        //reload timer with -250 * 1uS 'ticks'
        INTCON,TOIF = 0;    //...and clear flag.

        timingInterruptLED OFF;

    }//And that's the timer tick dealt with.
}//interrupts dealt with.

//**********Main from here.....................

main(void)
{
    //Variables from here....

    unsigned char   data_store[2];      //2byte array - store ADC return data
    unsigned char* data_pointer;        //pointer to character array.

//********User Response loop follows.........................

    //Set up interrupts.............

        //Timer interrupt....

    OPTION, PSA = 0;        //Assign PreScaler to timer.
    OPTION, TOCS = 0;       //Clock Source is internal instruction clock.

    OPTION, PS0 = 0;        //Set up prescaler ratio...
    OPTION, PS1 = 0;        //
    OPTION, PS2 = 0;        //...Prescaler is 1:2 - Interrupt every 2us.

    INTCON,TOIF = 0;        //Clear interrupt flag.

         //Interrupt masks...

    INTCON,GIE = 1;         //Enable all unmasked interrupts

    //Set up PORTS and TRIS registers......

    PORTA = 0;              //All OFF!
    TRISA = 0;              //All outputs

    PORTB = 0;              //All OFF.
    TRISB = 0;              //All outputs...

       //Set up serial link...
```

```
    RxDirection = 1;        //Rx - input...
    TxDirection = 0;        //Tx - output.

    OPTION,RBPU = 0;        //...Enable Weak pull-ups on input pins...
    RxData = 1;             //Serial Rx HIGH...
    TxData = 1;             //Serial Tx High...

        //Set up ADC Bus...

    ADC_Clock_Dir = 0;      //Clock is generated by pic - output...
    ADC_Data_Dir = 1;       //Data from ADC - input...
    ADC_CS_Dir = 0;         //Controlled by pic - output.

    data_pointer = data_store;      //Point to 2byte array

    event_counter = sample_period;  //Number of 250 uSeconds between each sampling event
    INTCON,TOIE = 1;                //Enable timing interrupts


while(1)    //Main Routine - keep it going....
    {
        if(sample_NOW)
        {
            decisionTimeLED ON;

            INTCON,TOIE = 0;            //Disable timing interrupt - but timer continues 'ticking'
            GetADCData(data_pointer);   //Initiate ADC

            decisionModeLED ON;

            putch(*data_pointer);       //Serial transmission of first byte->PC

            decisionTimeLED     OFF;
            decisionModeLED     OFF;

            INTCON,TOIE = 1;    //Reenable timing interrupt
            sample_NOW = 0;
        }
    }
}//End main()
```

The acquisition code structure, is straightforward. After initialising IO ports, timing interrupts and sampling parameters, the program goes into an infinite loop, doing nothing until the sampling interrupt occurs. At this point the PIC initiates execution of the analogue to digital conversion, retrieves the data from the ACD, then transmits this across a serial link to the host PC. A small Visual Basic Program, running on PC, collects and stores the data. This will be dealt with at the end of this appendix.

D.3 Serial Port Implementation

Communication between PIC and PC is performed by the implementation of an asynchronous serial protocol in software. In this method, an eight bit data byte is sandwiched between start and stop bits. Bits are transmitted in order of LSB to MSB, the duration of which is controlled by software generated delays. The delay length for each bit is calculated, in terms of instruction cycles, at compile time. Parity checking and handshaking techniques are not employed.

From the circuit diagram of Figure D.1, it can be seen that between the PIC and PC there is a 232-TTL level converter. This converts the RS-232 voltage levels (+5 to +15V for logic LOW and -5 to -15V for logic HIGH), to CMOS/TTL levels (+5V HIGH and 0V LOW). Figure D.2 shows the TTL and RS-232 levels, for the transmission of one byte of value 150 (base 10).

The data transfer rate was set to 38400 Baud, under which the transmission of one byte takes 330μs to complete. In this software implementation, faster transfer rates were not reliably achievable. The asynchronous nature of this protocol relies upon independent clocks at both PC and PIC running at precisely the same rate, making faster communications unreliable between these devices.

Sampling at the required rate of 2kHz, requires that both data acquisition and data transmission processes complete within 500μs. Taking the naïve view that the efficient transmission of 12-bit ADC values requires that two (12-bit) words be packaged into three transmission bytes, the total transmission time for two 12-bit words would be 990μs, or 495μs per word. Improving the timing constraints by considering only the bit-transmission time running at 38400Baud, and imagining a constant bit-stream protocol, transmission of each bit would require 36.67μs or transmission of a full 12-bit word would require 440μs. This leaves only 60μs to initiate and capture the full 12-bit data word from the ADC. Clearly, the serial data transmission is the limiting factor in the data throughput.

The timing situation can be improved by considering only the first 8-bits of the ADC's digital signal. This is justified in that, as indicated in Chapter 5, the maximum conversion error of the ADC device is $\pm 8 \times$ LSB. Considering only the first 8-bits of conversion data has the effect of increasing the maximum error from 0.2% to 0.37%. As indicated the transmission of one byte requires 330μs allowing

a maximum of 170μs for the conversion process. In this instance the reduction of precision due to discarding the 4 Least Significant Bits, is justified against the improved timing constraints.

Simple loop delays are used to generate the time required for each bit transmission. These are generated by firstly calculating the number of instruction cycles required for each bit and subtracting from this the number of instructions used to set up and execute the loop test conditions. This number is then divided by the number of instruction cycles used during each loop. This is the number of times the wait loop has to execute to generated the required bit-time. Code Listing D.3 provides the C-code used to implement serial transmission.

Listing D.3: Serial Transmission Function

```
/*
 *  Serial port driver for 16Cxx chips
 *  using software delays.
 *
 *  Copyright (C)1996 HI-TECH Software.
 *  Freely distubutable.
 */
//  Based around HI-TECH's serial functions - modified for clarity.
//  All 'tuneable' parameters removed to 'logger.h'
//
//  NH  7/6/'99

#include    <conio.h>

#define DLY     3           /* cycles per null loop */
#define TX_OHEAD    13      /* overhead cycles per loop */
#define DELAY(ohead)    (((XTAL/4/BRATE)-(ohead))/DLY)

void
putch(char c)
{
    unsigned char   dly, bitno;

    TxDirection = 0;            //Replaces 'INIT_PORT' (NH)

    TxData = 0;                 /* start bit */
    bitno = 8;
    do {
        dly = DELAY(TX_OHEAD);  /* wait one bit time */
        do
            /* nix */ ;
        while(--dly);           //Until delay is complete - dly=0
        if(c & 1)               // bit value TRUE??
            TxData = 1;         //Transmit HIGH
        if(!(c & 1))            // bit value NOT (TRUE)??
            TxData = 0;         //Transmit LOW
        c = (c >> 1) | 0x80;    // bit-shift data by one place.
    } while(--bitno);

    TxData = 1;                 //Stop-bit on transmission line.
}
```

The code used in this serial implementation was modified from Hi-Tec's library file *serial.c*, which can be found in the samples directory of their compiler installation.

D.4 Control of the ADC

Communications between PIC and ADC are synchronous, with the PIC supplying the required clock pulses. There are three essential stages to this protocol.[3] Firstly the PIC must initiate conversion by driving the ADC's -Chip Select (-CS) pin LOW. Secondly, the PIC provides fifteen clock pulses, of which, the first three are used for set up purposes, the remaining twelve each causing a data bit to be expressed on the ADC's data out pin. During this period the PIC must retrieve and store each data bit as it arrives. Thirdly, the PIC must complete the conversion process by driving -CS HIGH. The timing diagram is shown schematically in Figure D.3. The clock period was 5μs, leading to an overall sample time of 75μs. A delay of 5 μs is consumed by only 5 instruction cycles and as such was implemented using fixed non-significant instructions. Again because of the extremely short clock cycle, the code was implemented as a series of explicit bit assignments, rather than being rolled into loops. Code listing D.4 provides the implementation.

Listing D.4: Control of ADC

```
//  ADC.c
//
//  Contains code to drive ADC.
//  Data is referenced through int array pointer -
//  declare this before use.
//
//  Neil Henderson  20/5/'99
//
//  changed to use char* rather than int*
//  NH  16/6/'99.
//
//
//
//  *************************
//
//  ADCfs.c     (ADC fast sample)
//
//  Bitlengths reduced to improve sample time.
//  NH  13/1/2000
//
```

[3]For further details see Linear Technology's data sheets for the TLC1285. These are available from http://www.linear.com

```
//  Functionality extracted from nested loops - Loops unraveled to improve speed.
//  NH  13/1/2000
//

void GetADCData( char* datapointer )

    //Function to initiate ADC and store returned data in two byte array.

{
    signed char cpulse;     //count pulses generated.
    char dbyte1, dbyte2;    //Store data retrieved from ADC
    char wastetime;

    CS = 0; //-CS Low - initiate ADC
    dbyte1 = 0;

    //Now state all transitions explicitly - try and minimize time requirements...
    Clk = 0;        //**********************
    wastetime = 0;
    Clk= 1;
    wastetime = 1;
                    //  |
    Clk = 0;        //  |
    wastetime = 0;  //
    Clk= 1;         // Initial 3 pulse cycles
    wastetime = 1;  //
                    //  |
    Clk = 0;        //  |
    wastetime = 0;
    Clk= 1;
    wastetime = 1;  //**********************


    //Collect only the MSByte - disregard LSByte data....

    Clk = 0;                    //*************************
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte MSB
    {
        dbyte1++;               //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B6
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B5
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

   Clk = 0;
   dbyte1 = dbyte1 <<1;         //bitshift left
   if (Din)                     //MSByte B4
```

```
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B3
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B2
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B1
    {
        dbyte1++;   //Add Din as LSB
    }
    Clk = 1;
    wastetime = 1;

    Clk = 0;
    dbyte1 = dbyte1 <<1;        //bitshift left
    if (Din)                    //MSByte B0
    {
        dbyte1++;   //Add Din as LSB
    }

    Clk = 1;
    wastetime = 1;              //**********************

    //MSByte collected - finished with ADC.
                //Store data in external array

    *datapointer = dbyte1;      //MSB -> B4

    CS = 1;     //Finished with ADC Bus
}
```

The total time required for sampling and data transmission was 420ms. Sampling at 2kHz requires that both processes finish within 500ms, leaving no room for improvement in the sample rate, nor transmission of the ADC's full 12bit range. This implementation has reached its limit and is inappropriate for further development.

D.5 PC Data Receive Code

A simple Visual Basic program was written to retrieve data transmitted from the PIC. The core of which loops until the user issues a stop command, each time retrieving a single transmitted byte from the PC's serial buffer and saving to file. The code for this core function can be found in listing D.5.

Listing D.5: Retrieve Data from Serial Buffer

```
Private Sub Cmd_Start_Click()

''Subroutine to Retrieve and store each byte from the serial port's buffer.

    Dim CommData As Variant
    Dim dataByte() As Byte
    Dim Person As String

    Sampling = True

    GraphForm.Hide                          ''Disable All buttons except STOP...
    Cmd_Start.Enabled = False
    Cmd_Stop.Enabled = True
    Cmd_FinishCollecting.Enabled = False
    CardOnDesk_Option.Enabled = False
    CardOnMat_Option.Enabled = False
    CardInHand_Option.Enabled = False

    Call UpdateStats                        ''Add 1 to relevant sample type
    Person = MainForm.PersonName + TodaysDate() + SampleQualifier()

            ''Now generate Filename from Person's Name,Date, Option +Number

    Filename = MainForm.DirectoryPath + Person + ".txt"

    NameLabel.Caption = Person

    outputfile = FreeFile                           ''Open File...
    Open Filename For Output As #outputfile

    MSComm1.PortOpen = True                         ''Open Comm Link...

    While (Sampling)
        DoEvents

        Do
            DoEvents

        Loop Until ((MSComm1.InBufferCount > 0))    ''We've got a character..!

        CommData = MSComm1.Input
        dataByte = CommData

        If (Sampling) Then                          ''User hasn't pressed [Stop]
            Print #outputfile, CByte(dataByte(0))
        End If
    Wend

