DE GRUYTER OPEN

Journal of ELECTRICAL ENGINEERING, VOL 68 (2017), NO4, 256–266

Authentication based on gestures with smartphone in hand

Juraj Varga, Dominik Švanda, Marek Varchola, Pavol Zajac *

We propose a new method of authentication for smartphones and similar devices based on gestures made by user with the device itself. The main advantage of our method is that it combines subtle biometric properties of the gesture (something you are) with a secret information that can be freely chosen by the user (something you know). Our prototype implementation shows that the scheme is feasible in practice. Further development, testing and fine tuning of parameters is required for deployment in the real world.

K e y w o r d s: smartphone, accelerometer, gestures, gyroscope, access control

1 Introduction

Access control is one of the most difficult problems of computer security. Demands on security are often in conflict with the efficiency requirements and user convenience. This is especially pronounced on mobile devices, such as smartphones and tablets. These devices are frequently in a hostile environment, thus it is essential that they can be locked. On the other hand, a limited user interface and a frequent need to unlock the device make it important task to study authentication methods that improve the user experience but do not compromise the security.

Newer mobile devices integrate a range of special sensors that can be used to develop new methods of authentication that is not available on stationary PCs. Some of the examples include use of touch screen (pattern unlock), fingerprint reader, camera (biometric identification), near field antenna and barcode/QR scanners (proximity tokens). We summarize existing specialized authentication methods in more details and point out their advantages and disadvantages.

In our proposed method we focus on the use of an internal accelerometer (and potentially also a gyroscope). The main idea is simple: the users move the mobile device in a pre-specified pattern (a gesture). The gesture is recognized internally based on the readings of the accelerometer. We summarize various existing algorithms for gesture recognition. The captured gesture is evaluated with the algorithm. If it is within a specified tolerated distance from a specified pattern, user is authenticated and the device is unlocked.

The gesture based authentication combines two authentication factors: user chosen secret (something you know), with biometrics (something you are). The biometric part comes from the fact that it is difficult for two users to make the same gesture in the same way.

Another advantage is the physical difficulty of the brute-force attack as each try takes a relatively long time to execute. Moreover, the attacker most likely would need some non-trivial special physical equipment to move the smartphone about. Finally, the device can be configured to require additional authentication after some number of failed tries, similar to PINs.

We have tested our prototype implementation in a small series of experiments. Our main goal was to check whether the method can be used for reliable user authentication. In a successive experiment we have also checked how resilient the method is against shoulder-surfing attack. More details on experiments are presented in Section 6.

Based on the results of testing, we made improvements to the prototype, mostly in the performance area. We give details on these improvements and following present some experiments and comparison with the prototype version.

2 Overview of the access control methods on mobile devices

Access control mechanisms include identification, authentication, and successive authorization of the user. In the context of mobile devices, there is typically a single user, and single level of privilege. Thus, we will only focus on authentication phase. In this phase we suppose that the device is in the state of limited access (locked). The owner of the device uses some process to prove that he is indeed the authorized owner. Upon successful authentication, the user is granted full access to the applications and data within the device. We will simply say that the user unlocks the device.

There are various authentication methods in practice. These are usually based on some factors connected to the user:

Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovičova 3, 812 19 Bratislava, Slovakia, pavol.zajac@stuba.sk

DOI: 10.1515/jee-2017–0037, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

- something that user knows: passwords, PIN codes, or any other pre-selected secret information.
- something that user owns: cryptographic token generating one-time passwords (OTPs), proximity key;
- something that user is: biometric features connected directly to the user, such as fingerprint, retina scans, even user face.

In the following subsection we will briefly overview some of the existing authentication mechanisms relevant to our research. We focus on Android platform, which is open for research and development.

PIN Code

Personal identification number was used in the past as a basic prevention from unauthorized access. Currently it is used in credit cards, as a pass code on secure doors or for unlocking a SIM card. This code is formed by a sequence of numbers that can repeat, which for standard length of four digits means 10000 possible codes (see Fig. 1 first from the left). In Android OS it is possible to set a PIN code longer than four digits, which adds to overall security and hinders brute-force attacks, but lowers user convenience. Moreover, after five unsuccessful attempts the device is locked for 30 seconds, which also increases the brute-force cracking time.

