Portable Trust: Blockchain identities and biometric-based authentication

Julian Faber[1], Lucio Guerchi[1], Joren Hammudoglu[1], Shashank Rao[1], Johannes Rauhamaa[1], Ioannis Petros Samiotis[1], Jamey Sparreboom[1]

***Abstract*— In today's world, security and safety is in high demand. In this project we implemented a safe biometric-based authentication through Android systems' cameras. The users use the camera of their phone to acquire their fingerprint, the image is then processed and the appropriate features are been extracted. These features are then being used for matching and authenticated the user.**

This research is conducted in the context of the "Hacking Lab Applied security analysis" course, given at Delft University of Technology.

I. INTRODUCTION

Contracts, transactions, and the records of them are among the defining structures in our economic, legal, and political systems. They protect assets and set organisational boundaries. They establish and verify identities and chronicle events[**?**].

We created a strong biometric-based identification and authentication primitive to build trust and make trust portable.

First we propose an architecture to store identities in a tamper-proof manner using append-only data structures, now dubbed *the blockchain*. Second, we implemented an open source identification and authentication tool using fingerprint matching. Third, we specifically crafted our technology to be fully permissionless, at all levels in our architecture we do not require permission of organisations, governments, or appointed holders of records to create trustworthy identifiers.

We addressed the problem of creating trust with a minimal amount of infrastructure, zero institutional embedding and lack of preexisting trust. On the technical side we used existing encryption libraries, solid open source image processing frameworks, and robust fingerprint minutiae key point matching.

For our portable biometric-based trust, we implemented a fingerprint identification system for Android smartphones. The main reasons for using fingerprints as our biometric-based trust is the ability to implement it using a smartphone's camera (a feature that the majority of phones have nowadays) and the vast preexisting research that has been conducted on the subject. Lastly, fingerprints are a unique biometric information for humans and it is being widely used for identification. The reason we wanted to use the smartphone's camera for acquiring fingerprints was that not all Android smartphones are equipped with fingerprint scanners and the fingerprint data from these scanners can only be used for authenticating the phone's user and not share the data with the phone's applications. Finally, our architecture is permission-less and open source, by using the smartphone camera we remove the need to ask the fingerprint scanner manufacturer for approval.

[1]Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2600 AA Delft, The Netherlands

II. RELATED WORK

In this section we analyze a selection of projects and algorithms that are related to our implementation. This gives us the opportunity to present our main references and explain the differences that our application has compared to them. The selection of the related work was based on how close they are to our implementation and on their popularity in the community of fingerprint extraction and identification.

A. Fingerprint Matching on Android using OpenCV

This resource [3] is found in OpenCV forums, in the "Answers" section, where the process of fingerprint matching in Android is presented by a user. The explanation is extensive and each step is described in detail. We used this reference as an example of fingerprint extraction and matching technique to study the process that needed to be implemented in our application.

The suggested process can be summarized as follows:

- Fingerprint acquisition through camera
- Quality determination of image
- Image preprocessing and extraction of skeleton map
- Feature extraction

- Fingerprint search and match

This algorithm is the basis of our application's processes (explained in IV) and it was altered to reflect the needs of our project in security and robustness.

Based on the time frame available for the application's implementation, we made the decision to follow parts of the image processing but apply a different algorithm for feature extraction.

B. RxFinger Print

Rxfingerprint [1] project's objective is to secure users information via fingerprint authentication using Android Fingerprints API [2]. Since this project is based on Android's Fingerprint API, it essentially uses different techniques for feature extraction. Nevertheless, the encryption-decryption practices used here though, were used as a study on fingerprint authentication in Android devices.

C. Fingerprint Recognition

In this GitHub project, part of Advanced Computer Vision' subject, MSc Artificial Intelligence of the University of Southampton [13]. Using OpenCv [14] libraries for Android OS; feature extraction is accomplished using a Gabor filter [6], ridge orientation and SIFT [4], and feature matching using SIFT [4] and RANSAC [5].

Our image processing pipeline is based on the one described by this project:

- Raw Image
- Grayscale
- Masking
- Histogram equalization

with different stage implementations such as the use of Oriented FAST and Rotated BRIEF (ORB) instead of SIFT.