    Close                                           ''any open files
```

```
    MSComm1.PortOpen = False

End Sub
```

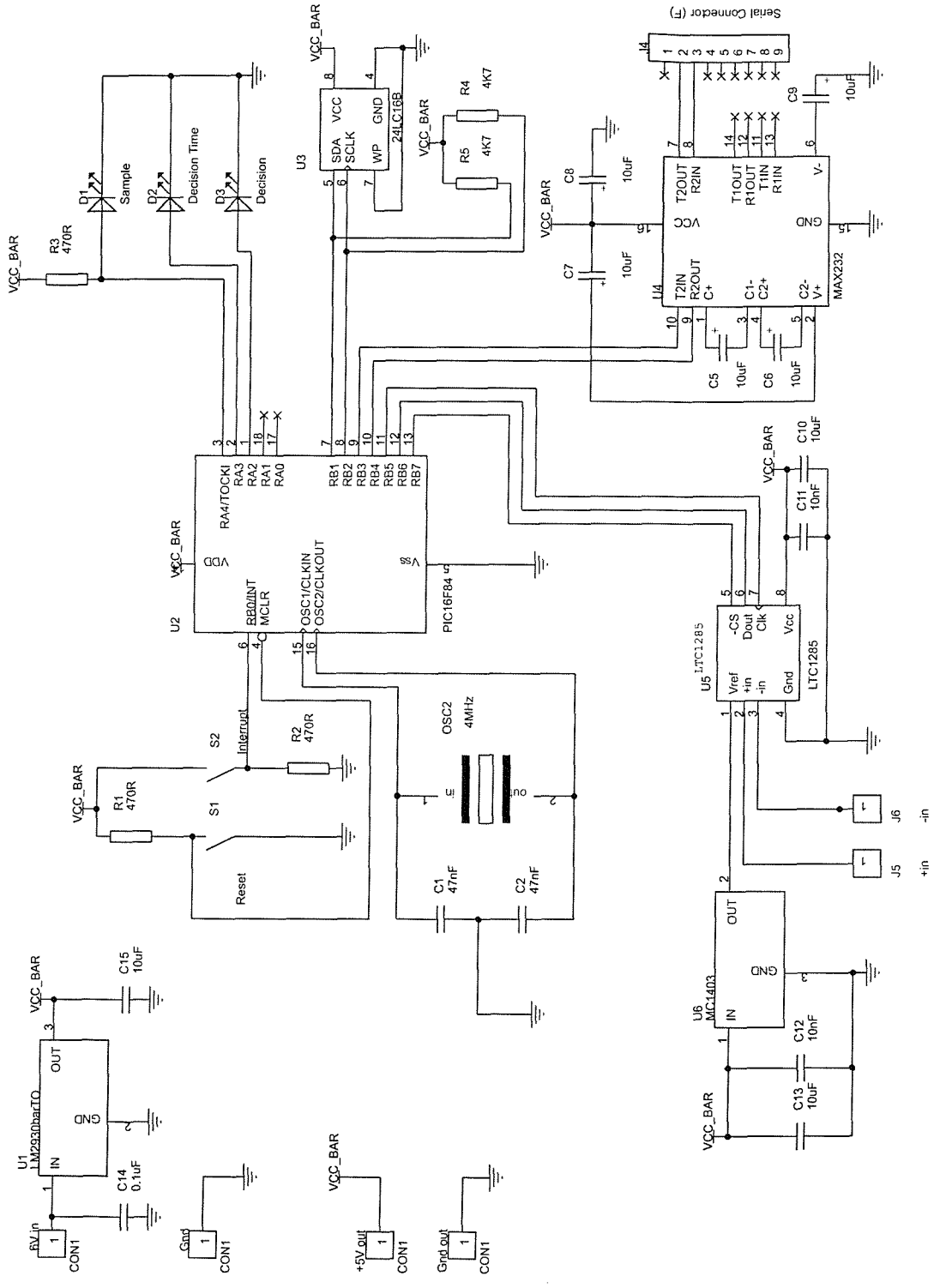


Figure D.1: Data Acquisition Schematic

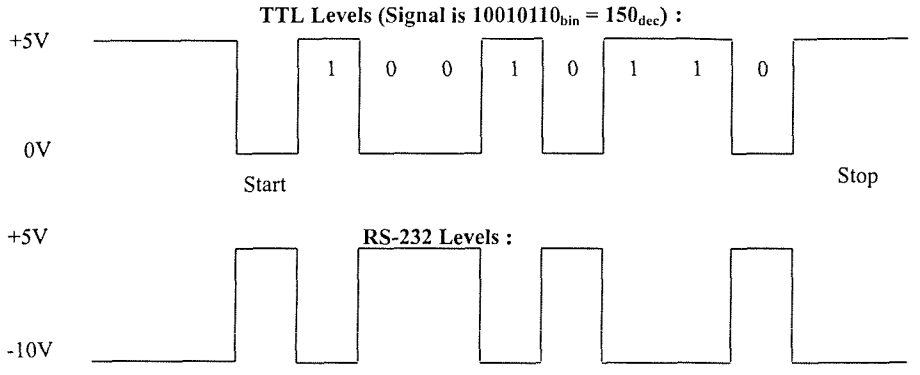


Figure D.2: Serial Transmission Protocol

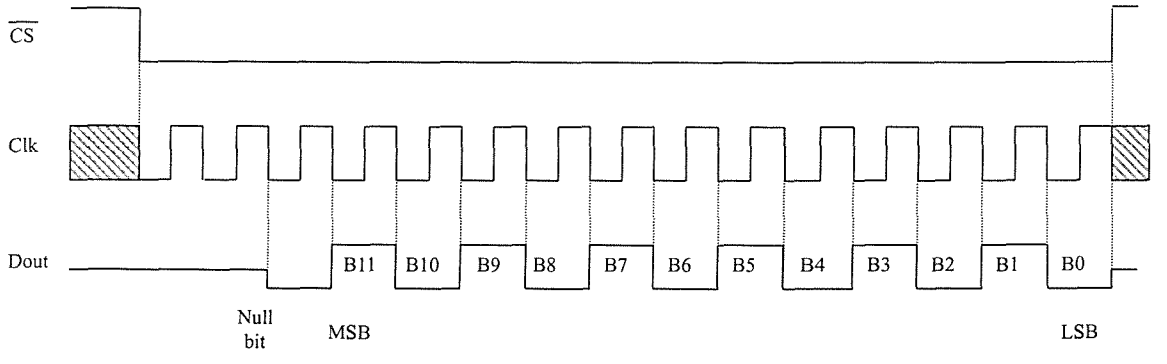


Figure D.3: ADC Timing Requirements

Appendix E

Feature Extraction Algorithm

E.1 Introduction

The pressure sequence method involves capturing the analog response of a participant tapping rhythms upon a piezoelectric pressure sensor, this is outlined in detail in Chapter 5. Figures 5.3 to 5.10 show captured sequences, sampled at 2kHz, from 16 different participants.

Much of the work involved in demonstrating the viability of the pressure sequence method is to show its discrimination potential. For this, some method of quantitively comparing sequences must be sought. Rather than comparing raw digitised sequences, pertinent characteristics may be extracted and used as a basis for comparison, this is known as *feature extraction*. Visual inspection of captured sequences with the same number of pulses (presented in Chapter 5) show differences between sequences in terms of pulse amplitudes, pulse widths and the time intervals between pulses. It is, hence, upon these characteristics that the verification functions of Chapter 5 are based.

For example, table E.1 provides this feature representation corresponding to figures 5.3(a) and 5.3(b). Both sequences have three pulses and originate from two participants - participant 'A' and 'B', respectively. Since the components of a feature vector are dimensionless, units of each component are purposefully omitted. For interest, however, pulse amplitudes are presented in terms of digital steps, whilst pulse widths and interval durations are given in terms of sample periods.

Participant	Feature Vector - Components							
	$Amplitude_1$	$Width_1$	$Interval_1$	$Amplitude_2$	$Width_2$	$Interval_2$	$Amplitude_3$	$Width_3$
'A'	78	90	568	66	79	213	71	99
'B'	132	93	407	156	107	811	203	133

Table E.1: Comparison of two feature vectors

The component subscripts of table E.1 refer to the pulse number with which they are associated.

Since there are two features associated with each pulse (*pulse amplitude* and *pulse width*) and one interval between two pulses, a sequence of n pulses will result in a feature vector of $(3n - 1)$ components.

It is with these extracted features, combined to form a feature vector, pressure sequences will be compared. From here on the term *sequence* (in the context of "...sequence X...") will refer to unprocessed analog signals, whilst *vector* (in similar context) will refer to feature vectors derived from sequences.

E.2 Implementation Details

The feature extraction algorithm in this implementation simply searches across a raw data file looking for data crossing of a noise threshold. This threshold is set to an ADC value of 5, in this case equivalent of around 49mV. Once data crosses the threshold, the algorithm traverses these super-threshold values, comparing subsequent values to previous, ascertaining the pulse's peak level. A counter, measuring pulse width, is initiated upon positive threshold crossing, and incremented for each data point above the noise threshold.

This level of complexity would be sufficient to characterise a simple pulse, rising above the nose threshold, reaching some peak level, then returning to sub-threshold levels.

However, many of the pulses captured, do not follow elementary trajectories. Participant I's third pulse, described in Figure 5.11, is an example of a more complex form. In this, the sensor's output voltage rises to a peak then falls to a local minima, rises to a local maxima before finally returning to ground. If the voltage minima was to drop below the noise threshold, then the above algorithm would

characterise this response as two separate pulses.

An extreme example of this form, from another user - User 'Q' - is shown in Figure E.1. In reaching the local minima, the sensor is generating charge in the opposite sense, which is masked by the charge amp's 0V supply line (See Appendix D and section B.2 for circuit diagram). No choice of positive noise threshold would allow this response to be characterised as a singular pulse, using the outlined algorithm. For the feature extraction algorithm to cope with this form of pressure

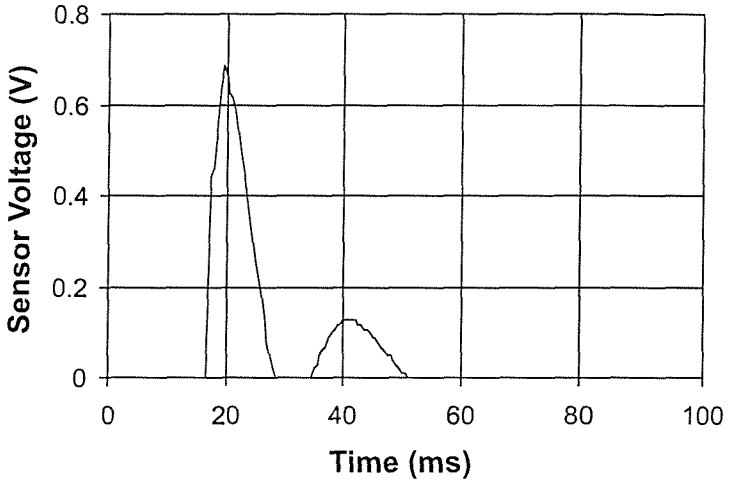


Figure E.1: Participant Q - Pulse through Zero

response, it must *look-forward* for a short interval. If, within this short time interval, the raw data stream returns above the noise threshold, then this region will be considered part of the same pulse, otherwise, the pulse will be assumed to have reached completion upon the last negative crossing. The overall pulse width will be measured from the first positive threshold crossing, to the negative crossing beyond which the data does not return above threshold, within the look-forward time interval.

If the look-forward interval is too short then the second part of the response will be considered a separate pulse. Too long, on the other hand, and subsequent pulses will be considered part of previous pulses. A suitable look-forward interval was chosen by considering both the shortest inter-pulse interval and the longest intra-pulse sub-threshold interval. From captured pressure sequences, these were found to be 64.5ms and 22.5ms, respectively. Taking an interval approximately halfway

between shortest inter-pulse and intra-pulse sub-threshold times, the look-forward interval was set to 45ms.

E.3 Visual Basic Implementation

The following code sections implement the feature extraction approach outlined above. Listing E.1 provides detail of the main program structure, whilst listing E.2 contains the essence of the feature extraction algorithm. Listing E.3 is called by the main algorithm when data crosses the noise threshold. This code function characterises the pulse in terms of amplitude and width. The other functions within listing E.4 contain ancillary file handling functions, and are included for completeness.

Code was implemented using Visual Basic V5.0.

Listing E.1: Feature Extraction - Main

```
Option Explicit
''********* Constants from here...
Const PulseThreshold = 5     '' = ~49 mV ... This is the noise threshold.
Const PulseEndGap = 90       '' = 45 ms  ... This is the 'look forward' time interval

Const DataDirectory = "c:\My Documents\Pressure Biometric\ _
                                        Captured Data\Enrolment Sequences\"
Const ResultsDirectory = "c:\My Documents\Pressure Biometric\ _
      Results\ComparativeAnalysis\Enrolment Sequence Summaries - LowThreshold\"


''********* Globals from here...
Private RawDataFileno As Variant
Private ResultsFile As Variant
Private FileList(2000) As String


Private Sub cmdExtractFeatures_Click()

    '' Procedure sequentially steps through all raw pressure sequence files
    '' in 'DataDirectory'
    '' Extracts Pulse Height, Pulse Width and interval length - then saves to file.


    Dim NumberofFiles As Integer            ''Number of files in DataDirectory
    Dim i As Integer                        ''loop/file counter
    Dim ExtractedFeatures As String         ''Extracted features held in string

    Dim FeatureFileno As Variant            ''File channel for output file

    Dim RawDataFilename As String
    Dim FeatureFilename As String
```

```
    NumberofFiles = GenerateFileList        ''Create a list - held in global 'FileList' -

    For i = 1 To NumberofFiles

        DoEvents
        RawDataFilename = FileList(i)           ''Select RawDataFile
        Label1.Caption = RawDataFilename
        RawDataFileno = FreeFile                ''Open File channel
        Open (DataDirectory + RawDataFilename) For Input As RawDataFileno

        ExtractedFeatures = FeatureExtraction   ''Extract Features from raw pressure Sequence

        FeatureFilename = ConvertFilename(RawDataFilename)
        FeatureFileno = FreeFile                                ''Open channel for Output
        Open ResultsDirectory + FeatureFilename For Output As FeatureFileno
        Print #FeatureFileno, ExtractedFeatures                 ''Print Extracted Features to file
        Close                                                   ''Close all file

    Next I

End Sub
```

Listing E.2: Feature Extraction Function

```
Private Function FeatureExtraction() As String

    ''File MUST be PREVIOUSLY OPENED through the [global] channel 'CurrentFile'

    Dim DataVal As Integer
    Dim IntervalWidth As Integer
    Dim NumberofPulses As Integer
    Dim BioSampleStarted As Boolean
    Dim JustReturnedFromPulse As Boolean
    Dim Analysis As String

    IntervalWidth = 0
    NumberofPulses = 0
    BioSampleStarted = False
    JustReturnedFromPulse = False
    Analysis = ""

    Do While Not (EOF(RawDataFileno))      ''Sift through data...

        DoEvents
        Input #RawDataFileno, DataVal

        If (DataVal >= PulseThreshold) Then       ''We've got a pulse

            If (BioSampleStarted) Then            ''We've just returned from an interval
                IntervalWidth = IntervalWidth + 1 ''Because this super-threshold value...
                                                  ''   closes the interval.
                Analysis = Analysis + CStr(IntervalWidth) + ";"
                IntervalWidth = 0
            End If

            Analysis = Analysis + PulseCharacteristics()
            NumberofPulses = NumberofPulses + 1
            BioSampleStarted = True
            JustReturnedFromPulse = True
```

```
        End If

        If (DataVal < PulseThreshold) And (BioSampleStarted) Then
            If (JustReturnedFromPulse) Then       ''Must add the interval gap read in...
                                                  ''  PulseCharacteristics()
                IntervalWidth = IntervalWidth + PulseEndGap
                JustReturnedFromPulse = False
            End If
            IntervalWidth = IntervalWidth + 1
        End If

    Loop

    FeatureExtraction = CStr(NumberofPulses) + ":" + Analysis

End Function
```

Listing E.3: Feature Extraction - Pulse Characteristics

```
Private Function PulseCharacteristics() As String

    Dim EndofPulse As Boolean
    Dim SuperThreshold As Boolean
    Dim PulseWidth As Integer
    Dim GapWidth As Integer
    Dim PulseHeight As Integer
    Dim DataVal As Integer

    EndofPulse = False
    SuperThreshold = False
    PulseWidth = 0          ''We've already crossed the threshold in calling function.
    GapWidth = 0            ''...so...each subsequent super-threshold value closes a
                            ''      pulse time interval.
    PulseHeight = 0

    Do While Not (EOF(RawDataFileno)) And Not (EndofPulse)
        DoEvents
        Input #RawDataFileno, DataVal
        If (DataVal >= PulseThreshold) Then  ''...we're on a pulse...

            If Not (SuperThreshold) Then     ''...we've just crossed a gap - update Pulse width
                SuperThreshold = True
                PulseWidth = PulseWidth + GapWidth
                GapWidth = 0
            End If

            PulseWidth = PulseWidth + 1
            If (DataVal > PulseHeight) Then
                PulseHeight = DataVal
            End If

        Else                                 ''...we're on a gap...
            SuperThreshold = False
            GapWidth = GapWidth + 1
            If (GapWidth = PulseEndGap) Then    ''We've reached the end of the pulse.
                EndofPulse = True
            End If
        End If
    Loop

    PulseCharacteristics = CStr(PulseHeight) + "," + CStr(PulseWidth) + ","
```

```
End Function
```

Listing E.4: File Handling Functions

```
Private Function GenerateFileList() As Integer

    ''Builds a list of all files in DataDirectory...

    Dim File As String
    Dim FileListString As String
    Dim NumberofFiles As Integer
    Dim i As Integer

    File = ""
    FileListString = ""
    NumberofFiles = 0

    File = Dir(DataDirectory)
    If File <> "" Then
        NumberofFiles = NumberofFiles + 1
        FileList(NumberofFiles) = File
    End If

    Do While File <> ""
        DoEvents
        File = Dir                          ''Get next file in dir...
        If File <> "" Then
            NumberofFiles = NumberofFiles + 1
            FileList(NumberofFiles) = File      ''Add to Global Array
        End If

    Loop
    Label1.Caption = CStr(NumberofFiles)

    GenerateFileList = NumberofFiles            ''Return the number of files in directory

End Function


Private Function ConvertFilename(Fname As String) As String

    ''Simple Function to convert Raw Data Filename to Feature Filename...