Very important criterion for choosing a PIN code is for majority of users is fact, that it is easy to memorize. Majority of users therefore uses easily memorable combinations or dates [3]. This way attacker does not need to go through whole key space, but only very probable and easy combinations. This operation can be moreover automated and R2B2 [4] was able to search all 10000 possible combinations in less than 20 hours.

Alphanumeric Password

The second way of securing Android device is to use alphanumeric password (Fig. 1 second from the left). This way is currently one of the most secure means of protecting access not only for mobile devices, but also for e-mail clients, social networks etc. Standard password contains combination of letters, digits or special characters for enhanced security. Well chosen password of sufficient length containing all of these characters is secure and effectively mitigates brute-force attacks. In Android devices it is possible to insert four or more letter password to secure access to this device [2].

Using correct and long alphanumeric password is currently the best way of securing a mobile device against unauthorized access. However, secure password is very complex and therefore for quick unlocking of mobile device users use simpler dictionary passwords. These are easy to remember and quick to write on small keyboard. Unfortunately, this practice weakens this method, because security and quality of password is in trade-off with user convenience and unlocking time of device [5].

The authors of [6] investigated possibility of side-channel attacks on smart phones. They built a simple application that listened and captured data from the accelerometer when the user was prompted to enter a PIN code or password. The obtained data were used to reconstruct input from user - the device moves a little when the user makes some sort of input, and these small changes can reveal correct PIN code or password. The results of this sort of attack were very impressive, with more than 50% rate of success. This research also suggest that accelerometer is an important system resource that should have a special protection while the device is locked (including the authentication process itself). This on the other hand suggest that accelerometer readings can be used in authentication process itself.

Pattern Recognition

Android introduced a new way of securing a mobile device (Fig. 1 second from the right). Locking by a pattern is a combination of speed and security, which satisfies requirements of majority of users. There are nine points on the device screen, which the user connects by drawing a line, connecting at least four points. The user can choose his own pattern from 985 824 possible combinations of lengths four to nine. This equals to security of seven-digit numeric password, but this way is quicker and easier to remember. If the user avoids simple non-overlapping patterns, it is a very good alternative to above mentioned security measures, which blends security and short unlocking time [6].

Pattern locking is fast and simple, but it has its weaknesses. The main one is improperly chosen non-overlapping pattern. There are only 10096 of these patterns, which equals to security of four to five digit numeric password, which makes brute-force attack possible. Another disadvantage is possibility of seeing the finger movement and attacker can repeat this movement in a few tries. In the case the attacker did not see the movement, there is a so called Smudge Attack [7], which exploits the visible grease smudges on display left by the user finger. Therefore the users need to comply with best practices and choose correct and secure way to use this security measure.

Facial Recognition

On the newer Android devices possibility of unlocking device by a front camera was added (Fig. 1 first from the right). This method recognizes owner's face and unlocks the device. It is a very quick method providing at least basic security measures. In case of unsuccessful recognition of bio-metric elements there is still possibility to configure unlocking gesture or password. There were some improvements added to this method, mostly the need to move eyelids, which ensured that the person unlocking the device was in fact living user and not a static photography.

This method is from all mentioned methods of physical access the least secure. It is more of an accessory than a real security service. Original proposal was easy to trick using a photograph, even blinking is possible to bypass by switching of two photographs (one with closed eyes,

Fig. 1. Methods of Physical Access in Android OS

the other one with opened eyes). Very often system does not recognize the user and the overall effectiveness and performance of this method degrades. Moreover, many devices do not have the front camera, which renders this method useless on this kind of devices [8].

Fingerprint Recognition

Currently on the Google Play there are also many applications implementing this security measure. However, there is no published research paper on security of this measure, mostly because only some devices have a fingerprint scanner and only some versions of Android support this feature. Moreover, many users claim that some of these applications contain malicious code.

Summary of authentication methods

As we have seen, each method of authentication has its own advantages and disadvantages. Methods based on secret information are prone to a wrong trade-off between security and user convenience. Essentially, strong passwords are difficult to remember and to enter in the user interface. Cryptographic tokens provide strong security, but are costly and the user needs an extra hardware to take care off (which can be inconvenient in mobile setting). Biometric information is easy to use and can provide very strong authentication mechanism. However, its basic feature is unchangeability, which can lead to critical failure across various secured devices.