III. ALGORITHM & IMPLEMENTATION OF FINGERBLOX

The FingerBlox is a system for fingerprint authentication through the integrated camera in Android devices. It is divided into 3 major components: fingerprint acquisition, the image processing pipeline, and fingerprint matching. We further explain each step in the following subsections and we provide an overview of the application's pipeline in the end of the section. UI demonstration of each step can be found in Figure **??**.

A. Image acquisition

As explained in Introduction, in FingerBlox implementation we wanted to make use of the Android camera for fingerprint acquisition. This helps the application to be more easily available to users with smartphones not equipped with fingerprint scanners. Also, extracting fingerprint data from integrated fingerprint sensors is either made a difficult process or totally impossible deliberately by the phone's company for security reasons, thus making the fingerprint acquisition through camera the ideal option.

The image acquisition process is as follows: the user opens the application and he is prompted to take a picture of his fingerprint inside the elliptic area. The elliptic area is being used as a mask for the cropping step later, in the image processing step. The application then makes use of the built-in camera functionality to take a picture of the finger placed inside the area. If the application doesn't detect skin in the area, then the picture cannot be captured. Once the picture is taken, the captured image is processed to extract features from the user's fingerprint.

B. Image processing pipeline

The implemented image pipeline has an ordered sequence of processing stages using OpenCV [14]. Starting with skin detection using HSV (hue, saturation, and value (brightness)) color space approach [7], the first step of this stage is to convert the RBG image taken with the camera to HSV color space and apply a human skin color threshold. This is followed by erosion [8] and dilation [8] morphological operations to isolate the fingerprint from the rest of the image. As final step of this stage a Gaussian blur linear filter is applied to reduce noise.

The second stage converts the image color to gray-scale for histogram equalization, also the image can be processed more easily by the hardware due to less information on each pixel.

In the third stage, the image is elliptically cropped using a skin detection mask to isolate the fingerprint from the rest of the image. After this point further stages focus on enhancing and extraction of fingerprint minutiae features.

In the fourth stage Histogram equalization is performed as contrast enhancement to distinguish ridges (darker curves) valleys (brighter curves) as explained by Gao, Q. et al.[10].

For the fifth stage skeletonization is used to produce a lightweight representation of the fingerprint.

During the sixth stage thinning is achieved using the Zhang-Suen algorithm [11] to reduce the width of the ridges to one pixel.

In the seventh stage, the minutiae key point detection algorithm is executed to find ridge endings and bifurcations by iterating over pixels and classifying image locations in rectangular coordinates(x,y) as ridge endings when the it has one neighbor pixel and as bifurcations when neighbors pixel number equals three.

Finally, Oriented FAST and Rotated BRIEF (ORB) calculate descriptors. ORB was selected because is two times faster than SIFT as Rublee, E. et al. [12] demonstrated it, additionally is more suitable for low-power consumption devices such as smartphones. Descriptors are stored and used for matching purposes.

C. Matching

Fingerprint matching is done using the ORB descriptors obtained as described in previous section and compared with the stored ones. After that step, a match test is being used to match the fingerprints and grant access to the user.

D. User Experience

Screenshots: capture, (real-time?) processing (pipeline), feature extraction, matching performance (top10)

IV. EVALUATION

The performance of our system is evaluated against a small dataset of 20 fingers belonging to one of the authors. Of these 20 finger images, 10 belong to the same right hand, where 5 are used for extracting feature points and other 5 for matching. It is similarly repeated for the left hand.

The reference used for naming the fingers is shown in figure 1.

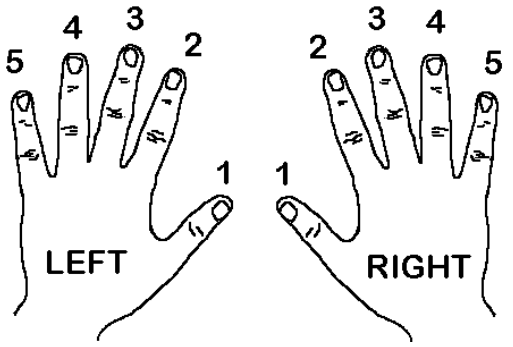

Fig. 1: Reference used for naming fingers.