    Dim FnameLength As Integer
    Dim basename As String
    Dim NumberIndicator As String


    FnameLength = Len(Fname)
    basename = Left(Fname, FnameLength - 5)          ''Extract name before NumberIndicator
    NumberIndicator = Mid(Fname, (FnameLength - 4), 1)''Extract File number indicator

    ConvertFilename = basename + "F" + NumberIndicator + ".txt"  ''Build and return
                                                                 ''   Converted filename
End Function
```

Appendix F

Enrolment & Verification Functions

The enrolment and verification functions discussed at length in Chapter 5 were implemented using the MATLAB[1] technical computation package. MATLAB offers high level vector and matrix manipulation functions, making it inherently suitable for the analysis and direct comparison of pressure sequence verification functions.

F.1 Main Program Framework

In order to directly compare verification schemes, in terms of both error rates and computational requirements, a common infrastructure is required. To this end a framework program was written to support the tasks common to all verification methods. Explicit MatLab code for this program is provided in listing F.1. The essential code structure of which can be distilled to the following;

- Retrieve all impostor vectors (from file)
- For EACH participant:
  - Retrieve enrolment vectors
  - Call Enrolment function - Generate reference vector
  - For a number of Acceptance Thresholds:

[1]MATLAB version 5.3, available from The MathWorks Inc. 24 Prime Park Way, Natick, MA 01760-1500, USA. Web: http://www.mathworks.com

        * Call Verification function - Verify participant's test vectors
    - For all impostor vectors:
        * If impostor vector has same number of pulses as reference, then verify impostor

- Save results.

Listing F.1: MatLab - Main Program

```
%  Pressure Sequence - Enrolment/Verification Functions
%
%  Matlab Script File - Version 1.2 - Condensed to Functions
%
%  NOTE:   This is the Skeletal Code to perform both enrolment and
%          verification. Change Enrolment and verification functions
%          for each verifier
%
%  NH 10/8/'00

%%%%%%%%%%%%%%% Constants from here...
ResultsPath = 'C:\My Documents\Pressure Biometric\Results\MatLab Analysis\Results\';
ReferenceVectorDir = 'C:\My Documents\Pressure Biometric\Results\MatLab Analysis\
                                                          \Reference Vectors\';
TestSequenceDir = 'C:\My Documents\Pressure Biometric\Results\ComparativeAnalysis\
                                                    Test Sequence Summaries\';

extension = '.txt';
TestSummaryIndicator = 'TS';            %Self-Test Summary Filename indicator

NumberofUsers = 33;                     %Number of Enrolled Users


%%%%%%%%%%%%%%% Variables & Matrices from here...
EnrolmentFlops = zeros(1,2);        %[EnrolmentFlops, NumberofEnrolments]
VerificationFlops = zeros(1,2);     %[VerificationFlops, NumberofVerifications]

EnrolmentVectors = [0];             %Holds all eight Enrolment Feature Vectors
ImpostorVectors = [0];              %Holds ALL Impostor Feature Vectors
ComparisonResult = zeros(1,2);      %Return data from Comparison function
                                    %[Comparison Performed??, Accepted or Rejected]
Rejections = zeros(1,2);            %Running Tally of [False Rejections,No. comparisons]
Acceptances = zeros(1,2);           %Running tally of [False Acceptances,No.comparisons]

StartThreshold = 0;

NumberofThresholdSteps = 80;        %Number of Comparison Threshold Calculation Steps.
ThresholdStep = 0.25;               %Threshold Calculation Step Size.
MaxNumberofPulses = 15;
global Results;                     %Results array - 3rd dimension-pulse number pages.
Results = zeros(NumberofThresholdSteps,5,MaxNumberofPulses);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Transfer Impostor Feature vectors from File to Impostor MATRIX... %%%%%%%

disp('....Retrieving Impostor Vectors...');
ImpostorVectors = RetrieveImpostorVectors;      %Retrieve Impostor Vectors
disp('Impostor Feature Vectors - Retrieved!');

%%%%%%%%% Impostor Feature Vectors Retrieved  - Matrix Created. %%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%% Main Program Loop From Here ... %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for Person = 1:NumberofUsers                       %Go through all Participants

   PersonName = PeopleList(Person);                %Person to be dealt with.
   disp(['Enroling User ' int2str(Person)]);

   EnrolmentVectors = RetrieveEnrolmentVectors(PersonName);
   [m,n] = size(EnrolmentVectors);
   NPulses = (n+1)/3;


   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%%%%%%%%%%%%%%%%%%%%%%%%%% ENROL USER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   flops(0);                                           %Reset FLOP counter
   ReferenceProfile = Enrol(EnrolmentVectors);         %ENROL this Participant

   EnrolmentFlops(1) = EnrolmentFlops(1) +flops;       %operations required
   EnrolmentFlops(2) = EnrolmentFlops(2)+1;            %Number of enrolments performed

   SaveReferenceToFile(ReferenceProfile, PersonName);

   %%%%%%%%%%%%%%%%% ENROLMENT COMPLETED & Data Saved %%%%%%%%%%%%%%%%%%%%%%%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%%%%%%%%%%%%%%%%%%% Verification From Here...%%%%%%%%%%%%%%%%%%%%%%%%%%%%


   for thresh = 1:NumberofThresholdSteps

      Threshold = StartThreshold + (thresh-1)*ThresholdStep;
      disp(['Threshold :' num2str(Threshold)]);

      %%%%%%%%%%%% FALSE REJECTION RATE FROM HERE... %%%%%%%%%%%%%%%%%%%%%%%%

      Rejections = zeros(1,2);
      Acceptances = zeros(1,2);

      for i = 1:8         %Self-Test Feature Vectors....For False Rejection Rate

         basefilename = [PersonName TestSummaryIndicator int2str(i)];
         filename = [TestSequenceDir basefilename extension];          %Generate filename
         T=dlmread(filename,',');                              %Retrieve Test Feature Vector...

         flops(0);                                                     %Reset Flops counter
         ComparisonResult = Verify(T,ReferenceProfile,Threshold);

         if ComparisonResult(1)         % ...if Test and Reference Vectors have same no. pulses...
                                        %       ....So Test...
            VerificationFlops(1) = VerificationFlops(1) + flops;    %Update verification FLOPs
            VerificationFlops(2) = VerificationFlops(2) + 1;
            Rejections(2) = Rejections(2) + 1;                      %Another Comparison made...
            Rejections(1) = Rejections(1) + ~ComparisonResult(2);   %...and the result stored.
         end       % ... if branch.

      end %for Feature Vector loop.
```

```
      %%%%%%%%%% False Rejection Complete. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      %%%%%%%%%% FALSE ACCEPTANCE RATES FROM HERE... %%%%%%%%%%%%%%%%%%%%%

      [NumberofImpostors,components] = size(ImpostorVectors);

      for i = 1:NumberofImpostors

         ImpostorVector = ImpostorVectors(i,:);
         flops(0);
         ComparisonResult = Verify(ImpostorVector,ReferenceProfile,Threshold);

         if ComparisonResult(1)                    %Test and Mean vectors have same no.pulses
                                                   %  ....So Perform Test....
            Acceptances(2) = Acceptances(2)+1;                %Another comparison made...
            Acceptances(1) = Acceptances(1)+ComparisonResult(2); %...And the Result stored.
            VerificationFlops(1) = VerificationFlops(1) +flops;  %Update verification FLOPs
            VerificationFlops(2) = VerificationFlops(2) +1;
         end   %if

      end % Number of Impostors Loop.

      %%%%%%%%%% False Acceptance Complete. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Update Results Matrix...
      Results(thresh,1,NPulses) = Threshold;
      Results(thresh,2,NPulses) = Results(thresh,2,NPulses) + Rejections(1);   %No.of Rejections
      Results(thresh,3,NPulses) = Results(thresh,3,NPulses) + Rejections(2);   %Comparisons made
      Results(thresh,4,NPulses) = Results(thresh,4,NPulses) + Acceptances(1);
      Results(thresh,5,NPulses) = Results(thresh,5,NPulses) + Acceptances(2);

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Append to User Results File...
      FRR = (Rejections(1)/Rejections(2))*100;
      FAR = (Acceptances(1)/Acceptances(2))*100;
      SaveUserResults(PersonName, Threshold, FRR, FAR);

   end  %Threshold value Loop

   %%%%%%%%%%%%%%%%%%% Verification Complete.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end %Person Loop.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% END Main Program Loop. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% Finally - SAVE RESULTS... %%%%%%%%%%%%%%%%%%%%%%%%

SaveAllResults(Results, EnrolmentFlops, VerificationFlops, NumberofThresholdSteps,
                                        MaxNumberofPulses);
```

Enrolment and verification functions, specific to each verification method, were written and inserted into the above code. The next section considers the code implementation for each method.

F.2 Enrolment & Verification Code

This section provides the enrolment and verification functinos for each of the verification schemes presented in section 5.4. These code blocks slot directly into the preceding program code.

F.2.1 ℓ_1 & ℓ_2 norm Functions (Fixed Threshold)

Each participant's reference vector, $\mathbf{R}$ is constructed, simply, from the mean of each component of their enrolment vectors. This can be stated as follows:

$$r_i = \frac{1}{m} \sum_{j=1}^{m} e_{j_i} \tag{F.1}$$

where r_i is the i^{th} component of reference vector, $\mathbf{R}$, m is the number of enrolment vectors, $\mathbf{E}$ and e_{j_i} is the i^{th} component of the j^{th} enrolment vector.

MatLab provides high level vector functions, and so equation (F.1) is implemented for all components of $\mathbf{R}$ as follows (code listing F.2):

Listing F.2: Mean Component Enrolment

```
function ReferenceProfile = Enrol(EnrolmentVectors)

    %% Enrolment function - 'Enrol( ... )'
    %%
    %% Calculates the mean of each component from all Enrolment Vectors
    %% Returns the mean reference vector as the 'R' field of the 'ReferenceProfile'
    %% structure.
    %%
    %% l1, l2 norms and component-wise linear verification schemes.
    %%
    %% NH   12/8/'01.

    %%% Calculate Mean ...
    ReferenceProfile.R = mean(EnrolmentVectors);
```

The verification of a vector, $\mathbf{T}$, is performed using a vector distance measure - if $\|\mathbf{R} - \mathbf{T}\|_k$ is less than a globally set threshold then $\mathbf{T}$ will be accepted as being *sufficiently* close to $\mathbf{R}$. Code for the implementation of the ℓ_1 verification measure is presented in listing F.3.

Listing F.3: ℓ_1 Verification Code

```
function ComparisonResult = Verify(T,RefProfile,Threshold)

    %% R is the class reference vector (the 'R' field of RefProfile), it should be
    %% free of numberofpulses and extraneous last column data.
    %% T, on the other hand should be a test vector,
    %% taken directly from file, and will contain pulse number and last column data.
    %%
    %% Note that the test vector, T, is taken directly from the Impostor Matrix -
    %% it may therefore contain a large number of trailing zeros - these MUST be
    %% removed, before a vector comparison is performed.
    %%
    %% NH 12/8/'00

    TestFeatureVector = [];     %Extracted feature components from T.
    SameNumberofPulses = 0;     %Do both the Test and Reference have the same number of pulses?
    accept = 0;                 %has the test vector passed the comparison?
    ComparisonResult = zeros(1,2);

    [n,m] = size(RefProfile.R); %Get the number of components in M

    NumberofPulsesinReference = (m+1)/3;

    if (NumberofPulsesinReference == T(1) ) %Both Test and Reference vectors have the same
                                            %   number of pulses - so make the comparison...
        SameNumberofPulses = 1;             %Flag returned as first component of Result -
                                            %   Vectors have same number of pulses, therefore
                                            %   comparison will be/has been made.

        for i=2:(m+1)                       %Extract only Test Vector Features....
            TestFeatureVector = [TestFeatureVector, T(i)];
        end

        TestDistance = norm((RefProfile.R-TestFeatureVector),1);     %Calculate l1 norm
        accept = (TestDistance < Threshold);

    end %conditional branch

    ComparisonResult = [SameNumberofPulses,accept]; %[comparison performed??, outcome??]
```

The implementation of the fixed threshold ℓ_2 norm is identical in almost every respect to the ℓ_1 implementation, differing only in the distance calculation. For the ℓ_2 norm, this is executed with:

```
TestDistance = norm((RefProfile.R-TestFeatureVector),2);     %Calculate l2 norm .
```

F.2.2 ℓ_1 & ℓ_2 norm Functions (User Specific Threshold)

With a user specific acceptance threshold, a user's reference vector, **R** must firstly be created in the manner of listing F.2. The next stage is to compare all of a user's enrolment vectors to **R**, generating a distribution of enrolment distances;

$\|\mathbf{R} - \mathbf{E}_j\|_1$ for $j = 1, 2, ..., m$ where m is the number of enrolment vectors and $\mathbf{E}_j$ is the j^{th} enrolment vector. From the resulting distribution of distances, the mean distance and standard deviation are calculated for each participant. This process is covered in detail in section 5.3. Listing F.4 provides the code implementation for user specific enrolment under the ℓ_1 norm verification scheme.

Listing F.4: User Specific Enrolment

```
function ReferenceProfile = Enrol(EnrolmentVectors)

    %%%%%%%% Calculate Mean Enrolment Vector %%%%%%%%%

    R = mean(EnrolmentVectors);

    %Now Calculate the mean and std of enrolment sequences from the Mean reference vector
    Distances = [];
    for i=1:8
       SequenceVector = EnrolmentVectors(i,:);
       Distances(i,:) = norm((R-SequenceVector),1);
    end
    MeanEnrolmentNorm = mean(Distances);
    StdEnrolmentNorm = std(Distances);

   ReferenceProfile.R = R;
   ReferenceProfile.MeanEnrolmentNorm = MeanEnrolmentNorm;
   ReferenceProfile.StdEnrolmentNorm = StdEnrolmentNorm;
```

The mean enrolment distance and its standard deviation are stored in fields of the `ReferenceProfile` structure, and are required during verification.

The verification implementation of the user specific approach is similar to that of using a fixed acceptance threshold (shown in listing F.3). Rather than simply comparing the distance between reference and test vectors, this approach must firstly compute the distance $\|\mathbf{R} - \mathbf{T}\|_1$ and compare this distance to the mean enrolment distance. If the two are within a fixed proportion of the standard deviation of the enrolment distance, then $\mathbf{T}$ will be accepted. Listing F.5 provides the code implementation of the verification function given in equation 5.25.

Listing F.5: User Specific Verification

```
function ComparisonResult = Verify(T,RefProfile,Threshold)

    %% R is the class reference vector (the 'R' field of RefProfile), it should be
    %% free of numberofpulses and extraneous last column data.
    %% T, on the other hand should be a test vector,
    %% taken directly from file, and will contain pulse number and last column data.
```

```
%%
%% Note that the test vector, T, is taken directly from the Impostor Matrix -
%% it may therefore contain a large number of trailing zeros - these MUST be
%% removed, before a vector comparison is performed.
%%
%% NH 12/8/'00

TestFeatureVector = [];          %Extracted feature components from T.
SameNumberofPulses = 0;          %Do both the Test and Reference have the same number of pulses?
accept = 0;                      %has the test vector passed the comparison?
ComparisonResult = zeros(1,2);

[n,m] = size(RefProfile.R);      %Get the number of components in R

NumberofPulsesinReference = (m+1)/3;

if (NumberofPulsesinReference == T(1) ) %Both Test and Reference vectors have the same
                                        %   number of pulses - so make the comparison...
    SameNumberofPulses = 1;             %Flag returned as first component of Result - Vectors
                                        %   have same number of pulses, therefore comparison
                                        %   will be/has been made.

    for i=2:(m+1)               %Extract only Test Vector Features....
        TestFeatureVector = [TestFeatureVector, T(i)];
    end

    TestDistance = norm((RefProfile.R-TestFeatureVector),1);     %Calculate l1 norm

    accept = (sqrt((RefProfile.MeanEnrolmentNorm - TestDistance)^2))
                                                < (RefProfile.StdEnrolmentNorm*Threshold);

end %Comparison branch.

ComparisonResult = [SameNumberofPulses,accept];     %[comparison performed??, outcome??]
```

In common with the fixed threshold approach of section F.2.2, the ℓ_2 method differs from that of the ℓ_1 only in the distance calculation.

F.2.3 Mahalanobis Distance (Fixed Threshold)

The Mahalanobis distance verifier requires calculation of component variances of the enrolment vectors. As indicated in section 5.7.3 there are two obvious methods of performing this; either compute the covariance matrix and extract the leading diagonal using the enrolment vectors, or simply calculate the variances from each component explicitly. It was found that using the approach was more efficient in terms of the required number of floating point operations. This method will hence be presented here. Listing F.6 provides the code implementation for the fixed threshold version of this verifier.

Listing F.6: Fixed Threshold, Mahalanobis Distance - Enrolment

```
function ReferenceProfile = Enrol(EnrolmentVectors)

    %%%%% Calculate Mean Reference Vector & inverse variance matrix.......