In security sensitive applications, multi-factor authentication is preferred. In banking applications, password is complemented by OTPs, or validated by biometric readings. In our proposal we combine properties of chosen secret (part of the authentication process is secret) with biometric properties (how the secret is proven is different for different persons).

3 Algorithms for gesture recognition

The main focus of our research is gesture recognition in 3D space. There exist many algorithms for this purpose. The important criterion is allowance of some inaccuracy, because repeating a gesture in 3D space in a completely same way is practically impossible. However, deviations must be in an allowed limit and tolerance for deviations cannot be too high, because it would negate effectiveness and security of chosen algorithm. Algorithms for speech processing are often used, because they have tolerance for error and despite that have very high rate of success. The most common algorithms for input processing and comparing to pattern are

Linear time warping An algorithm used to compare samples with the same length. Pattern and sample are compared and an Euclidean distance is calculated. Sample with the lowest distance is closest to the pattern [9].

Dynamic time warping This is an algorithm used to measure difference between samples which differentiate in time and speed. The algorithm looks for the best match and removes time dependences and speed of performing the gesture. This algorithm is often used in human speech recognition, where speed of word pronunciation may differ from original sample. Samples which differ in time and speed can be compared by quadratic Euclidean distance based on similar markers, even if there are non-linear differences between them [10].

Hidden Markov model

Model uses statistical modeling of processed signal and therefore it is possible to recognize dynamic and time-variable gestures. Markov process is stochastic process in which next state depends only on current state. This means, that whole process does not need to remember previous states [11].

The problem of gesture recognition in three dimensional space was already investigated before introduction of mobile devices. For example, it was intended to be a way of controlling smart devices (televisions, gaming consoles or intelligent houses). Some of these projects we based our research on are presented below:

XWand

Project XWand [12] tried to implement own hardware device containing movement sensors (magnetometer, gyroscope, accelerometer) and use this device to control other devices (intended for Microsoft XBox 360 gaming console). XWand is a wireless device in a shape of a stick

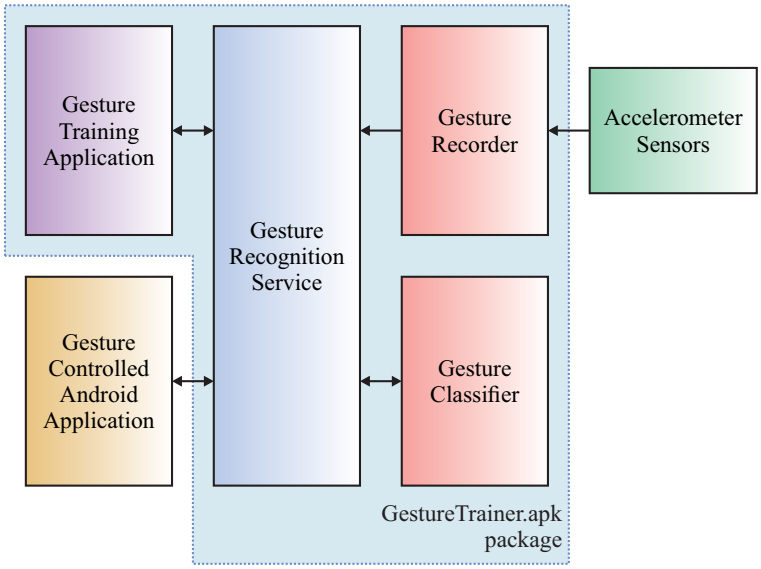

Fig. 2. Scheme of Android gesture recognition tool [15]

in which are the sensors located. Using movements in space are recognized gestures paired with specific actions. In this case the DTW and HMM algorithms used for gesture recognition and classification.

Inertial measurement framework for gesture recognition and applications

This one is a research conducted on MIT [13]. The atomic gestures were defined in this project. These cannot be further divided, and more complex gestures contain them. The main advantage is that it is enough to evaluate a small number of gestures and other gestures can be artificially created from them. In general, the gestures have only two atomic movements, straight line and movement back and forth. Using combination of these gestures it is possible to create complex ones. Movement on a straight line has two peak values, which are based on acceleration and following stopping of the hand. During movement back and forth comes to acceleration to the other direction and another following halt. Thus this movement can be described based on three peak values.

