Biometric Security Using Finger Print Recognition

Subhra Mazumdar, Venkata Dhulipala
University of California, San Diego

Abstract–**Our goal is to implement finger print recognition on the PXA27x DVK platform in view of increasing popularity of biometric security for digital handheld devices. To this end we are using Siemens ID Mouse finger print capture device mentioned in [5] that scans the finger print and gives out a high quality image while maintaining sub-pixel geometric accuracy. This image will then be stored and processed to extract the critical pixel information (minutiae). Subsequent scans on any finger print other than that of the authorized user(s) will be disallowed. A secure process will be then forked based on the authorization. The image recognition algorithm has to be highly precise and also efficient enough to enable accurate instantaneous access.**

1. Project Background

The analysis of fingerprints for matching purposes generally requires the comparison of several features of the print pattern. These include patterns, which are aggregate characteristics of ridges, and minutia points, which are unique features found within the patterns. It is also necessary to know the structure and properties of human skin in order to successfully employ some of the imaging technologies. The three basic patterns of fingerprint ridges are the arch, loop, and whorl. An arch is a pattern where the ridges enter from one side of the finger, rise in the center forming an arc, and then exit the other side of the finger. The loop is a pattern where the ridges enter from one side of a finger, form a curve, and tend to exit from the same side they enter. In the whorl pattern, ridges form circularly around a central point on the finger.

1.1 Fingerprint sensors

A fingerprint sensor details of which can found in [4] captures the digital image of a fingerprint pattern called *live scan*. This live scan is digitally processed to create a biometric template (a collection of extracted features) which is stored and used for matching. The commonly used fingerprint sensor technologies are optical, ultrasonic and capacitive. Siemens ID Mouse falls under the capacitive category of fingerprint sensors. Capacitance sensors utilize the principles associated with capacitance in order to form fingerprint images. The two equations used in this type of imaging are:

$$C = \frac{Q}{V}$$

$$C = \epsilon_0 \epsilon_r \frac{A}{d}$$

where, C is the capacitance in farads, Q is the charge in coulombs, V is the potential in volts, ε_0 is the permittivity of free space, measured in farad per metre, ε_r is the dielectric constant of the insulator used, A is the area of each plane electrode, measured in square metres, d is the separation between the electrodes, measured in metres.

As shown in Fig. 1 texture of the fingerprint formed by cured alternation of the ridges and valleys has to be transformed into the digital image. For this purpose, the distance between sensor surface and finger ridges or valleys is transformed to the discrete information, and from this information grey level representation of discrete levels is given on the output. The underlying principles are discussed in greater detail in [7].

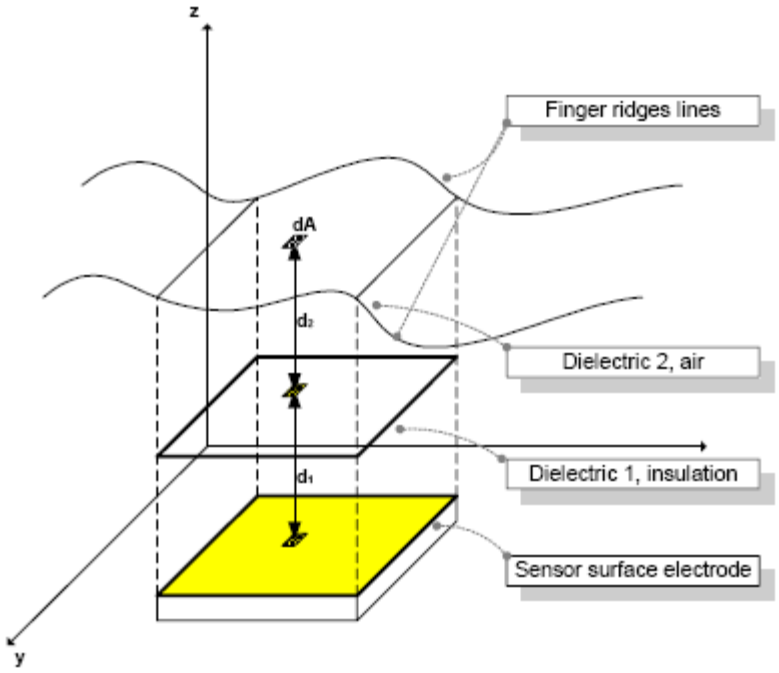

Fig.1

In this method of imaging, the sensor array pixels each act as one plate of a parallel-plate capacitor, the dermal layer (which is electrically conductive) acts as the other plate, and the non-conductive epidermal layer acts as a dielectric. Listed below are some state-of-the-art commercially available finger print scanners –