   R = mean(EnrolmentVectors);
   V = var(EnrolmentVectors);
   D = diag(V);
   Vinv = inv(D);

   ReferenceProfile.R = R;
   ReferenceProfile.Vinv = Vinv;
```

Calculation of the mean reference vector and the inverse of the variance matrix is straightforward, making use of MatLab's higher matrix manipulation functions. For the calculation of the user specific Mahalanobis verifier the mean distance and standard deviation of the enrolment vectors, from the reference, is required. This is performed with the code of listing F.7.

Listing F.7: User Specific Threshold, Mahalanobis Distance - Enrolment

```
function ReferenceProfile = Enrol(EnrolmentVectors)

    %% Calculate Mean Reference Vector, inverse variance matrix.......
    %% ...and the mean (+ Standard Deviation) of the enrolment distance.

    R = mean(EnrolmentVectors);
    V = var(EnrolmentVectors);
    D = diag(V);
    Vinv = inv(D);

    %% Now Calculate the mean and std of enrolment vectors from the reference
    Distances = [];
    for i=1:8
       SequenceVector = EnrolmentVectors(i,:);
       Distances(i,:) = MahalanobisDistance(SequenceVector,R,Dinv);
    end
    MeanEnrolmentMD = mean(Distances);
    StdEnrolmentMD = std(Distances);

   ReferenceProfile.R = R;
   ReferenceProfile.Vinv = Vinv;
   ReferenceProfile.MeanEnrolmentMD = MeanEnrolmentMD;
   ReferenceProfile.StdEnrolmentMD = StdEnrolmentMD;
```

For robustness and convenience the distance function is performed in a separate function, as show in listing F.8.

Listing F.8: Mahalanobis Distance Function

```
function MD = MahalanobisDistance(T,M,Cinv)

    MD = (T-M)'*Cinv*(T-M);
```

Verification for the fixed threshold and user specific threshold approaches is performed with only minor modifications to the verification functions of the ℓ_1 norm, given in listings F.3 and F.5, respectively. The ℓ_1 distance measure in both code functions is simply replaced by a call to the Mahalanobis function of listing F.8.

F.2.4 Component Wise Linear

As indicated in section 5.7.4, the component-wise linear verifier compares each component of a test vector explicitly to the corresponding component of a reference vector. If the distance between the two is *sufficiently* small then the test vector will be accepted.

Enrolment for this method is the generation of a reference vector, whose components are the mean of the enrolment vector components. This is as stated in equation F.1 and the enrolment implementation is as given in listing F.2.

Verification, on the other-hand is more involved, requiring the explicit comparison of each component. Listing F.9 provides the implementation code.

Listing F.9: Component-Wise Linear - Verification

```
function ComparisonResult = Verify(T,RefProfile,Threshold)

    %% R is the class reference vector (the 'R' field of RefProfile), it should be
    %% free of numberofpulses and extraneous last column data.
    %% T, on the other hand should be a test vector,
    %% taken directly from file, and will contain pulse number and last column data.
    %%
    %% Note that the test vector, T, is taken directly from the Impostor Matrix -
    %% it may therefore contain a large number of trailing zeros - these MUST be
    %% removed, before a vector comparison is performed.
    %%
    %% NH 15/8/'00


    TestFeatureVector = [];    %Extracted feature components from T.
    SameNumberofPulses = 0;    %Do both the Test and Reference have the same number of pulses?
    acceptVector = 1;          %Assume Vector has passed until proved otherwise
    acceptComponent = 0;       %individual components will be compared.

    ComparisonResult = zeros(1,2);
```

```
    [n,m] = size(RefProfile.R);              %Get the number of components in R

    NumberofPulsesinReference = (m+1)/3;

    if (NumberofPulsesinReference == T(1) )  %Both Test and Reference vectors have the same
                                             %   number of pulses - so make the comparison...
        SameNumberofPulses = 1;              %Flag returned as first component of Result - Vectors
                                             %   have same number of pulses, therefore comparison
                                             %   will be/has been made.

        for i=2:(m+1)      %Extract only Test Vector Features....
            TestFeatureVector = [TestFeatureVector, T(i)];
        end

        for c = 1:(3*NumberofPulsesinReference-1)
            acceptComponent = ((((RefProfile.R(c) - TestFeatureVector(c))^2)^0.5)
                                                    <= (2*Threshold*RefProfile.R(c)/100));
            acceptVector = acceptVector & acceptComponent;
        end
    end


    ComparisonResult = [SameNumberofPulses,acceptVector];      %[comparison performed??, outcome??]
```

F.2.5 Component-Wise Non-Linear

The component-wise non-linear verifier is presented in detail in section 5.7.5. In summary this verification scheme enrols a user by generating both a reference vector comprising of the mean component values from that user's enrolment vectors, and a vector comprising of the standard deviation of each of the enrolment vector components.

Enrolment code makes use of MatLab's *mean* and *std* functions for the component-wise calculation of mean and standard deviation, and is shown in listing F.10.

Listing F.10: Component-Wise Non-Linear - Enrolment

```
function ReferenceProfile = Enrol(EnrolmentVectors)

    %% Calculate Mean...

    R = mean(EnrolmentVectors);

    %% and standard deviation...

    StdEnrolment = std(EnrolmentVectors);

  ReferenceProfile.R = R;
  ReferenceProfile.StdEnrolment = StdEnrolment;
```

Verification of a test vector under the non-linear component-wise scheme is very

similar to that of the linear component-wise version. They differ in the acceptance threshold; for the linear approach each test component must be within a certain proportion of the corresponding reference component, whilst the non-linear approach requires that each component be within the reference component $\pm$ a proportion of the user's standard deviation for that component (see equation 5.35).

Listing F.11 provides the coding detail.

Listing F.11: Component-Wise Non-Linear - Verification

```
function ComparisonResult = Verify(T,RefProfile,Threshold)

    %% R is the class reference vector (the 'R' field of RefProfile), it should be
    %% free of numberofpulses and extraneous last column data.
    %% T, on the other hand should be a test vector,
    %% taken directly from file, and will contain pulse number and last column data.
    %%
    %% Note that the test vector, T, is taken directly from the Impostor Matrix -
    %% it may therefore contain a large number of trailing zeros - these MUST be
    %% removed, before a vector comparison is performed.
    %%
    %% NH 15/8/'00

    TestFeatureVector = [];         %Extracted feature components from T.
    SameNumberofPulses = 0;         %Do both the Test and Reference have the same number of pulses?
    acceptVector = 1;               %Assume Vector has passed until proved otherwise
    acceptComponent = 0;            %individual components will be compared.

    ComparisonResult = zeros(1,2);

    [n,m] = size(RefProfile.R);             %Get the number of components in R

    NumberofPulsesinReference = (m+1)/3;

    if (NumberofPulsesinReference == T(1) ) %Both Test and Reference vectors have the same
                                            %   number of pulses - so make the comparison...
        SameNumberofPulses = 1;             %Flag returned as first component of Result - Vectors
                                            %   have same number of pulses, therefore comparison
                                            %   will be/has been made.

        for i=2:(m+1)               %Extract only Test Vector Features....
            TestFeatureVector = [TestFeatureVector, T(i)];
        end

        for c = 1:(3*NumberofPulsesinReference-1)
            acceptComponent = ((((RefProfile.R(c) - TestFeatureVector(c))^2)^0.5)
                                            <= (Threshold*RefProfile.StdEnrolment(c)));
            acceptVector = acceptVector & acceptComponent;
        end
    end

ComparisonResult = [SameNumberofPulses,acceptVector];   %[comparison performed??, outcome??]
```

F.2.6 Linear and Non-Linear Component-Wise Verifiers with Proportional Acceptance

This method is outlined in sections 5.7.4 and 5.7.5, and is a variation upon the component-wise approaches above.

Essentially the addendum of *proportional acceptance* is a relaxation of the condition *for all components*, accepting a test vector if a certain proportion of its components satisfy linear, or non-linear test conditions. Equations 5.33 and 5.36 describe the acceptance conditions for linear and non-linear verification functions, respectively. Whilst listing F.12 provides the code implementation for the linear approach.

Listing F.12: Component-Wise Linear with Proportional Acceptance

```
function ComparisonResult = Verify(T,RefProfile,Threshold, ComponentProportion)

    TestFeatureVector = [];   %Extracted feature components from T.
    SameNumberofPulses = 0;   %Do both the Test and Reference have the same number of pulses?
    acceptVector = 0;         %Vector passed??
    acceptComponent = 0;      %individual components will be compared.

    ComponentsAccepted = 0;   %Running Tally of the number of components satisfying test
    ComparisonResult = zeros(1,2);

    [n,m] = size(RefProfile.R);              %Get the number of components in R

    NumberofPulsesinReference = (m+1)/3;

    if (NumberofPulsesinReference == T(1) )  %Both Test and Reference vectors have the same
                                             %   number of pulses - so make the comparison...
        SameNumberofPulses = 1;              %Flag returned as first component of Result - Vectors
                                             %   have same number of pulses, therefore comparison
                                             %   will be/has been made.

        for i=2:(m+1)      %Extract only Test Vector Features....
            TestFeatureVector = [TestFeatureVector, T(i)];
        end

        for c = 1:(3*NumberofPulsesinReference-1)
            acceptComponent = ((((RefProfile.R(c) - TestFeatureVector(c))^2)^0.5)
                                                    <= (2*Threshold*RefProfile.R(c)/100));
            if (acceptComponent)
                ComponentsAccepted = ComponentsAccepted +1;
            end
        end
    end

    acceptVector = ((ComponentsAccepted/NumberofPulsesinReference)*100 >= ComponentProportion )
    ComparisonResult = [SameNumberofPulses,acceptVector];      %[comparison performed??, outcome??]
```

The non-linear component-wise code implementation differs from F.12 only in the

component test, which is contained within listing F.11.

Appendix G

Java Card Applets

This appendix presents the Java code applets and functions described in Chapter 6. Sections G.1 & G.2 provide the structure of the main applets running on the host PC and iButton, respectively, whilst Section G.3 presents the functional detail of the enrolment and verification mechanisms, considered in Chapter 6.

G.1 Host Applet

The host PC applet initiated the execution of enrolment and verification on the iButton. Its primary tasks are to

- Transmit enrolment vectors from PC to iButton
- Initiate enrolment code
- Initiate verification code
- Request and receive results from iButton
- Decipher and Display Results on PC.

Its main code loop executes when an iButton is inserted to its reader (`public void iButtonInserted(JibMultiEvent event)`), where upon a communication link is established, and the `IdentityVerification` applet is selected. If this is successful, then enrolment vectors are transmitted, and the iButton is requested to perform enrolment, then transmit the results and the time taken back to PC.

Results are displayed on screen and checked for accuracy. Verification is then performed in a similar manner.

In order to communicate with the iButton, the applet running on the host requires the OpenCard communication protocols. These are provided in the `com.ibutton.oc` and `opencard.core.terminal` packages, and are included in the applet header.

Listing G.1: PC Host Infrastructure Code

```
import com.ibutton.oc.*;
import com.ibutton.oc.CommandAPDU;
import opencard.core.terminal.*;
import com.ibutton.opt.jibletselect.Selector;
import java.io.*;

public class Identity_Verification_Host implements JibMultiListener
{
    // BEGIN INSTRUCTION DECLARATIONS
    public static final byte IDENTITY_VERIFICATION_CLA = (byte)0x80;
    public static final byte IDENTITY_VERIFICATION_INS_TRANSFER = (byte)0;
    public static final byte IDENTITY_VERIFICATION_INS_PERFORM_ENROLMENT = (byte)1;
    public static final byte IDENTITY_VERIFICATION_INS_SETDIMENSIONS = (byte)2;
    //   END INSTRUCTION DECLARATIONS

    //The object that will be used to communicate with the iButtons.
    protected JibMultiFactory factory;

    private String appletName;      //The name (AID) of the iButton applet used by this host.
    private String appletPath;

            // Verifier Specific Constants/Globals...
    int numberofPulses = 7;     //8;//14;
    int vectorDimension = (3*numberofPulses)-1;
    int vectorBytesRequired = 4*vectorDimension;
    int numberofEnrolmentVectors = 8;
    int[] featureVector = new int[vectorDimension];

    int enrolmentVectors[][] =          // Enrolment Feature Vectors Here.
        {
            {},
            {},
            {},
            {},
            {},
            {},
            {},
            {}
        };


    public Identity_Verification_Host(String appletPath, String appletName)
    {
        //Sets up the listener for iButton inserted and removed events.

        this.appletName = appletName;
        this.appletPath = appletPath;

        factory = new JibMultiFactory();
```

```
    factory.addJiBListener(this);       //Add the listener that will be notified
                                        //when an iButton is inserted or removed.

    try                                 //Begin polling for iButtons.
    {
        factory.startPolling(true);
    }
    catch(CardTerminalException cte)
    {
        cte.printStackTrace();
    }
}

public void addJiBListener(JibMultiListener l)
{
    //Adds the specified iButton listener to receive inserted and removed events.
    factory.addJiBListener(l);
}

public void removeJiBListener(JibMultiListener l)
{
    //Removes the specified iButton listener
    factory.removeJiBListener(l);
}

/**
 *
 *
 * @param event  the insertion event.
 */
public void iButtonInserted(JibMultiEvent event)
{
    //Called when an iButton inserted.
    //The master pin for this iButton.
    //If a password is set on the iButton you must
    //provide that here.
    String password = "";

    factory.setSlot(event.getSlotID());

    try                         //Try to select the applet.
    {
        Selector appletSelector = new Selector(false);
        if(appletSelector.select(appletPath,appletName + ".jib",appletName,
                                        event.getChannel(),password))

        {
            //****************************************
            //* Insert any code to be done when      *
            //* an iButton is inserted here.         *
            //****************************************

            System.out.println("iButton Inserted...");
            setdimensionsDispatch();            //Inform iButton about Vector Properties

            String fVector = "";
            for (int i = 0; i<=(numberofEnrolmentVectors-1);i++) {   //Prepare Vectors...

                featureVector = enrolmentVectors[i];
                fVector = "";
                for (int j = 0; j<=(vectorDimension-1);j++) {
                    fVector = fVector + enrolmentVectors[i][j] + " ";
                }
                System.out.println(fVector);
                transferDispatch();                         //Call Transmission function
```

```
            }
            Perform_EnrolmentDispatch();                    //Request
            System.out.println("Finished.");
        }
        else
        {
            int sw = appletSelector.getLastStatusWord();
            System.out.println("Applet load failed: "+Integer.toHexString(sw));
            System.out.println("Reason : "+decodeSW(sw));
        }

    }
    catch(CardTerminalException cte)
    {
        System.err.println("ERROR:  CardTerminalException occurred while communicating
                                                                    with iButton.");
    }
    catch(FileNotFoundException fnfe)
    {
        System.err.println("Unable to find file: " + appletName + ".jib");
    }
    catch(IOException ioe)
    {
        System.err.println("Unable to read file: " + appletName + ".jib");
    }
    catch(Exception e)
    {
        //Exceptions that occur in iButtonInserted events
        //will be drained in the OpenCard internals if we
        //don't catch them here.
        System.err.println("Exception in iButtonInserted:");
        e.printStackTrace();
    }
}

/**
 * Called when an iButton is removed.
 *
 * @param event  the removal event.
 */
public void iButtonRemoved(JibMultiEvent event)
{
    try
    {
        //*****************************************
        //* Insert any code to be done when       *
        //* an iButton is removed here.           *
        //*****************************************
    }
    catch(Exception e)
    {
        //Exceptions that occur in iButtonRemoved events
        //will be drained in the OpenCard internals if we
        //don't catch them here.
        System.err.println("Exception in iButtonRemoved:");
        e.printStackTrace();
    }
}


private String decodeSW(int sw)             //Decode Return Status Word
{
    switch (sw)
    {
        case 0x6100:
```