Android Gesture recognition tool

This is a tool capable of recognition of movement gestures using obtained data from accelerometer included in an Android device [14]. Gestures can be assigned to classes and evaluate distance of gestures from trained patterns. This tool contains a service, which recognizes new gestures and results are provided to management application. Therefore it is possible to determine, whether current gesture is in the list of trained ones, or how far it is from a trained one. The shorter the distance, the more is the gesture closer to the original pattern. It uses genetic time warp (GTW) algorithm for recognition, which mitigates shortcomings of DTW algorithm. GTW uses genetic principles to find shortest path in the matrix of values and moreover is better for classification of multidimensional signals - in this case, input from three axes of accelerometer.

As is shown in Fig. 2, this framework contains several separate modules, which can be further upgraded:

AccelerometerSensors: Hardware sensor, outputs current values of accelerometer axis X,Y and Z.

GestureRecorder: Constantly monitors accelerometer (movement and acceleration). A gesture is only detected if absolute value of acceleration surpasses defined threshold in a given time. This eliminates interfering movements, because gestures are recorded and recognized automatically. For our purposes we had to modify this module not to operate automatically, but only when prompted by holding a button.

GestureClassifier: Here is a DTW algorithm used to train and recognize gestures. It also contains logic for management of existing trained gestures.

GestureRecognition Service: A service obtainable by any application, runs as a process in background and reacts on recorded gestures from . It only depends on the specific mode, whether is the gesture chosen as a pattern or is compared to existing pattern. The result is then forwarded to listening application.

GestureTrainingApplication: An application for gesture management. Serves for both training and recognition of gestures.

This framework can communicate with any gesture-controlled application. Application designer does not need to make any special modifications and can let all settings on default values [15].

4 Authentication method based on gestures

In this section we describe our proposed method and its prototype implementation. The method can be summarized as follows:

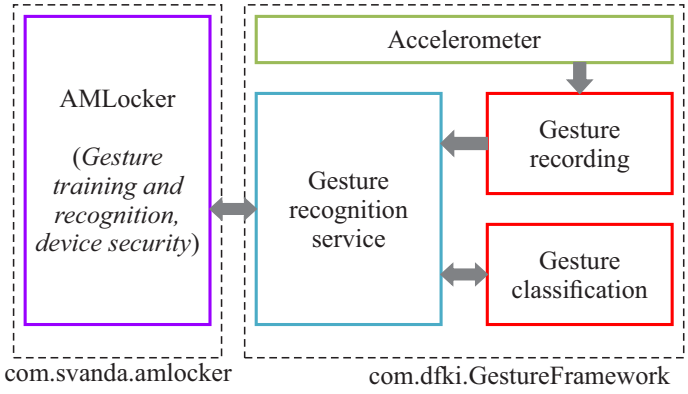

Fig. 3. Modified architecture of AMLocker

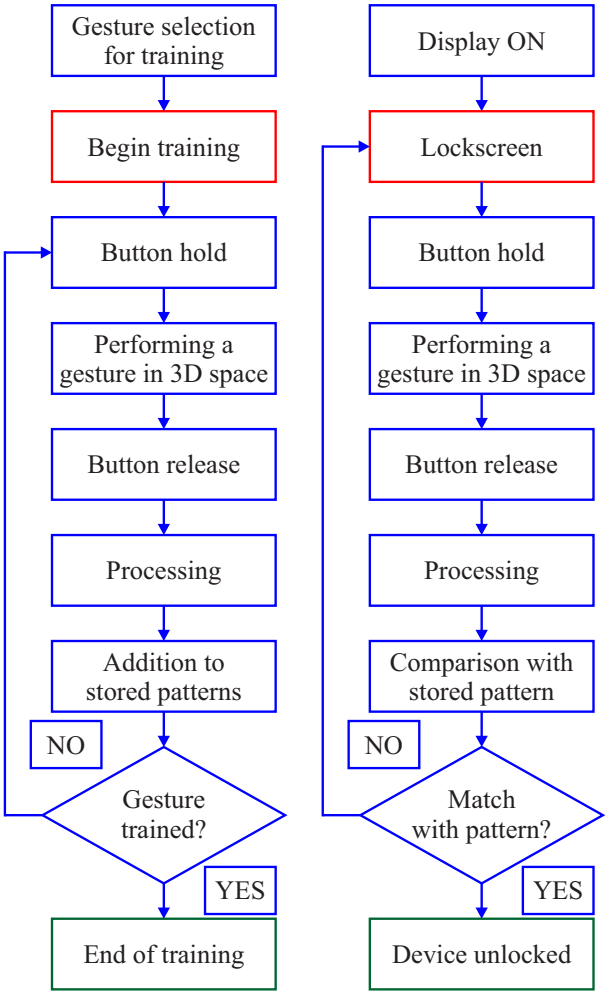

Fig. 4. Diagrams of training and unlocking process

- User chooses a secret gesture with the mobile phone in the 3D space. In the concrete implementation, the selection of the gesture can either be unrestricted, or limited to a subset of special gestures (*eg* , only movements along x-y-z axes) and their combinations.
- User records the gesture in training phase of the authentication algorithm.
- When attempting authentication, user repeats the gesture. If the gesture is recognized by the authentication algorithm, device is unlocked.
- User can change the gesture any time after the successful authentication to the device.

We recommend to augment the mechanism by combining it with strong backup password. Password entry should be required after a specified number of authentication failures.

Practical Requirements

The following criteria should provide authentication and security of physical access to mobile device running Android OS:

- Access restriction against non-authorized users.
- Ability to authenticate by unique device gesture in 3D space.
- Capturing a gesture in 3D space using accelerometer, including simple training and seamless recognition during unlocking.
- Ability to capture gestures in full motion range, providing sufficient complexity to obtain security for this proposition.
- (Optional) Ability to use gestures to start user-defined shortcuts - a few assigned applications for quick use.
- Implement everything in one user-friendly application with simple design.

After a thorough examination of official application market we did not find any application, that could be used for physical access protection. Applications closest to our specification ([16-18]) need to use accelerometer to work properly, but in fact they do not use full-fledged gestures. These actions were not complex enough to at least match current means of securing physical access to mobile devices.

Algorithm

In our work we decided to use an existing framework [15] for gesture recognition, rather than building a new one from the beginning. This framework along with

all evaluation algorithms was proven effective in gesture recognition and moreover, is native for Android OS. However, we needed to make changes in modules to satisfy our requirements. Therefore we built a new proof-of-concept application for gesture management and following use of these gestures for unlocking device running Android OS.

Proof-of-concept Application

We implemented proposed mechanism as a standard Android application written in Java language which requires no system permissions to run correctly. Application is an application in which we implemented all general requirements. Applications for training and recognition were merged into one module, as seen on Fig. 3. Gesture recording module was also changed - automatic recording requires that the user holds a button and then make a gesture with the device. This allows us to record gestures of variable time length, which adds to gesture variability and therefore to its security. Application consists of four basic parts:

Lockscreen: an activity which is shown when the power button is pressed. This customized lockscreen replaces the default one. Here the button used for unlocking is found, with ability to insert an override password in case the user forgets or cannot perform required gesture.

Gesture training activity: an activity used to train gestures for unlocking the device or to be set as application shortcuts. This screen also serves to manage gestures - deleting or reassigning.

Activity for choosing a default application launcher: in this lockscreen the user has to set our application as a default lockscreen. This has to be done, because Android does not allow to block home button in third-party applications and therefore gain access to device. When our lockscreen is chosen, user gains access only by performing correct gesture or entering correct password.

Activity for changing a master password: the last activity implements a system fail-safe. When the user forgets the gesture or cannot perform it, he can insert an override password to gain access to the device. This password can be configured on this screen.

As we mention above, user interaction during unlocking is reduced to holding the button and making a gesture, for best user convenience. The design of the application logic is summarized in Fig. 4.

5 Proof-of-concept testing and results

Implemented application is fully functional according to the design specifications. It does not require any permissions. In the proof-of-concept testing phase we measured these four aspects:

- Rate of Tolerance for Various Thresholds.
- Number of Training Sessions.
- Time Delay.
- Authentication by Unauthorized User.

These tests were conducted by four different test subjects. Each subject had different device for testing. On these devices there were three different versions of Android OS installed.