Technology	Company	Model	Dpi	Area (h×w)	Pixels
Capacitive (sweep)	Fujitsu www.fme.fujitsu.com	MBF300	500	0.06"×0.51"	32×256 (H×256)
Capacitive	Infineon www.infineon.com	FingerTip	513	0.56"×0.44"	288×224 (64,512)
Capacitive	ST-Microelectronics us.st.com	TouchChip TCS1AD	508	0.71"×0.50"	360×256 (92,160)

where, Dpi stands for dots-per-inch, a quantity that correlates with image resolution

1.2 Matching Algorithms

Matching algorithms are used to compare previously stored templates of fingerprints against candidate fingerprints for authentication purposes. In order to do this either the original image must be directly compared with the candidate image or certain features must be compared.

1.2.1 Pattern-based (or image-based) algorithms

Pattern based algorithms compare the basic fingerprint patterns (arch, whorl, and loop) between a previously stored template and a candidate fingerprint. This requires that the images be aligned in the same orientation. To do this, the algorithm finds a central point in the fingerprint image and centers on that. In a pattern-based algorithm, the template contains the type, size, and orientation of patterns within the aligned fingerprint image. The candidate fingerprint image is graphically compared with the template to determine the degree to which they match and a match score is generator.

1.2.2 Minutia Feature extraction based algorithms

Other algorithms use minutiae features on the finger. The major Minutia features as shown in Fig.2 of fingerprint ridges are: ridge ending, bifurcation, and short ridge (or dot). The ridge ending is the point at which a ridge terminates. Bifurcations are points at which a single ridge splits into two ridges. Short ridges (or dots) are ridges which are significantly shorter than the average ridge length on the fingerprint. Minutiae and patterns are very important in the analysis of fingerprints since no two fingers have been shown to be identical.

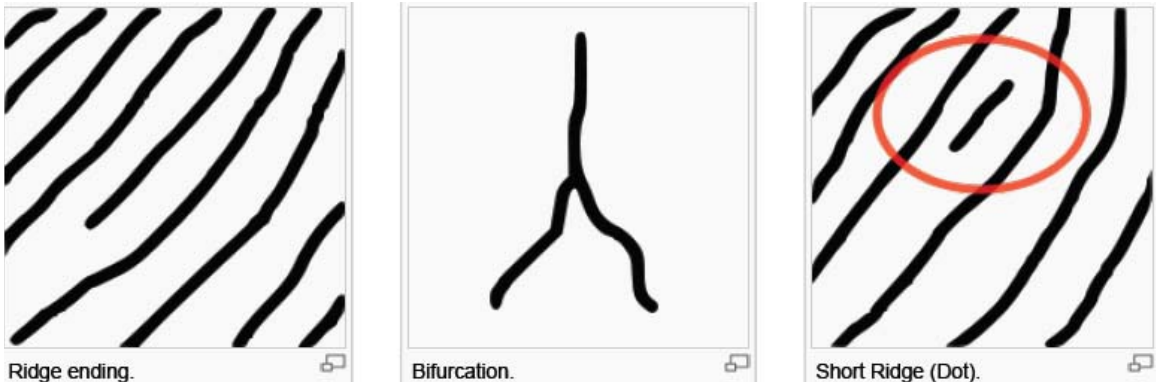

Fig.2

2. Project Description

2.1 Fingerprint Recognition Using Siemens ID Mouse

We are using Siemens ID Mouse for our project. It is a USB2.0 device that uses capacitive fingerprint reader. The PXA27x platform support Linux kernel bundle version of up to 2.6.9 as stated in the Mainstone document found in [2], whereas the drivers of Siemens ID Mouse are available in versions 2.6.10 and above. So we had to download and make the ID Mouse driver patch to 2.6.9 and re-compile the kernel to run on the target device. An appropriate patch was obtained from the authors of the driver mentioned in [1].

2.2 Configuring and Accessing ID Mouse through PXA27x

Linux kernel version 2.6.9 recognizes the device connected on the USB port but does not create the corresponding device system file to access the fingerprint obtained from the device. For this purpose we had to manually create a device system file with the *mknod* bash command that creates a device file with arguments, device type, minor and major numbers. We used *mknod /dev/idmouse0 c 180 250* to match the kernel version 2.6.9 and had to edit *mainstone_defconfig* configuration to include the *CONFIG_USB_IDMOUSE* and *CONFIG_USB_DEVFS* switches without which the ID Mouse driver does not work. These help the device to communicate appropriately with platform through the driver software. ID Mouse was accessed through the system *open* and *read* calls on the device file. The *open* call returns the file handle of the device and stores the fingerprint image into a buffer. A subsequent *read* returns a block-wise pointer to read the image as a PGM file, the minutiae information of which is subsequently be used to implement the recognition algorithm.