```
        return "Response Bytes Remaining";
    case 0x6301:
        return "Success Packet";
    case 0x6400:
        return "Insufficient Memory";
    case 0x6681:
        return "Bad Master PIN";
    case 0x6700:
        return "Wrong Length";
    case 0x6901:
        return "Invalid AID Length";
    case 0x6902:
        return  "\r\nInvalid API Version\r\n"+
                "-------------------\r\n"+
                "This error occurred because the applet (.jib file) was not built\r\n"+
                "for this kind of Java Powered iButton.\r\n\r\n";
    case 0x6903:
        return "Invalid Password";
    case 0x6904:
        return "Invalid Signature Length";
    case 0x6905:
        return "Hash Corruption";
    case 0x6906:
        return "Hash Failure";
    case 0x6982:
        return "Invalid Signature";
    case 0x6984:
        return "Data Invalid";
    case 0x6985:
        return "Conditions Of Use Not Satisfied";
    case 0x6A80:
        return "Wrong Data";
    case 0x6A81:
        return "Function Not Supported";
    case 0x6A82:
        return "Unable to Select Applet";
    case 0x6A84:
        return "Class Length Overrun";
    case 0x6A86:
        return "Invalid Loader Command";
    case 0x6A87:
        return "Incomplete Packet";
    case 0x6B00:
        return "Incorrect Parameters (P1,P2)";
    case 0x6C00:
        return "Correct Expected Length";
    case 0x6D00:
        return "INS Value Not Supported";
    case 0x6F00:
        StringBuffer sb = new StringBuffer("Uncaught Exception: ");
        try
        {
            ResponseAPDU res = factory.getLastError();
            int err = 0x0ff & res.data()[0];
            sb.append("Last Error reported as 0x");
            if (err<16) sb.append('0');
            sb.append(Integer.toHexString(err));
        }
        catch(Exception e)
        {
            sb.append("Could not getLastError!");
        }
        return sb.toString();
    case 0x8450:
        return "Unable to Find Applet";
```

```
        case 0x8453:
            return "Unable to Select Applet";
        case 0x9000:
            return "Success";
        default:
            StringBuffer unrec = new StringBuffer("Unrecognized SW ");
            if (sw < 0x1000) unrec.append('0');
            if (sw < 0x100)  unrec.append('0');
            if (sw < 0x10)   unrec.append('0');
            unrec.append(Integer.toHexString(sw));
            return unrec.toString();
    }
}

    // BEGIN INSTRUCTION DISPATCHER FUNCTIONS
public void setdimensionsDispatch() throws CardTerminalException
{
    // Send enrolment vector properties to iButton

    ResponseAPDU response;
    int[] sequenceParameters = {vectorDimension,numberofEnrolmentVectors};
    byte[] vectorDByteRepresentation = new byte[8];

    System.out.println("...Sending Sequence Parameters...");
    intArrayToByteArray(vectorDByteRepresentation,0,sequenceParameters);

    response = factory.sendAPDU(new CommandAPDU((byte)IDENTITY_VERIFICATION_CLA,
                                (byte)IDENTITY_VERIFICATION_INS_SETDIMENSIONS,
                            (byte)0x0, (byte)0x0, vectorDByteRepresentation, (byte)0));

    String responseStr = decodeSW(response.sw());
    System.out.println(responseStr);
}


public void setdimensionsDispatch(int slotNumber) throws CardTerminalException
{
    factory.setSlot(slotNumber);
    setdimensionsDispatch();
}


public void transferDispatch() throws CardTerminalException
{
    //Sends Enrolment Vectors to the iButton

    ResponseAPDU response;
    byte[] data = new byte[4*vectorDimension];

    System.out.println("....Sending Enrolment Vector....");

    intArrayToByteArray(data, 0, featureVector); //Convert integer to byte array

            response = factory.sendAPDU(new CommandAPDU((byte)IDENTITY_VERIFICATION_CLA,
                                        (byte)IDENTITY_VERIFICATION_INS_TRANSFER,
                                        (byte)0x0, (byte)0x0, data, (byte)0));
    String responseStr = decodeSW(response.sw());
    System.out.println(responseStr);
}

public void transferDispatch(int slotNumber) throws CardTerminalException
{
    factory.setSlot(slotNumber);
    transferDispatch();
}
```

```
public void Perform_EnrolmentDispatch() throws CardTerminalException
{
    //Launches a request to initiate Enrolment procedure

    ResponseAPDU response;
    int[] dataResponse = new int[vectorDimension+3];
    String returnedData = "";

    System.out.println("...Requesting Enrolment....");
    response = factory.sendAPDU(new CommandAPDU((byte)IDENTITY_VERIFICATION_CLA,
                                (byte)IDENTITY_VERIFICATION_INS_PERFORM_ENROLMENT,
                        (byte)0x0, (byte)0x0, null /*add data to send here*/, (byte)0));

    String responseStr = decodeSW(response.sw());
    System.out.println(responseStr);

    System.out.println("Response Recieved...Processing...");
    dataResponse = intArrayFromByteArray(response.data(),0,(vectorBytesRequired+12));

    for (int i= 0;i<=(vectorDimension+2);i++)   //We have vectorDimension +1 elements
                                                //                (time,d0,d1,...,dn-1)
    {
        returnedData = returnedData+ dataResponse[i] +" ";
    }

    System.out.println(returnedData);
}

public void Perform_EnrolmentDispatch(int slotNumber) throws CardTerminalException
{
    factory.setSlot(slotNumber);
    Perform_EnrolmentDispatch();
}


    //   END INSTRUCTION DISPATCHER FUNCTIONS

    // BEGIN CONVENIENCE FUNCTIONS
public static void intToByteArray(byte[] outArray, int start, int value)
{
    //BigEndian
    outArray[start]     = (byte)((value & 0xFF000000) >>> 24);
    outArray[start + 1] = (byte)((value & 0x00FF0000) >>> 16);
    outArray[start + 2] = (byte)((value & 0x0000FF00) >>>  8);
    outArray[start + 3] = (byte) (value & 0x000000FF);
}

public static int intFromByteArray(byte[] inArray, int start)
{
    //BigEndian
    return (((int)inArray[start    ] << 24) & 0xFF000000 |
            ((int)inArray[start + 1] << 16) & 0x00FF0000 |
            ((int)inArray[start + 2] <<  8) & 0x0000FF00 |
            ((int)inArray[start + 3]      ) & 0x000000FF);
}

public static void intArrayToByteArray(byte[] outArray, int start, int[] value)
{
    //BigEndian
    for(int i = 0; i < value.length; i++)
    {
        outArray[start + (i * 4)]     = (byte)((value[i] & 0xFF000000) >>> 24);
```

```
            outArray[start + (i * 4) + 1] = (byte)((value[i] & 0x00FF0000) >>> 16);
            outArray[start + (i * 4) + 2] = (byte)((value[i] & 0x0000FF00) >>>  8);
            outArray[start + (i * 4) + 3] = (byte) (value[i] & 0x000000FF);
        }
    }

    public static int[] intArrayFromByteArray(byte[] inArray, int start, int length)
    {
        int[] ia = new int[length / 4];
        //BigEndian
        for(int i = 0; i < length / 4; i++)
        {
            ia[i] = (((int)inArray[start + (i * 4)    ] << 24) & 0xFF000000 |
                     ((int)inArray[start + (i * 4) + 1] << 16) & 0x00FF0000 |
                     ((int)inArray[start + (i * 4) + 2] <<  8) & 0x0000FF00 |
                     ((int)inArray[start + (i * 4) + 3]      ) & 0x000000FF);
        }
        return ia;
    }
        //   END CONVENIENCE FUNCTIONS

    public static void main(String[] args)
    {
        //The following path has been auto-generated by
        //the Project Wizard.  If the project is moved
        //to a different directory, you MUST update this
        //path accordingly.
        new Identity_Verification_Host("C:\\ ... \\ 'Identity Verification Class Path'.....");
    }
}
```

G.2 iButton Infrastructure Applet

The iButton applet essentially listens for, and responds to, requests from the host PC. It firstly, receives enrolment vectors and packages them in an array. Then, upon request, calls the enrolment and verification processes, and transmits the results back to the host. Timing is performed using the iButton's internal real-time clock, hence the package `com.dalsemi.system.Clock` is included in the preamble. The clock has a minimum resolution of one second, and hence, enrolment and verification are each performed 100 times, and average times are calculated back on the host.

Listing G.2: iButton Infrastructure Code

```
import javacard.framework.*;
import com.dalsemi.system.Clock;
import java.lang.Math;

public class Identity_Verification_Applet extends Applet {

    // BEGIN INSTRUCTION DECLARATIONS
    public static final byte IDENTITY_VERIFICATION_CLA = (byte)0x80;
    public static final byte IDENTITY_VERIFICATION_INS_TRANSFER = (byte)0;
```

```
public static final byte IDENTITY_VERIFICATION_INS_PERFORM_ENROLMENT = (byte)1;
public static final byte IDENTITY_VERIFICATION_INS_SETDIMENSIONS = (byte)2;
public static final byte IDENTITY_VERIFICATION_INS_PERFORMVERIFICATION = (byte)3;
// END INSTRUCTION DECLARATIONS

final static private byte SELECT_CLA = (byte)0x00;          //Used to select an applet.
final static private byte SELECT_INS = (byte)0xA4;          //      "
final static private short MAX_SEND_LENGTH = (short)1000;   //Max bytes in a single apdu.

        // initialisation Code from here....

public byte[] apduData;
int[] incomingData;                 //Holds integer passed from host.
int[] enrolmentVectors;
int maxNumberofEnrolmentVectors;
int numberofEnrolmentVectors = 0;
int vectorDimension;

int[] featureVector;                //Single Feature vector from Host
byte[] outgoingData;                //Returned Reference Byte Vector
int[] returnedFVector;              //Returned Reference Data
int[] referenceVector;              //Reference Vector From Enrolment Function

    // Add Enrolment / Verification Specific Globals here...

    // End Specific Globals.


public Identity_Verification_Applet()
{
    register();                     //Register this applet with the JCRE
}

public static void install(APDU apdu)
{
    new Identity_Verification_Applet();
}

public void process(APDU apdu)
{
    byte[] buffer = apdu.getBuffer();

    if((buffer[ISO.OFFSET_CLA] == SELECT_CLA) &&
       (buffer[ISO.OFFSET_INS] == SELECT_INS))      //Determine if applet is being selected.
    {
        return;
    }
    apduData = new byte[buffer[ISO.OFFSET_LC] & 0x0FF];
    short apduDataOffset = 0;
    short bytesRead = apdu.setIncomingAndReceive(); //Read in the entire APDU.

    while (bytesRead > 0)           //Loop until all bytes have been read.
    {
        Util.arrayCopyNonAtomic(buffer, ISO.OFFSET_CDATA, apduData, apduDataOffset, bytesRead);
        apduDataOffset += bytesRead;
        bytesRead = apdu.receiveBytes(ISO.OFFSET_CDATA);
    }

    apdu.setOutgoing();                             //Prepare the apdu for sending.
    apdu.setOutgoingLength(MAX_SEND_LENGTH);

    if(buffer[ISO.OFFSET_CLA] != IDENTITY_VERIFICATION_CLA) //Check for a valid CLA.
    {
        ISOException.throwIt(ISO.SW_CLA_NOT_SUPPORTED);
    }
```

```
    else
    {
        switch (buffer[ISO.OFFSET_INS])     //Call the appropriate dispatch method for the INS.
        {
            case IDENTITY_VERIFICATION_INS_PERFORMVERIFICATION:
                performverificationDispatch(apdu,buffer[ISO.OFFSET_P1],buffer[ISO.OFFSET_P2]);
                break;
            case IDENTITY_VERIFICATION_INS_SETDIMENSIONS:
                setdimensionsDispatch(apdu, buffer[ISO.OFFSET_P1], buffer[ISO.OFFSET_P2]);
                break;
            case IDENTITY_VERIFICATION_INS_TRANSFER:
                transferDispatch(apdu, buffer[ISO.OFFSET_P1], buffer[ISO.OFFSET_P2]);
                break;
            case IDENTITY_VERIFICATION_INS_PERFORM_ENROLMENT:
                Perform_EnrolmentDispatch(apdu, buffer[ISO.OFFSET_P1], buffer[ISO.OFFSET_P2]);
                break;
            default:
                ISOException.throwIt(ISO.SW_INS_NOT_SUPPORTED);
        }
    }
}

protected void sendByteArray(APDU apdu, byte[] data)
{
    short offset = 0;
    while((data.length - offset) > MAX_SEND_LENGTH)
    {
        apdu.sendBytesLong(data, offset, MAX_SEND_LENGTH);
        offset += MAX_SEND_LENGTH;
    }
    apdu.sendBytesLong(data, offset, (short)(data.length - offset));
}


// BEGIN INSTRUCTION DISPATCHER FUNCTIONS

public void performverificationDispatch(APDU apdu, byte p1, byte p2)
{
    //Performs and Times Comparison between reference Vector and test Vector.
    int vectorBytesRequired = 4*(2*vectorDimension+2);
    int startTime;
    int finishTime;

    outgoingData = new byte[vectorBytesRequired];   //Returned Reference Byte Vector
    returnedFVector = new int[vectorDimension+2];   //Returned Reference + Enrolment Time

    int Threshold = 5;      //Acceptance Threshold
    boolean Accept = false;

    for (int i= 0;i<=vectorDimension-1;i++) //copy first enrolment vector for verification
    {
        featureVector[i] = enrolmentVectors[i];
    }

    startTime = Clock.getClock();
    for(int i=0; i<100;i++) {
        Accept = VerifyUser(referenceVector, featureVector, ...., );
    }
    finishTime = Clock.getClock();


    returnedFVector[0] = (finishTime-startTime);    //Send Enrolment Data Back to Host...
    if (Accept == true) {
        returnedFVector[1] = 1;
```

```
    }
    else {
        returnedFVector[1] = 0;
    }
    intArrayToByteArray(outgoingData, 0, returnedFVector);
    sendByteArray(apdu, outgoingData);
}


public void setdimensionsDispatch(APDU apdu, byte p1, byte p2)
{
    //Recieves Initialisation data from the Host.../

    int[] sequenceParameters = new int[2];

    sequenceParameters = intArrayFromByteArray(apduData,0,8);
    vectorDimension = sequenceParameters[0];
    maxNumberofEnrolmentVectors = sequenceParameters[1];

    enrolmentVectors = new int[vectorDimension*maxNumberofEnrolmentVectors];

    featureVector = new int[vectorDimension];       //Single Feature vector from Host
    referenceVector = new int[vectorDimension];     //Reference Vector From Enrolment
    numberofEnrolmentVectors = 0;

        // Add Enrolment / Verification Specific Globals here...