Rate of Tolerance for Various Thresholds

In the first testing session we measured influence of the rate of tolerance for error on the success in unlocking the device. If the user was perfect in performing a gesture, the distance from the pattern would be equal to zero. Since this is impossible in real-life conditions, some margin of error must be set. If the distance between recorded gesture and pattern is shorter than chosen threshold, the gesture is accepted as correct and access to the device is granted. We decided to test the success rate for threshold values of 5, 7, 10 and 15. During each test the users trained (five training sessions) an arbitrary gesture and tried to unlock their device 10 times (see Fig. 5: 1 for success, 0 for failure).

As we can see from the results, when the threshold value is too high (10 and 15), the users had almost 100% success rate in unlocking the device. With these values set, even the unauthorized user can gain access to this device. On the other hand, when this value is too low (5), even authorized users had problems accessing their device. Therefore we recommend to use threshold values from interval (6, 8). These values can be also set in the application and every user can customize them for his convenience.

Number of Training Sessions

In the second testing session we measured the influence of the number of the training sessions on gesture recognition performance. In the previous test the gesture was trained five times. In this test, after each attempt test subjects put the device down and wrote the value they obtained. This procedure would simulate real life usage, because nobody holds his device in hand all the time. Therefore the results would be worse than by performing unlocking continuously. The results are in the top part of Fig. 6.

Firstly, the test subjects trained the gestures 3 times. As we can see from the top part, the average distance of gestures from patterns are sufficient for some threshold values, but there is a very high variability in values, when the gesture is being processed and recognized. This can lead to lower user convenience, because the unlocking would take several attempts to succeed.

In the middle and bottom parts of the figure, there are values obtained with 5 and 10 training sessions, respectively. In the last column, there is a rate of how the average value improved against the previous test. After the second test (5 sessions) we recorded average improvement of 25%, in the third test the results improved again by 25%. In case the gestures had already high rate of success, this improvement was not that visible as with those with previously worse score. In the end, average values fell below threshold value of 5.

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Combined	Success (%)
1	0	1	1	0	0	0	1	1	0	0	4	40
2	0	1	1	1	1	1	0	0	0	0	5	50
3	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	1	0	0	1	1	4	40

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Combined	Success (%)
1	1	1	1	1	1	1	1	1	1	0	9	90
2	1	1	1	1	1	1	1	1	1	1	10	100
3	0	0	0	1	0	1	1	0	1	1	5	50
4	1	1	0	0	0	0	0	1	1	1	5	50

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Combined	Success (%)
1	1	1	1	1	1	1	1	1	1	1	10	100
2	1	1	1	1	1	1	1	1	1	1	10	100
3	1	1	1	1	0	1	1	0	1	1	8	80
4	1	1	1	1	1	1	1	1	1	1	10	100

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Combined	Success (%)
1	1	1	1	1	1	1	1	1	1	1	10	100
2	1	1	1	1	1	1	1	1	1	1	10	100
3	1	1	1	1	1	1	1	1	1	1	10	100
4	1	1	1	1	1	1	1	1	1	1	10	100

Fig. 5. Results from testing session 1

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Average
1	4.88	14.75	11.71	7.72	14.23	10.21	9.17	4.28	4.46	7.28	8.87
2	5.17	5.26	5.66	6.69	5.67	7.68	6.28	7.72	6.04	7.9	6.41
3	10.6	8	10.7	12.7	13.1	5.95	4.1	10.1	10.2	8.03	9.34
4	2.9	6.1	5.25	4	4.7	4.5	9.53	5.9	4.5	6.1	5.35

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Average	Improved (%)
1	6.22	4.81	4.44	6.48	6.42	6.97	4.98	4.02	5.19	7.97	5.75	35
2	6.69	4.91	2.53	4.22	3.41	4.67	7	5.61	6.37	5.6	5.10	20
3	7.92	7.20	8.05	6.45	12.5	5.42	5.72	12.2	5.65	5.66	7.68	18
4	5.75	3.24	2.42	3.32	3.90	3.56	3.37	4.41	5.46	3.93	3.94	26

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Average	Improved (%)
1	3.30	3.07	2.54	3.05	2.57	2.99	4.46	3.52	3.14	3.46	3.21	44
2	3.33	3.86	4.69	4.34	5.72	3.73	4.72	4.94	4.97	5.03	4.53	11
3	4.82	5.11	5.93	4.22	5.90	3.55	5.98	5.94	5.27	6.77	5.35	30
4	3.18	2.21	3.10	3.26	2.52	4.39	3.43	4.59	2.95	3.74	3.34	15

Fig. 6. Results from testing session 2

This test proves, that number of training session has a positive effect on how successful the gesture is recognized. 10 sessions showed very good properties, so this value will be used in the next set of tests.