The matching algorithm gives percentage of matching between two images. This metric is currently used as a performance measure for the algorithm. A higher percentage indicates a higher confidence over that fingerprint matching.

Table I and Table II show the matching percentages of Left and Right hand respectively. The finger labels on the Y-axis represent the images that have been used for extracting feature points and the image labels on X-axis are the images against which they have been tested.

The mean percentage of performance of our system is the mean of the matching percentages represented in the diagonal values of the table I and table II. The mean matching accuracy is 55%.

Since our image processing pipeline was built to identify real-world finger images, we could not perform evaluation on available finger images dataset like NIST. NIST dataset has gray-scale or binary images which would not be ideal when implementing our *skin detection* algorithm. Due to the limitation of evaluation data, other performance metrics like precision, recall and F1-scores could not be measured.

	L1	***L2***	***L3***	***L4***	***L5***
L1	52	3	42	11	5
L2	11	56	26	14	10
L3	7	37	59	42	4
L4	14	17	23	67	9
L5	5	4	8	4	35

TABLE I: Comparison of Matching accuracies across each finger of Left hand.

	R1	***R2***	***R3***	***R4***	***R5***
R1	67	5	10	10	4
R2	12	54	34	11	6
R3	9	26	55	40	9
R4	6	10	31	58	5
R5	5	8	8	9	46

TABLE II: Comparison of Matching accuracies across each finger of Right hand.

V. RECOMMENDATIONS FOR FUTURE WORK

Our project had fingerprint identification as its main focus from the start. The topic has been thoroughly researched in the scientific community and we wanted to apply those findings in a portable device. As the title suggests though, we would like to experiment with other biometric data as well such as vein detection [**?**] and retina scan [**?**], all of which though were out of the scope of this project mainly because of the given time frame.

In the future, the efficiency of other Machine Learning techniques could be examined for fingerprint matching. Also the possibility of using Deep Learning for fingerprint recognition could be tested, as the use of neural networks in smartphones has been tested before and it has proven to have potential [**?**].

VI. CONCLUSIONS

ACKNOWLEDGMENT

We would like to thank Johan Pouwelse for his insights and help that he provided us throughout the project.

REFERENCES

[1] M. Ramin, 'RxFingerprint', 2017. [Online]. Available: https://github.com/Mauin/RxFingerprint. [Accessed: 21-04-2017].

[2] Android 6.0 API, 2015. [Online]. Available: https://developer.android.com/about/versions/marshmallow/android-6.0.html [Accessed: 21-04-2017].

[3] OpenCv Answers' Forums, 2013. [Online]. Available: http://answers.opencv.org/question/6364/fingerprint-matching-in-mobile-devices-android-platform/ . [Accessed: 21-04-2017]

[4] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on (Vol. 2, pp. 1150-1157). Ieee.

[5] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395.

[6] Feichtinger, H. G., & Strohmer, T. (Eds.). (2012). Gabor analysis and algorithms: Theory and applications. Springer Science & Business Media.

[7] Oliveira, V. A., & Conci, A. (2009). Skin Detection using HSV color space. In H. Pedrini, & J. Marques de Carvalho, Workshops of Sibgrapi (pp. 1-2).

[8] Soille, P. (2013). Morphological image analysis: principles and applications. Springer Science & Business Media.

[9] Maio, Dario, and Davide Maltoni. "Direct gray-scale minutiae detection in fingerprints." IEEE transactions on pattern analysis and machine intelligence 19.1 (1997): 27-40.

[10] Gao, Q., Forster, P., Mobus, K. R., & Moschytz, G. S. (2001, May). Fingerprint recognition using CNNS: Fingerprint preprocessing. In Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on (Vol. 3, pp. 433-436). IEEE.

[11] Zhang, T. Y., & Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. Communications of the ACM, 27(3), 236-239.

[12] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2564-2571). IEEE.

[13] Noureldien Hussein, 2016 [Online]. Available: https://github.com/noureldien/FingerprintRecognition [Accessed: 08-03-2017]

[14] OpenCv. 2017. [Online] Available: http://opencv.org/ [Accessed: 10-03-2017]