        // End Globals.
}


public void transferDispatch(APDU apdu, byte p1, byte p2)
{
    // Retrieves Enrolment Vectors from Host.../

    int vectorBytesRequired = 4*vectorDimension;

    featureVector = intArrayFromByteArray(apduData,0,vectorBytesRequired);
    numberofEnrolmentVectors +=1;

        // Have Enrolment Vector, Now place in appropriate location of ...
        //      enrolmentVectors[] array.

    int enrolmentIndexStart = ((numberofEnrolmentVectors-1)*vectorDimension);

    for (int i = 0; i<=(vectorDimension-1);i++)
    {
        enrolmentVectors[enrolmentIndexStart+i] = featureVector[i];
    }
 }


public void PerformEnrolmentDispatch(APDU apdu, byte p1, byte p2)
{
    // Calls and times enrolment process.../
    int vectorBytesRequired = 4*(vectorDimension+3);
    int startTime;
    int finishTime;

    outgoingData = new byte[vectorBytesRequired];   //Reference Byte Vector
    returnedFVector = new int[vectorDimension+3];   //Reference vector + Enrol Time


    startTime = Clock.getClock();
    for(int i=0; i<10;i++) {
```

```
            EnrolUser(referenceVector);
        }
        finishTime = Clock.getClock();

        //Send Enrolment Data Back to Host...
        returnedFVector[0] = (finishTime-startTime);
        returnedFVector[1] = distanceMean;
        returnedFVector[2] = distanceStdDev;

        for(int i = 0; i<=(vectorDimension-1);i++) {
            returnedFVector[i+3] = referenceVector[i];
        }


        intArrayToByteArray(outgoingData, 0, returnedFVector);
        sendByteArray(apdu, outgoingData);

    }


    public void EnrolUser(int[] referenceVector)
    {
        // Specific Enrolment Function Here.


        //              ....
    }


    boolean VerifyUser(int[] refVector, int[] testVector, ...., )
    {
       // Specific Verification Function Here.

       //              ....
    }

    //   END INSTRUCTION DISPATCHER FUNCTIONS


    // BEGIN CONVENIENCE FUNCTIONS

           //......Convenience Functions are the same as Host Functions.

    // END CONVENIENCE FUNCTIONS
}
```

G.3 Enrolment and Verification Functions

G.3.1 ℓ_1 Norm, Per User Basis

As discussed in sections 5.7.1 and 6.3.1, the reference vector for the ℓ_1 verifier uses the component means of the enrolment vectors. On a user specific basis, the mean distance of the enrolment vectors, from the reference vector, is calculated and used

to match the verifier's performance to the variability of each user. The enrolment function below (Listing G.3) implements this approach in Java.

Listing G.3: ℓ_1 Norm Enrolment Function

```
public void EnrolUser(int[] referenceVector)
{
    int enrolmentVectorStartIndex;
    int enrolmentVectorIndex;
    int vectorDistance;

    int distanceSum=0;
    int distanceSumofSquares=0;
    int distanceVariance=0;


    for (int j = 0;j<=(vectorDimension-1);j++)    //Reset reference vector information...
    {
        referenceVector[j] = 0;
        componentSum[j] = 0;
        sumofSquares[j] = 0;
        featureVector[j] = 0;
    }

    for (int j = 0; j<=(maxNumberofEnrolmentVectors-1);j++)
    {
        L1Distance[j] = 0;
    }

            //Sum Each Dimension of EnrolmentVectors
    for(int ev = 0;ev<=(numberofEnrolmentVectors-1);ev++){
        enrolmentVectorStartIndex = (ev*vectorDimension);
        for(int d = 0;d<=(vectorDimension-1);d++){
            enrolmentVectorIndex = enrolmentVectorStartIndex + d;
            componentSum[d] +=enrolmentVectors[enrolmentVectorIndex];
        }
    }
            //Find The Mean EnrolmentVector
    for(int d = 0;d<=(vectorDimension-1);d++){
        referenceVector[d] = componentSum[d]/numberofEnrolmentVectors;
    }

            //For Each Enrolment Vector, Find the L1 Distance from mean...
    for(int ev = 0; ev<=(numberofEnrolmentVectors-1);ev++) {

        enrolmentVectorStartIndex = (ev*vectorDimension);
        for(int d = 0;d<=(vectorDimension-1);d++){  //Extract Enrol. vector from main array
            enrolmentVectorIndex = enrolmentVectorStartIndex + d;
            featureVector[d] =enrolmentVectors[enrolmentVectorIndex];
        }

        vectorDistance = L1Norm(featureVector, referenceVector);
        L1Distance[ev] = vectorDistance;
    }

            //Distances Calculated - now calculate Mean and Std. Deviation of Distances...
    for (int ed = 0;ed<=(numberofEnrolmentVectors-1);ed++) {

        distanceSum += L1Distance[ed];
        distanceSumofSquares += L1Distance[ed] * L1Distance[ed];
    }
```

```
    distanceMean = distanceSum/numberofEnrolmentVectors;
    distanceVariance = (distanceSumofSquares - (distanceSum*distanceSum) /
                              (numberofEnrolmentVectors))/(numberofEnrolmentVectors-1);
    distanceStdDev = SQRT(distanceVariance);

}
```

Verification on a per user basis, requires that a test vector is compared to a user's reference vector. If the two are within a predefined proportion of the user's mean enrolment to reference distance, then the test vector will be accepted. Listing G.4 implements this process.

Listing G.4: ℓ_1 Norm Verification Function

```
boolean VerifyUser(int[] refVector, int[] testVector, int meanDistance, int enrolmentStdDev,
                                                                              int Threshold)
{
    //Performs Verification of testVector Against Enrolment Vector...
   boolean Accept = false;
   int distance;

   distance = L1Norm(testVector, refVector);
   if (distance > meanDistance) {
        Accept = ((distance - meanDistance) <= (Threshold*enrolmentStdDev));
   }
   else {
        Accept = ((meanDistance - distance) <= (Threshold*enrolmentStdDev));
   }

   return Accept;
}
```

Listing G.5: ℓ_1 Norm Distance Implementation

```
public int L1Norm(int[] fVector, int[] refVector)
{
    int distance=0;

    for (int d = 0; d<=(vectorDimension-1);d++) {
        if (fVector[d]>=refVector[d]) {
            distance += (fVector[d]-refVector[d]);
        }
        else {
            distance += (refVector[d]-fVector[d]);
        }
    }
    return distance;
}
```

Listing G.6: Square Root Implementation

```
public static int SQRT(int a)
{
    int square = 1;
    int delta = 3;

    while(square<=a) {
        square+=delta;
        delta +=2;
    }
    return (delta/2 - 1);
}
```

G.3.2 ℓ_2 Norm, Per User Basis

The user specific ℓ_2 norm verifier takes a similar structure to that of the user specific ℓ_1 norm verifier, presented above. The only difference between the two is the ℓ_2 distance function, given by Listing G.7, below.

Listing G.7: ℓ_2 Norm Distance Implementation

```
public int L2Norm(int[] fVector, int[] refVector)
{
    int distance=0;

    for (int d = 0; d<=(vectorDimension-1);d++) {
        if (fVector[d]>=refVector[d]) {
            distance += (fVector[d]-refVector[d])*(fVector[d]-refVector[d]);
        }
        else {
            distance += (refVector[d]-fVector[d])*(refVector[d]-fVector[d]);
        }
    }
    return SQRT(distance);
}
```

G.3.3 Mahalanobis Distance, Per User Basis

The user specific Mahalanobis distance verifier is a little different from those of the ℓ_1 and ℓ_2 norm verifiers. The main difference is that component variances of the enrolment vectors must be calculated during enrolment, and are used during verification as a substitute for the covariance matrix. This is as discussed in Section 6.3.1, and the core differences in enrolment are presented in Listing G.8.

Listing G.8: Mahalanobis Enrolment - Core Differences

```
Public void EnrolUser(int[] meanReferenceVector)
{
        // Declarations as l1 Norm enrolment function
```

```
        //
        // ....

        //Sum Each Dimension of EnrolmentVectors
    for(int ev = 0;ev<=(numberofEnrolmentVectors-1);ev++)
    {
        enrolmentVectorStartIndex = (ev*vectorDimension);
        for(int d = 0;d<=(vectorDimension-1);d++)
        {
            enrolmentVectorIndex = enrolmentVectorStartIndex + d;
            componentSum[d] +=enrolmentVectors[enrolmentVectorIndex];
            componentSumofSquares[d] += enrolmentVectors[enrolmentVectorIndex] *
                                        enrolmentVectors[enrolmentVectorIndex];
        }
    }
            //Find The Mean EnrolmentVector
    for(int d = 0;d<=(vectorDimension-1);d++)
    {
        meanReferenceVector[d] = componentSum[d]/numberofEnrolmentVectors;
        componentVariance[d] = (componentSumofSquares[d] - (componentSum[d]*componentSum[d]) /
                                (numberofEnrolmentVectors))/(numberofEnrolmentVectors-1);
    }

            //For Each Enrolment Vector, Find the Distance from Reference...
    for(int ev = 0; ev<=(numberofEnrolmentVectors-1);ev++)
    {
        enrolmentVectorStartIndex = (ev*vectorDimension);
        for(int d = 0;d<=(vectorDimension-1);d++)    //Extract Enrolment vector from main array
        {
            enrolmentVectorIndex = enrolmentVectorStartIndex + d;
            featureVector[d] =enrolmentVectors[enrolmentVectorIndex];
        }

        vectorDistance = Mahalanobis(featureVector, meanReferenceVector, componentVariance);
        MDistance[ev] = vectorDistance;
    }

            // Distances Calculated - now calculate Mean and Std. Deviation of Distances...
            //
            // .... As L1 Norm Enrolment
}
```

The remainder of the user specific Mahalanobis verifier is similar to those of the user specific ℓ_1 and ℓ_2 norm verifiers (Listing G.4), with the exception of the distance function, which is given in Listing G.9.

Listing G.9: Mahalanobis Distance Implementation

```
public int Mahalanobis(int[] fVector, int[] refVector, int[] cVariance)
{
    int distance=0;
    int componentDifference;

    for (int d = 0; d<=(vectorDimension-1);d++)
    {
        componentDifference = fVector[d] - refVector[d];
        distance += (100*componentDifference*componentDifference)/cVariance[d];
    }
    return (distance/100);
}
```

G.3.4 ℓ_1, ℓ_2, Mahalanobis, and Component-Wise Linear (Fixed Threshold) Verifiers

The ℓ_1, ℓ_2, and component-wise linear approaches to verification use only the mean of each component of the enrolment vectors, for the reference vector. As explained in Section G.3.3, the Mahalanobis distance function is a little different in that the variance of each enrolment component is required. This is performed in the same manner as given in Listing G.8, and is otherwise the similar. The Java code for these enrolment functions is presented in Listing G.10.

Listing G.10: Fixed Threshold Enrolment Function

```
public void EnrolUser(int[] meanReferenceVector)
{
    //For the L1Norm, L2Norm, Component-Wise Linear and MICD Verifiers...
    //      with FIXED Acceptance Thresholds

    int enrolmentVectorStartIndex;
    int enrolmentVectorIndex;
    int vectorDistance;

    int distanceSum=0;

            //Sum Each Dimension of EnrolmentVectors
    for(int ev = 0;ev<=(numberofEnrolmentVectors-1);ev++)
    {
        enrolmentVectorStartIndex = (ev*vectorDimension);
        for(int d = 0;d<=(vectorDimension-1);d++)
        {
            enrolmentVectorIndex = enrolmentVectorStartIndex + d;
            componentSum[d] +=enrolmentVectors[enrolmentVectorIndex];
        }
    }

            //Find The Mean EnrolmentVector - Serves As ReferenceVector
    for(int d = 0;d<=(vectorDimension-1);d++)
    {
        meanReferenceVector[d] = componentSum[d]/numberofEnrolmentVectors;
    }
}
```

The verification function for fixed threshold verifiers is a simple test of the distance between a test vector and the reference vector being $\leq$ a preset threshold distance. The generic Java code for this is presented in Listing G.11. The call to `DistanceFunction` indicates substitution of a relevant vector distance function, and can be selected from Listings G.5, G.7 and G.9 for the ℓ_1, ℓ_2 and Mahalanobis verifiers respectively. The component-wise linear verification function simply compares each component of a test vector with the appropriate component of the reference vector. If the difference is less than a predefined fraction of the reference component, then the test vector is accepted.

Listing G.11: Fixed Threshold Verification Function

```
boolean VerifyUser(int[] refVector,int[] testVector, int Threshold)
{
    //Performs Verification of testVector Against Enrolment Vector...
   boolean Accept = false;
   Accept = (DistanceFunction(testVector,refVector) <= Threshold);

   return Accept;
}
```

G.3.5 Component-Wise Non-Linear

The component-wise non-linear approach to verification (as discussed in Section 5.7.5, requires that the standard deviation of the enrolment components are calculated. Variance of the enrolment components is calculated in the manner set-out in Listing ??, and the standard deviation is found by using the square root function, given in Listing G.6.

Verification is performed on a per-component basis, whereby each component of a test vector is compared to the appropriate component of the reference vector. If the difference between the two is $\leq$ a predefined proportion of the standard deviation for this component, then the test vector is accepted. Listing G.12 shows the implementation.

Listing G.12: Component-Wise Non-Linear Verification Function

```
boolean VerifyUser(int[] refVector,int[] stdDev, int[] testVector, int Threshold)
{
   boolean Accept = true;
   int componentDistance;

   for (int i = 0; i<=vectorDimension-1;i++) {
        if (refVector[i] > testVector[i] ) {
            componentDistance = (refVector[i] - testVector[i]);
        }
        else {
            componentDistance = (testVector[i] - refVector[i]);
        }