2.3 Fingerprint Image Capture, Storage and Minutiae Detection

2.3.1 *libfprint* Open Source Library and Cross Compilation

We use the *libfprint* open source fingerprint recognition library to handle image capture, enrollment, verification and identification. As stated in [3] and [6] the current model of *libfprint* is that a fingerprint is enrolled (storing its minutiae data), and at the time of authentication, the pre-stored (enrolled) minutiae data is compared to a *live scan*. In other words, *libfprint* can't compare two images in situ. *libfprint* uses state of the NBIS algorithms: MINDTCT – Minutiae Detection Algorithm and BOZORTH3 – Minutiae Matching algorithms, the details related to which can be found in [6]. The similarity between two fingerprints is returned in the form of a parameter called match score. *libfprint* provides enrollment, verification and identification API. Enrollment is a process in which a user's fingerprint is scanned, processed and stored for later comparison. *libfprint* extracts the minutiae information from the image and if quality of detected minutiae is above a threshold it declares it as a successfully enrollment. Verification is a process in which user provides one's identity and the system compares the scanned fingerprint against enrolled fingerprint data for that user identity and declares a match or

no match. In Identification, a user provides one's fingerprint scan which is compared against all previously enrolled fingerprint data to find the identity.

We cross compiled the *libfprint* fingerprint recognition library which in turn uses the *ImageMagick*, *Libusb*, *Libcrypto* and *Glib* packages for implementing API. We had to link and cross-compile all the packages before compiling *libfprint*. Installing these packages was a challenging task involving resolution of multiple dependency chains before successfully compiling the library of interest. Several forums had to be consulted to understand the package configuration options before compiling them. The *Libcrypto* library was used only in a limited of the source, so, its dependency and the dependent files were removed from the library.

2.3.2 Image Capture Using Sample Application

A small application was written to initialize the device and capture the image using *libfprint* library functions. We had to resolve arm-linux-gcc compilation, link-load errors. These errors were due to the way which static and dynamic libraries have to specified for compilation and linking an application. Consequently the *arm-linux-gcc –L <lib_path> <app.c>* command line options had to be used for our application to successfully pick up all the dependent static/dynamic libraries of *libfprint, ImageMagick*, *Libusb* and *Glib* packages and additionally soft links to these library objects had to be made in the */usr/include* directory of the platform for running the application executable. After this the image was successfully captured into PGM file and its minutiae were extracted using the functions in *libfprint.*

2.4 The Algorithm and Performance Evaluation

Fig.3 shows the algorithm that enrolls, identifies and verifies a user using the functions in *libfprint* library for minutiae detection and identification. Image goodness factor is the number of minutiae returned by the feature extraction function in the library. Image matching against the database is done through comparison of the minutiae and subsequent score generation based on the degree of match. The library function BOZORTH3 does a fairly good job in image identification and verification and provides the flexibility of setting the score threshold through only above which the fingerprint is to be considered authorized. The performance of any fingerprint identification system is measured in terms of two parameters i.e. False Positive and False Negative. False Positive is the situation in which an invalid user is granted access to the system while False Negative is the situation in which a valid user is denied access to the system. For any identification system it is very important not to grant access to an unauthorized user, so false positive should be very low and ideally zero. While false negative is an inconvenience faced by a valid user in which he has to scan his finger again. Thus, in order to develop a more secure and accurate identification system the rate of these two parameters should be as low as possible. With a BOZORTH score threshold of 40 we observed 0 false positive rate and 1 out of 10 false negative attempts. Finally we have a function that can delete an enrolled user's record from the database.