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Average	Degradation (%)
1	5.90	4.22	4.06	5.83	6.38	5.18	9.35	4.59	4.44	5.36	5.36	40
2	6.43	5.59	5.68	4.96	5.35	6.10	5.69	5.20	4.80	4.29	5.41	16
3	5.02	5.6	6.12	11.0	6.10	11.1	4.75	5.09	4.94	5.41	6.42	17
4	4.20	4.20	7.00	4.60	8.50	4.20	4.46	4.53	4.40	4.28	5.04	34

Fig. 7. Results from testing session 3

Attempt / Subject	1	2	3	4	5	6	7	8	9	10	Average
1	10.30	12.68	8.90	11.70	11.02	7.90	10.8	10.8	6.5	10.3	10.09
2	16.48	12.08	13.82	16.46	17.02	16.49	14.80	15.03	16.88	10.17	14.92
3	8.27	7.24	6.30	7.20	10.92	7.46	7.40	7.90	7.00	8.00	7.77
4	14.0	9.70	14.30	10.70	14.30	8.44	8.24	9.08	9.83	10.3	10.89

Fig. 8. Results from testing session 4

Attempt / Tolerance	1	2	3	4	5	6	7	8	9	10	Total	Success (%)
10	1	1	1	1	1	1	1	1	1	1	10	100
7	1	1	1	1	1	1	1	1	1	1	10	100
6	0	1	1	1	1	1	1	1	0	1	8	80
5	0	1	1	1	1	1	0	1	1	1	8	80
4	0	1	0	0	1	1	0	1	0	1	5	50

Attempt / Tolerance	1	2	3	4	5	6	7	8	9	10	Total	Success (%)
10	0	0	0	1	0	0	0	0	0	0	1	10
7	0	0	0	0	0	0	0	1	0	0	1	10
6	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 9. Results from testing session 5

Time delay

For the real use testing we chose an approach where user trains a gesture and tries to unlock assigned device. Then he waits for two hours. After this time passes, he tries to unlock the testing device again, see Fig. 7.

As we expected, each testing subject performed worse in the second run. This happens because right after training subject remembers the gesture more clearly than after some time idle. The only way how to resolve this problem is to choose less complicated gesture (although subject 1 had visually the easiest gesture) and train to get it in hand.

Authentication by unauthorized user

Making a gesture with mobile phone in public can raise some attention. The user can easily attract unwanted attention from potential attacker. Therefore we decided to conduct the last test to simulate a situation, when malicious user sees authorized user making a gesture. Then he steals his device and tries to replicate this movement. This test expresses the practical security potential of our proposition, see Fig. 8.

By repeating a gesture in 3D space by unauthorized person the distance grew rapidly. But in some cases we came dangerously close to the values that could lead to security breach. The case of subject number 3 proves, that threshold values higher than 7 might not be as secure as we hoped. We have to remark, that during these tests the attackers saw the distance they made on the device screen. By using this side-channel they could improve their gestures to get better result and eventually breach this security mechanism. This is why we decided to do one last experiment, when this information would not

be possible to obtain. Therefore we get a real life attack scenario to test the practical security of our mechanism. The chosen gesture was trained 10 times by the user and then he attempted to unlock the device with thresholds 4, 5, 6, 7 and 10. The results are in the top part of Fig. 9.

After this was done, the attacker took the device and tried to replicate the gesture he saw (10 times with the same threshold values). The results are in the bottom part of Fig. 9. The gesture alone was rather simple, so it could be easily replicated. As we can see, all threshold values apart from 4 are acceptable for unlocking the device for the legitimate user. However, the attacker if the threshold values are too high (7 and 10), even attacker is able to unlock the device. The tests also show, that threshold values should based on gesture complexity. If the gesture is easy to perform, threshold values should be lower (5 or 6). If a more complicated gesture is used, it is possible to use values around 7 or 8.