        Accept = Accept & (componentDistance <= Threshold*stdDev[i]);
   }
   return Accept;
}
```

Appendix H

Publications

The following is a list of publications resulting from the work of this thesis.

Henderson, N. J. & Hartel, P. H. (2000). 'Pressure Sequence' – a novel method of protecting smart cards, *Smart Card Research and Advanced Applications*, IFIP/TC8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Publishers, Bristol, UK, pp. 241–256. ISBN: 0-7923-7953-5.

Henderson, N. J., Papakostas, T. V., White, N. M. & Hartel, P. H. (2001). Polymer thick film sensors: Possibilities for smartcard biometrics, *IOP Sensors and Their Applications XI*, Institute of Physics, IOP Publishing, London, pp. 83–88.

Henderson, N. J., White, N. M. & Hartel, P. H. (2001). iButton enrolment and verification requirements for the pressure sequence smartcard biometric, *in* I. Attali & T. Jensen (eds), *Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001*, Java Card Forum, Eurosmart, INRIA, Springer, Cannes, France, pp. 124–134. Lecture Notes in Computer Science Series, ISBN: 3-540-42610-8.

Henderson, N. J., Papakostas, T. V., White, N. M. & Hartel, P. H. (2002). Low-cost planar PTF sensors for the identity verification of smartcard holders, *IEEE Sensors 2002*, IEEE, Orlando, Florida, p. To Appear.

Henderson, N. J., White, N. M., Veldhuis, R., Slump, K. & Hartel, P. H. (2002a). A novel approach to the verification of smartcard holders, *IEEE Transactions on Systems, Man and Cybernetics*. To be submitted (summer 2002).

Henderson, N. J., White, N. M., Veldhuis, R., Slump, K. & Hartel, P. H. (2002b). Sensing pressure for authentication and feedback, *IEEE Benelux Signal Processing Symposium*, IEEE, Leuven, Belgium, p. To Appear.

Bibliography

Adams, J. (2000). Biometrics and smart cards, *Biometric Technology Today* **8**(4): 8–11.

Alexandre, T. J. (1997). Biometrics on smart cards: An approach to keyboard behavioural signature, *Future Generation Conputer Systems* **13**(1): 19–26.

Arshak, K. I., M^{c}Donagh, D. & Durcan, M. A. (2000). Development of new capacitive strain sensors based on thick film polymer and cermet technologies, *Sensors and Actuators A: Physical* **79**(2): 102–114.

Arshak, K. I., Ray, A. K., Hogarth, C. A., Collins, D. G. & Ansari, F. (1995). An analysis of polymeric thick-film resistors as pressure sensors, *Sensors and Actuators A: Physical* **49**(1-2): 41–45.

Ashbourn, J. (2000). *Biometrics: Advanced Identity Verification*, Springer, London. ISBN: 1-852-33243-3.

Aufreiter, R. (2001). Bridging the gap between biometrics and cryptography, *Card Technology Today* **2001**(11): 9.

Bajaj, R. & Chaudhury, S. (1997). Signature verification using multiple neural classifiers, *Pattern Recognition* **30**(1): 1–7.

Baltzakis, H. & Papamarkos, N. (2001). A new signature verification technique based on a two-stage neural network classifier, *Engineering Applications of Artificial Intelligence* **14**(1): 95–103.

Bao, Z., Feng, Y. & Dodabalapur, A. (1997). High-performance plastic transistors fabricated by printing techniques, *Chemistry of Materials* **9**(6): 1299–1301.

Bellin, R. W. (1989). System and method for identifying an individual utilizing grasping pressures, US Patent: US 4,857,916.

Bleha, S. A. & Obaidat, M. S. (1993). Computer user verification using the perceptron algorithm, *IEEE Transactions on Systems, Man and Cybernetics* **23**(3): 900–902.

Bleha, S., Slivinsky, C. & Hussien, B. (1990). Computer access security systems using keystroke dynamics, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(12): 1217–1222.

Bogue, R. (2002). Conference report – sensors and their applications XI, *Sensor Review* **22**(1): 34–40.

Bonetti, P., Ravaioli, S. & Piergallini, S. (2000). The Italian academic community's electronic voting system, *Computer Networks* **34**(6): 851–860.

Borst, J., Preneel, B. & Rijmen, V. (2001). Cryptography on smart cards, *Computer Networks* **36**(4): 423–435.

Bree, H., Leussink, P., Korthorst, T., Jansen, H., Lammerink, T. & Elwenspoek, M. (1996). The μ-flown: a novel device for measuring acoustic flows, *Sensors and Actuators A: Physical* **54**(1-3): 552–557.

Brown, M. & Rogers, S. J. (1993). User identification via keystroke characteristics of typed names using neural networks, *International Journal of Man-Machine Studies* **39**(6): 999–1014.

Bryan, W. L. & Harter, N. (1897). Studies in the physiology and psychology of the telegraphic language, *Psychological Review* **4**(1): 27–53.

Burge, M. & Burger, W. (1998). *Ear Biometrics.* In (Jain et al. 1998).

Burge, M. & Burger, W. (2000). Ear biometrics in computer vision, *in* A. Sanfeliu, J. Villanueva, M. Vanrell, R. Alquezar, A. Jain & J. Kittler (eds), *15th International Conference on Pattern Recognition, vol. 2, Proceedings - Pattern Recognition and Neural Networks*, IEEE Computer Soc., Los Alamitos, pp. 822–826.

Burroughs, J., Bradley, D. & Brown, A. (1990). Light-emitting diodes based on conjugated polymers, *Nature* **347**(6293): 539–541.

Callister, W. D. (1997). *Material Science and Engineering: An Introduction (Fifth Edition)*, John Wiley & Sons, New York. ISBN: 0-471-32013-7.

Campbell, J. P. (1997). Speaker recognition: A tutorial, *Proceedings of the IEEE* **85**(9): 1437–1462.

Campbell, J. P. (1998). *Speaker Recognition.* In (Jain et al. 1998).

Castellà-Roca, J., Domingo-Ferrer, J. & Herrera-Joancomart, J. (2000). A performance comparison of java cards for micropayment implementation, *Smart Card Research and Advanced Applications*, IFIP/TC8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Publishers, Bristol, UK, pp. 19–38.

Castrucci, P. (1972). Information card, US Patent: US 3,702,464.

Cattell, R., Carroll, N. & Saby, E. (2002). Three smart card sectors to watch in 2002, *Cart Technology Today* **13**(5): 11.

Chan, K. C. C. & Wong, A. C. K. (1990). APACS: A system for the automatic analysis and classification of conceptual patterns, *Computer Intelligence* **6**(1): 119–131.

Chen, Z. (2000). *JavaCard Technology for Smart Cards*, Addison-Wesley, Boston, USA. ISBN: 0-201-70329-7.

Choi, D. (2000). Asian telecom group makes large vein recognition order, *Biometrics Technology Today* **8**(3): 4.

Clancy, T. (1999). *Powerplays: Ruthless.com*, Penguin Books, London. ISBN: 0-140-27924-5.

Clarke, R. (1994). Human identification in information systems, *Information Technology and People* **17**(4): 6–37.

Clayden, D. O. (1998). Biometric identification of individuals by use of subcutaneous vein patterns, US Patent: US 5,787,185.

Coetzee, L. & Botha, E. (1996). A fingerprint sensor using frustrated total internal reflection, *Applied Optics* **26**(8): 1441–1460.

Crenshaw, J. (1998). Integer square roots, *Embedded Systems Programming* **11**(2): 28–36.

Császár, C. & Harsányi, G. (1994). A very low-cost pressure sensor with extremely high sensitivity, *Sensors and Actuators A: Physical* **42**(1-3): 417–420.

Cummins, H. (1964). *Dermatoglyphics: A Brief Review*, Academic Press Inc. in *The Epidermis* Edited by Montagna and Lobitz.

Daugman, J. (1993). High confidence visual recognition of persons by a test of statistical independence., *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(11): 1148–1161.

Daugman, J. (1994). Biometric personal identification system based on iris analysis, US Patent: US 5,291,560.

Daugman, J. (1998). *Recognising persons by their Iris Patterns.* In (Jain et al. 1998).

Davies, S. G. (1994). Touching big brother, *Information Technology & People* **7**(4): 38–47.

Dell, M. & Bunney, C. (2001). Biometrics and aviation security, *Biometric Technology Today* **2001**(11): 7–8.

Dethloff, J. (1978). Identification system safeguarded against misuse, US Patent: US 4,105,156.

Dethloff, J. & Grötrupp, H. (1968). Identification card, German Patent: DE 1,945,777.

Doddington, G. R. (1985). Speaker recognition - identifying people by their voices, *Proceedings of the IEEE* **73**(11): 1651–1664.

Drake, M. (1997). Holographic fingerprint sensor, *Optical Engineering* **35**(9): 2499–2505.

Drury, C., Mutsaers, C. & Hart, C. (1998). Low-cost all-polymer integrated circuits, *Applied Physics Letters* **73**(1): 108–110.

Duta, N., Jain, A. & Mardia, K. V. (2002). Matching of palmprints, *Pattern Recognition Letters* **23**(4): 477–485.

Edwards (1984). Fingerprint sensor, US Patent 4,429,413.

Ellinboe, J. (1972). Active element card, US Patent: US 3,637,994.

Engelbrecht, R., Hildebrand, C. & Jung, E. (1995). The smart card: an ideal tool for a computer-based patient record, *Medinfo* **8**(1): 344–348.

Ernst, R. H. (1971). Hand id system, US Patent: 3,576,537.

Etemad, K. & Chellapa, R. (1994). Discriminant analysis for recognition of human face images, *Proceedings of the International Conference on Acoustics, Speach and Signal Processing*, pp. 2148–2151. Atlanta, Georgia.

Fagan, M. (1992). *Finite Element Analysis: Theory and Practice*, Longman Scientific & Technical, Harlow, UK. ISBN: 0-582-02247-9.

Flohr, U. (1998). The smartcard invasion, *Byte* **32**(1): 76.

Fu, S., Laing, M., Shiramatsu, T. & Wu, T. (1981). Electrical characteristics of polymer thick film resistors, part I: Experimental results, *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* **4**(3): 289–293.

Funk, W., Finke, M. & Daum, H. (2000). Comparative study of biometric identification systems, Bundesamt für Sicherheit in der Informationstechnik Technical Report (BSI, German Information Security Agency)). Available from: http://www.bsi.de.

Furui, S. (1997). Recent advances in speaker recognition, *Pattern recognition letters* **18**(9): 859–872.

Gaines, R. S., Lisowski, W., Press, S. J. & Shapiro, N. (1980). Authentication by keystroke timing: Some preliminary results, RAND Publication Series: Report No. R-2526-NSF.

Galton, F. (1892). *Fingerprints*, MacMillan & Co., London, Great Britain.

Garcia, J. D. (1986). Personal identification apparatus, US Patent: US 4,621,334.

Garnier, F., Haklaouri, R., Yasser, A. & Srivastava, P. (1994). All-polymer field-effect transistor realized by printing techniques, *Science* **265**(5179): 1684–1686.

George, M. (2000). Voice authentication smart card, *Network Security* **20**(10): 5.

Gilleo, K. (1995). *Polymer Thick Film*, Klewer, Boston. ISBN: 0-442-01220-9.

Grabham, N. (2002). *Development of a Magnetostrictive Thick-Film Material for MEMS Applications*, PhD thesis, Department of Electronics And Computer Science, The University of Southampton, Highfield, Southampton SO17 1BJ, UK, Highfield, Southampton SO17 1BJ, UK.

Grassmann, T. & Karl, A. (2001). System on card: multiple applications, many possibilities, *Secure* **2**(1): 29–32. Available from Infineon Technologies, at: http://www.silicon-trust.com/pdf/secure_2_pdf/techno3.pdf.

Guillou, L. C., Ugon, M. & Quisquator, J.-J. (2001). Cryptographic authentication protocols for smart cards, *Computer Networks* **36**(4): 437–451.

Hachez, G., Koeunne, F. & Quisquater, J.-J. (2000). Biometrics, access control, smartcards: A not so simple combination, *Smart Card Research and Advanced Applications*, IFIP/TC8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Publishers, Bristol, UK, pp. 273–288. ISBN: 0-7923-7953-5.

Hale, J. M. & Tuck, J. (1999). A novel thick-film strain transducer using piezoelectric paint, *Proceedings of the IMechE Part C: Journal of Mechanical Engineering Science* **213**(C6): 613–622.

Halliday, D. & Resnick, R. (1989). *Fundamentals of Physics*, John Wiley & Sons. ISBN: 0-471-02456-2.

Hansmann, U., Nicklous, M. S., Schäck, T. & Seliger, F. (2000). *Smart Card Application Development Using Java*, Springer-Verlag, Berlin Heidelberg, Germany. ISBN: 3-540-65829-7.

Harkin, G. F. (1998). Hand biometrics sensing device, World Patent: WO 98/40962.

Harsányi, G. (1991). Polymer thick-film technology: a possibility to obtain very low cost pressure sensors?, *Sensors and Actuators A: Physical* **27**(1-3): 853–857.

Harsányi, G. (1995). Polymeric sensing films: new horizons in sensorics?, *Sensors and Actuators, A: Physical* **46**(1-3): 85–88.

Henderson, N. J. & Hartel, P. H. (2000). 'Pressure Sequence' - a novel method of protecting smart cards, *Smart Card Research and Advanced Applications*, IFIP/TC8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Publishers, Bristol, UK, pp. 241–256. ISBN: 0-7923-7953-5.

Henderson, N. J., Papakostas, T. V., White, N. M. & Hartel, P. H. (2001). Polymer thick film sensors: Possibilities for smartcard biometrics, *IOP Sensors and Their Applications XI*, Institute of Physics, IOP Publishing, London, pp. 83–88.

Henderson, N. J., Papakostas, T. V., White, N. M. & Hartel, P. H. (2002). Low-cost planar PTF sensors for the identity verification of smartcard holders, *IEEE Sensors 2002*, IEEE, Orlando, Florida, p. To Appear.

Henderson, N. J., White, N. M. & Hartel, P. H. (2001). iButton enrolment and verification requirements for the pressure sequence smartcard biometric, *in* I. Attali & T. Jensen (eds), *Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001*, Java Card Forum, Eurosmart, INRIA, Springer, Cannes, France, pp. 124–134. Lecture Notes in Computer Science Series, ISBN: 3-540-42610-8.

Henderson, N. J., White, N. M., Veldhuis, R., Slump, K. & Hartel, P. H. (2002a). A novel approach to the verification of smartcard holders, *IEEE Transactions on Systems, Man and Cybernetics* **?**(?). To be submitted (summer 2002).

Henderson, N. J., White, N. M., Veldhuis, R., Slump, K. & Hartel, P. H. (2002b). Sensing pressure for authentication and feedback, *IEEE Benelux Signal Processing Symposium*, IEEE, Leuven, Belgium, p. To Appear.

Hicks, W. T., Allington, T. R. & Johnson, V. (1980). Membrane touch switches: Thick-film materials systems and processing options, *IEEE Transactions on Components, Hybrids, and Manufacturing Technology* **3**(4): 518–524.

Higgins, A. L. & Wohlford, R. E. (1986). A new method of text-independent speaker recognition, *IEEE International Conference on Acoustics, Speech, and Signal Processing*, IEEE, Toyko, pp. 869–872.

Hill, R. (1998). *Retina Identification.* In (Jain et al. 1998).

Höjerback, P. (2000). Smartcard chips ready for fingerprint matching, *Biometric Technology Today* **8**(3): 5.

Hollingum, J. (1984). New sensing role for conductive polymer, *Sensor Review* **4**(1): 23–24.

Hollingum, J. (1999). They've got the measure of you, *Sensor Review* **19**(1): 27–30.

Holt, S. B. (1968). *The Genetics of Dermal Ridges*, Springfield, Ill. : C.C. Thomas.

Horowitz, P. & Hill, W. (1990). *The Art of Electronics (Second Edition)*, Cambridge University Press, Cambridge, UK, pp. 425–426. ISBN: 0-521-37095-7.

Huang, K. & Yan, H. (1997). Off-line signature verification based on geometric feature extraction and neural network classification, *Pattern Recognition* **30**(1): 9–17.

Hurley, D. J., Nixon, M. S. & Carter, J. N. (1999). Force field energy functionals for image feature extraction, *Proceedings of the British Machine Vision Conference*, BMVC, pp. 604–613.

Iannarelli, A. (1989). *Ear Identification. Forensic Identification Series*, Paramont Publishing Company, California, USA. ISBN: 0-962-31780-2.

Jacoby, I. H., Giordano, A. J. & Fioretti, W. H. (1972). Personal identification apparatus, US Patent: 3,648,240.

Jain, A., Bolle, R. & Pankanti, S. (1998). *Biometrics: Personal Identification in Networked Society*, Kluwer, Boston. ISBN: 0-792-38345-1.

Jain, A. & Duta, N. (1999). Deformable matching of hand shapes for verification, *Proceedings of the IEEE International Conference on Image Processing*, IEEE, Kangerlussuaq, Greenland.

Jain, A., Hong, L. & Pankanti, S. (2000). Biometric identification, *Communications of the ACM* **43**(2): 90–98.