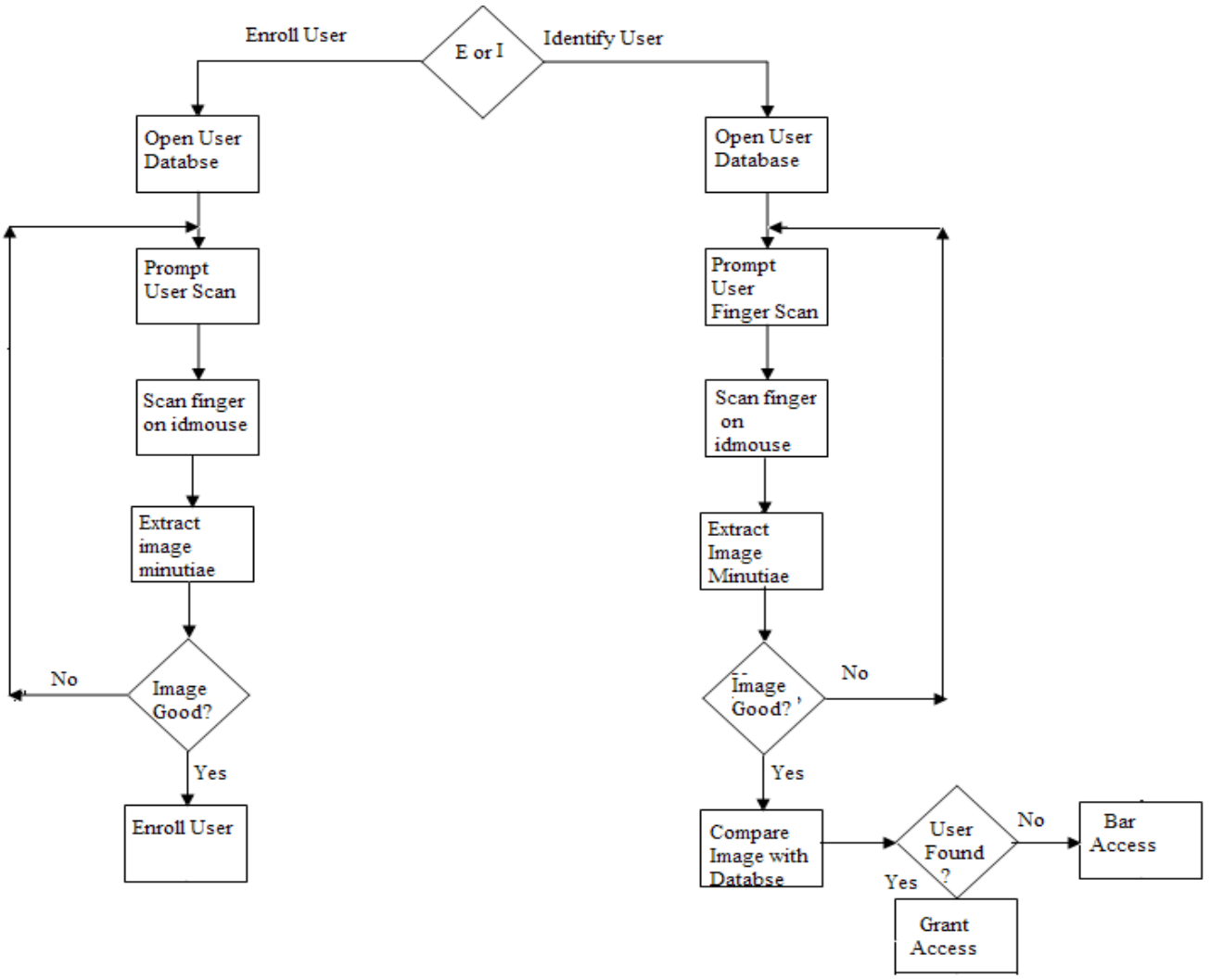

Fig.3

3. Conclusion and Future Projections

Through out experience in this project we learned to build a reliable Embedded Fingerprint Identification System using open source software components. This was an excellent embedded system design exercise, which involved Researching existing technologies and solutions, Requirement Definition (defining the project objective which can be achieved in given time frame), Project Planning, System Analysis and Design (Deciding components of the system and their interactions), Implementation, Integration and Testing. This project is close to a real-life project which can produce a useful embedded system to be deployed in close to life applications like that of an ATM machine, a vending machine or even access to secure areas. This application can in future be ported to a real-time user authentication embedded system with further enhancements like building the authorized user database and storing it into a central database like memory drive and scanning from that database for authentication. Also the application could be compressed and made compatible with for different operating systems resulting

in an interesting yet ambitious application of Biometric Chip Cards which is a credit with an embedded integrated circuit that is programmed to be activated for a transaction only upon accepting owner's fingerprint scan. Many such innovative applications can be designed around fingerprint recognitions.

4. References

[1] Florian Echtler; "idmouse" – http://www.fs.tum.de/~echtler/idmouse/

[2] Xscale-PXA27x Platform- http://cse.ucsd.edu/classes/wi08/cse237a/miniproj/doc.zip

[3] Jake Edge, "Fingerprint recognition using fprint" - http://lwn.net/Articles/259363/

[4] Fingerprint Recognition - http://en.wikipedia.org/wiki/Fingerprint_recognition

[5] Siemens ID Mouse - http://www.siemensidmouse.com/

[6] "The libfprint project" - http://www.reactivated.net/fprint/wiki/Main_Page

[7] "Characteristics and Application of the Fingerprint Recognition Systems", I. Plajh et al, MEASUREMENT SCIENCE REVIEW, Volume 3, Section 2, 2003