Jain, A. K., Hong, L., Pankanti, S. & Bolle, R. (1997). An identity-authentication system using fingerprints, *Proceedings of the IEEE* **85**(9): 1365–1388.

Jain, A. K., Ross, A. & Pankanti, S. (1999). A prototype hand geometry-based verification system, *2nd International conference on Audio- and Video-based Person Authentication*, pp. 166–171. Washington DC.

Jain, A. & Pankanti, S. (2001). Biometrics systems: Anatomy of performance, *IEICE Transactions of Information Systems* **E84D**(7): 788–799.

Jones, L. & Chin, A. F. (1983). *Electronic Instruments and Measurements*, John Wiley & Sons, pp. 100–101. ISBN: 0-7923-7953-5.

Joshi, D. G., Rao, Y. V., Kar, S., Kumar, V. & Kumar, R. (1998). Computer-vision-based approach to personal identification using finger crease patterns, *Pattern Recognition* **31**(1): 15–22.

Joyce, R. & Gupta, G. (1990). Identity authentication based on keystroke dynamics, *Communications of the ACM* **33**(2): 168–176.

Jung, S., Thewes, R., Goser, K. & Weber, W. (1999). A low-power and high-performance CMOS fingerprint sensing and encoding architecture, *IEEE Journal of Solid-State Circuits* **34**(7): 978–984.

Kandel, E. R. (1981). *Principles of Neural Science*, Elsevier, North Holland. ISBN: 0-838-57701-6.

Klink, G. (2000). Ultra thin ICs open new dimensions for microelectronic systems, *Advancing Microelectronics* **27**(4): ??

Kohonen, T. (1988). *Self-Organization and Associative Memory*, Springer-Verlag, Berlin. ISBN: 0-387-51387-6.

Lamel, L. & Gauvain, J. (2000). Speaker verification over the telephone, *Speech Communication* **31**(2-3): 141–154.

Lapère, M. & Johnson, E. (1997). User authentication in mobile telecommunication envoronments using voice biometrics and smartcards, *4th International Conference on Intelligence in Services and Networks (IS&N 97)*, Lecture Notes in Computer Science, Cernobbio, Italy, pp. 437–443.

Lay, Y. L. (2000). Hand shape recognition, *Optics & Laser Technology* **32**(1): 1–5.

Leach, J. (1995). Dynamic authentication for smartcards, *Computers & Security* **14**(5): 385–389.

Lee, H. C. & Gaensslen, R. E. (1991). *Advances in Fingerprint Technology*, Elsevier, New York. ISBN: 0-849-30923-9.

Lefki, K. & Dormans, G. (1994). Measurement of piezoelectric coefficients of ferroelectric thin films, *Journal of Applied Physics* **76**(3): 1764–1767.

Leggett, J. & Williams, G. (1988). Verifying identity through keystroke characteristics, *International Journal of Man-Machine Studies* **28**(1): 67–76.

Leggett, J., Williams, G. & Usnick, M. (1991). Dynamic identity verification via keystroke characteristics, *International Journal of Man-Machine Studies* **35**(6): 859–870.

Leppävuori, S., Väänänen, J., Lahti, M., Remes, J. & Uusimäki, A. (1994). A novel thick-film technique, gravure offset printing, for the realization of fine-line sensor structures, *Sensors and Actuators A: Physical* **42**(1-3): 593–596.

Leung, L. M., Kwong, C. F., Kwok, C. C. & So, S. K. (2000). Organic polymer thick film light emitting diodes (PTF-OLED, *Displays* **21**(5): 199–201.

Loasby, R. G. & Holmes, P. J. (1976). *Handbook of Thick Film Technology*, Electrochemical Publications Ltd., Ayr, Scotland. ISBN: 0-901-15005-3.

Lockie, M. (2000). Ear, eye, gait, keystroke, lip and nailbed biometrics, *Biometric Technology Today* **7**(9): 8–9.

Lockie, M. (2001a). Developments in finger imaging are announced, *Biometric Technology Today* **2001**(4): 4.

Lockie, M. (2001b). Hand measuring systems, *Biometric Technology Today* **2001**(4): 9–11.

Löfberg, B. (1986). Data carrier, US Patent: US 4,582,985.

Lundberg, B. & Sundqvist, B. (1986). Resistivity of a composite conducting polymer as a function of temperature, pressure and environment: Applications as a pressure and gas concentration transducer, *Journal of Applied Physics* **60**(3): 1074–1079.

Mainguet, J., Pegulu, M. & Harris, J. (2000). Fingerprint recognition based on silicon chips, *Future Generation Computer Systems* **16**(4): 403–415.

Maness, W. L., Golden, R. F., Benjamin, M. H. & Podoloff, R. M. (1988). Contact sensor for measuring dental occlusion, US Patent: US 4,734,034.

Maness, W. L., Golden, R. F., Benjamin, M. H. & Podoloff, R. M. (1989). Pressure and contact sensor system for measuring dental occlusion, US Patent: US 4,856,993.

Mansfield, T., Kelly, G., Chandler, D. & Kane, J. (2001). Biometric product testing final report, National Physical Laboratory Technical Report. Available from: http://www.cesg.gov.uk/technology/biometrics.

Matsui, T. & Furui, S. (1993). Concatenated phenome models for text-variable speaker recognition, *Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing*, IEEE, Minneapolis, pp. 391–394.

Matweb (2002). On-line materials database, http://www.matweb.com.

Miller, B. (1994). Vital signs of identity, *IEEE Spectrum* **31**(2): 22–30.

Miller, R. P. (1971). Finger dimension comparison identification system, US Patent: 3,576,538.

Monrose, F. & Rubin, A. (1997). Authentication via keystroke dynamics, *4th ACM Conference on Computer & Communications Secutity*, Association for Computing Machinery, pp. 48–56.

Monrose, F. & Rubin, A. D. (2000). Keystroke dynamics as a biometric for authentication, *Future Generation Computer Systems* **16**(4): 351–359.

Moreno, R. (1974). Data storage syatems, French Patent: FR 7,410,191. See also US 3,971,916.

Morgan-Matroc (1998). Piezoelectric ceramics data book.

Morris, A. S. (1988). *Principles of Measurement and Instrumentation*, Prentice Hall International, UK, pp. 93–109. ISBN 0-137-10161-9.

M'Raïhi, D. & Yung, M. (2001). E-commerce applications of smart cards, *Computer Networks* **36**(4): 453–472.

Nalwa, V. S. (1997). Automatic on-line signature verification, *Proceedings of the IEEE* **85**(2): 215–239.

Naszlady, A. & Naszlady, J. (1998). Patient health record on a smart card, *International Journal of Medical Informatics* **48**(1-3): 191–194.

Neumann, J. & Gabriel, K. (2002). CMOS-MEMS membrane for audio-frequency acoustic actuation, *Sensors and Actuators A: Physical* **95**(2-3): 175–182.

Newham, E. (2000). Signature verification technologies, *Biometric Technology Today* **8**(1): 8–11.

Nixon, M. S., Carter, J. N., Cunado, D., Huang, P. S. & Stevenage, S. V. (1998). *Gait Recognition.* In (Jain et al. 1998).

Obaidat, M. S. & Sadoun, B. (1997). Verification of computer users using keystroke dynamics, *IEEE Transactions on Systems, Man and Cybernetics - Part B - Cybernetics* **27**(2): 261–269.

Obiadat, M. S. (1998). *Keystroke Dynamics Based Authentication.* In (Jain et al. 1998).

Orr, R. (2000). The smart floor: A mechanism for natural user identification and tracking, GVU Technical Report: GIT-GVU-00-02. Graphics, Visualization and Usability Center, Gorgia Institute of Technology, Atlanta, GA USA.

Pankanti, S., Prabhakar, S. & Jain, A. (2001). On the individuality of fingerprints, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. To be published.

Papakostas, T. V. (2001). *Polymer Thick-Film Sensors and Their Integration With Silicon: A Route To Hybrid Microsystems*, PhD thesis, Department of Electronics And Computer Science, The University of Southampton, Highfield, Southampton SO17 1BJ, UK, Highfield, Southampton SO17 1BJ, UK.

Papakostas, T. V., Harris, N. R., Beeby, S. P. & White, N. M. (1998). Piezoelectric thick-film polymer pastes, *in* N. White (ed.), *Proceedings of the 12th European Conference on Solid-State Transducers - 9th UK Conference on Sensors and Their Applications*, Institute of Physics, pp. 461–464.

Papakostas, T. V. & White, N. M. (2000a). Influence of substrate on the gauge factor of polymer thick-film resistors, *Journal of Physics D: Applied Physics* **33**(14): L73–L75.

Papakostas, T. V. & White, N. M. (2000b). Polymer thick film sensors and applications, *IOP Sensors and Their Applications X*, Institute of Physics, IOP Publishing, Cardiff, UK, pp. "125–130".

Papakostas, T. V. & White, N. M. (2000c). Screen printable polymer piezoelectrics, *Sensor Review* **20**(2): 135–138.

Parkes, J. R. (1991). *Personal Identification - Biometrics*, Elsevier Science, North Holland, pp. 181–191. In Information Security (D.T. Lindsay and W.L. Price, eds.).

Penrose, L. S. (1969). Dermatoglyphics, *Scientific American* **221**(6): 73–84.

Persaud, K. C., Lee, D.-H. & Byun, H.-G. (1998). *Objective Odour Measurements.* In (Jain et al. 1998).

Phillips, P. J., Moon, H., Rizvi, S. A. & Rauss, P. J. (2000). The FERET evaluation methodology for face-recognition algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(10): 1090–1104.

Phipps, T. & King, R. (1997). A speaker verification biometric in 40 bytes, *CardTech/SecurTech Conference, Volume I*, Orlando, Florida, pp. 187–198.

Plamondon, R. & Lorette, G. (1989). Automatic signature verification and writer identification-the state of the art, *Pattern Recgonition* **22**(2): 107–131.

Plamondon, R. & Srihari, S. N. (2000). On-line and off-line handwriting recognition: A comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1): 63–84.

Praca, D. & Barral, C. (2001). From smart cards to smart objects: the road to new smart technologies, *Computer Networks* **36**(4): 381–389.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1993). *Numerical Recipies in C*, Cambridge University Press, Cambridge, United Kingdom. ISBN: 0-521-43108-5.

Prokoski, F. J. & Riedel, R. B. (1998). *Infrared Identification of Faces and Body Parts.* In (Jain et al. 1998).

Quisquater, J.-J. (1997). The adolescence of smart cards, *Future Generation Computer Systems* **13**(1): 3–7.

Rankl, W. & Effing, W. (1997). *The Smart Card Handbook*, John Wiley & Sons, Chichester, England. ISBN: 0-471-96720-3.

Rey, P., Charvet, P., Delay, M. & Hassan, S. A. (1997). A high density capacitive pressure sensor array for fingerprint sensor application, *Tranducers '97 - IEEE International Conference on Solid-State Sensors and Actuators*, IEEE, Chicago, USA, pp. 1453–1456.

Rice, J. (1985). Method and apparatus for the identification of individuals, UK Patent: GB 2,156,127.

Rivest, R. L., Shamir, A. & Adleman, L. M. (1978). A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* **21**(2): 120–126.

Roark, R. J. & Young, W. C. (1975). *Formulas for Stress and Strain (5th Ed.)*, McGraw-Hill, Inc., USA, p. 104. ISBN 0-070-53031.

Robertson, C., Shipton, R. D. & Grey, D. R. (1999). Minature sensors using high density screen printing, *Sensor Review* **19**(1): 33–36.

Robinson, J. A., Liang, V. M., Chambers, J. A. M. & MacKenzie, C. L. (1998). Computer user verification using login string keystroke dynamics, *IEEE Transactions on Systems, Man and Cybernetics - Part A - Systems and Humans* **28**(2): 236 – 241.

Rombach, P., Müllenborn, M., Klein, U. & Rasmussen, K. (2002). The first low voltage, low noise differential silicon microphone technology development and measurement results, *Sensors and Actuators A: Physical* **95**(2-3): 196–201.

Rudin, N. (1998). *DNA Based Identification.* In (Jain et al. 1998).

Rumelhart, D. E. & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance, *Cognitive Science* **6**(1): 1–36.

Sanchez-Reillo, R. (2001). Smart card information and operations using biometrics, *IEEE AESS Systems Magazine* **16**(4): 3–6.

Sanchez-Reillo, R., Sanchez-Avila, C. & Gonzales-Marcos, A. (2000). Biometric identification through hand geometry measurements, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(10): 1168–1171.

Schalkoff, R. (1992). *Pattern Recognition - Statistical, Structural and Neural Approaches*, John Wiley & Sons, New York. ISBN: 0-471-52974-5.

Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York. ISBN: 0-471-11709-9.

Shaffer, L. H. (1982). Rhythm and timing in skill, *Psychological Review* **89**(2): 109–122.

Shigematsu, S., Morimura, H., Tanabe, Y., Adachi, T. & Machida, K. (1999). A single-chip fingerprint sensor and identifier, *IEEE Journal of Solid-Start Circuits* **34**(12): 1852–1859.

Shu, W. & Zhang, D. (1998). Automated personal identification by palmprint, *Optical Engineering* **37**(2): 2359–2362.

Sidlauskas, D. P. (1988). 3d hand profile identification apparatus, US Patent: 4,736,203.

Singh, S. (1999). *The Code Book (Chapter 3: The Mechanisms of Secrecy)*, Fourth Estate Limited, London, pp. 101–142. ISBN: 1-85702-879-1.

Spillane, R. J. (1975). Keyboard apparatus for personal identification, *IBM Technical Disclosure Bulletin* **17**: 3346.

Stevens, S. (1960). The psychophysics of sensory function, *American Scientist* **48**(1): 226–253.

Stiver, J. & Peterson, D. C. (1998). Identification system, US Patent: US 5,793,881.

Tartagni, M. & Guerrieri, R. (1998). A fingerprint sensor based on the feedback capacitive sensing scheme, *IEEE Journal of Solid-State Circuits* **33**(1).

Timoshenko, S. P. & Goodier, J. N. (1970). *Theory of Elasticity (Third Edition)*, McGraw-Hill, New York, USA, pp. 6–12. ISBN: 0-070-64720-8.

Tishby, N. Z. (1991). On the application of mixture ar hidden markov models to text independant speaker recognition, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **39**(3): 563–570.

Topping, A., Kuperschmidt, V. & Gormley, A. (1998). Method and apparatus fot the automated identification of individuals by the nail beds of their fingernails, US Patent: US 5,751,835.

Torkkeli, A., Rusanen, O., Saarilahti, J., Seppä, H., Sipola, H. & Hietanen, J. (2000). Capacitive microphone with low-stress polysilicon membrane and high-stress polysilicon backplate, *Sensors and Actuators A: Physical* **85**(1-3): 116–123.

Townend, R. (1995). The international smart card industry guide 1995-96. Report is available from http://www.smartcard.co.uk.

Turk, M. & Pentland, A. (1991). Eigenfaces for recognition, *Journal of Cognitive Neuroscience* **3**(1): 7186.

Umphress, D. & Williams, G. (1985). Identity verification through keyboard characteristics, *International Journal of Man-Machine Studies* **23**(3): 263–273.

van der Putte, T. & Keuning, J. (2000). Biometrical fingerprint recognition: Don't get your fingers burned, *Smart Card Research and Advanced Applications*, IFIP/TC8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Publishers, Bristol, UK, pp. 19–38. ISBN: 0-7923-7953-5.

Verschuren, T. (1998). Smart access: Strong authentication on the web, *Computer Networks and ISDN Systems* **30**(16-18): 1511–1519.

White, N. & Turner, J. (1997). Thick-film sensors: past, present and future, *Measurement Science and Technology* **8**(1): 1–20.

Wildes, R. P. (1997). Iris recognition: An emerging biometric technology, *Proceedings of the IEEE* **85**(9): 1348–1363.

Willmore, M. R. (1994). Infra-red imaging and pattern recognition system, US Patent: US 5,351,303.

Wu, Q.-Z., Lee, S.-Y. & Jou, I.-C. (1998). On-line signature verification based on logarithmic spectrum, *Pattern Recognition* **31**(12): 1865–1871.

Yang, L., Widjaja, B. K. & Prasad, R. (1995). Application of hidden markov models for signature verification, *Pattern Recognition* **28**(2): 161–170.

Young, J. R. & Hammond, R. W. (1989). Method and apparatus for verifying an individual's identity, US Patent: US 4,805,222.

Young, N. (1997). Novel fingerprint scanning arrays using polysilicon TFTs on glass and polymer substrates, *IEEE Electron Device Letters* **18**(1): 19–20.

Zhang, D. & Shu, W. (1999). Two novel characteristics in palmprint verification: datum point invariance and line feature matching, *Pattern Recognition* **32**(4): 691–702.

Zoreda, J. L. & Otón, J. M. (1994). *Smart Cards*, Artech House, Inc., Norwood, MA 02062, USA. ISBN: 0-89006-687-6.

Zovco, C. I. & Nerz, T. C. (1999). White polymer thick film electroluminescent lamps and their applications for backlighting liquid crystal displays in portable electronic devices, *Displays* **20**(3): 155–159.

Zunkel, R. L. (1998). *Hand Geometry Based Verification.* In (Jain et al. 1998).