6 AMLocker Version 2.0

We were not completely satisfied with the initial testing. The proof-of-concept application worked well. However, to provide sufficient authentication capability, while not compromising security a difficult fine tuning of parameters was required. Moreover, the gesture could only be recognized when the user was stationary. To improve the recognition rate, and the usability, we have decided to extend the application with the input from gyroscope, and to upgrade the internal recognition algorithms. In this section, we provide more details and testing results from AMLocker version 2.0.

Adding a Gyroscope

The main disadvantage in using accelerometer as a sole source of data, is that it is only usable in static state. This means, it provides different data when the user is standing still and walking, during performing the same gesture. To negate this flaw and enable dynamic device unlocking, we decided to include data from the gyroscope in the gesture recognition process. Since it can be used to track orientation and rotation, the overall gesture recognition will be less prone to noise resulting from dynamic actions like walking. This way we can easily reduce this noise and offer users greater flexibility and more convenience in using our application.

Neural Network for Gesture Recognition

However, including an additional set (triplets) of data - one for each axis of the gyroscope - meant that we had to abandon Android Gesture Recognition Tool as a primary tool for the gesture recognition. We found out, that this tool is capable to work with only one set of data, and therefore was unusable for this change. The closest option to this algorithm usable in our conditions was to use neural networks for gesture recognition. As with the previous algorithm, we decided to use already existing framework, rather than building one from the beginning. We decided to use Encog Machine Learning Framework [19] for gesture recognition. This framework provides various architectures of neural networks, support algorithms for pre-processing and normalization of data or other learning mechanisms like SVM, HMM, or genetic algorithms.

The internal components of the authentication method remains basically the same, with few additions:

- Sensors: Hardware sensors, output current values of accelerometer and gyroscope axes X,Y and Z.
- GestureRecorder: This component remained the same as in previous version.
- GestureTrainer and GestureClassifier: Here the DTW algorithm used to train and recognize gestures was replaced by a multi-layer perceptron neural network. Firstly, we have to train a gesture and only after that we can classify it.
- NeuralNetworkTrainer and GestureRecognition Service: Before gesture recognition service can be used to unlock device, the neural network responsible for this action needs to be trained to recognize gesture(s) chosen by the user. The trained neural network is then saved on the device and is used every time the device needs to be unlocked. Similarly to the DTW algorithm, neural network uses Euclidean distance to measure a distance of current gesture from the trained pattern.
- GestureTrainingApplication: An application for gesture management received some minor changes in user interface required by addition of neural network tools.

The neural network used in our Proof-of-Concept application has following properties:

- Network type: Feed-forward neural network.
- Network architecture: Multi-layer perceptron with variable number of neurons in each layer (user choice).
- Activation function: Elliot function, which is a faster approximation of sigmoidal function.
- Training method: Resilient Propagation, which is faster than standardly used Backpropagation method.
- Training strategy: Early Stopping Strategy is used to reduce the training time of the network.

Proof-of-Concept Testing and Results

Updated application is fully functional and has the same properties and requirements as the previous version. In the repeated proof-of-concept testing phase we measured three of four aspects as in the case of previous version:

- Number of Training Sessions.
- Rate of Tolerance for Various Thresholds.
- Authentication by Unauthorized User.

These tests were conducted by four different test subjects using two different devices. On these devices there were two different versions of Android OS installed.

Number of Training Sessions

As in the case of the prototype, we measured the influence of the number of the training sessions on gesture recognition performance. We chose three, five and ten training sessions respectively. After each session the users tried to unlock device ten times. Similarity with the trained gesture was set to 80%, or 0.8. The maximum number of epochs during training was set to 200 with stopping error of 0.001. In case of three training sessions the training took 10 to 48 seconds, in case of ten training sessions the training took 45 to 182 seconds. We remark, that training happens only once per gesture. The results are in Tab. 1. TS stands for number of training sessions.

Table 1. Results from testing Session 1, in %

Subject 1	Subject 2	Subject 3	Subject 4
60	60	50	40
100	100	80	60
100	90	90	80

Table 2. Results from testing Session 2, in %

Subject 1	Subject 2	Subject 3	Subject 4
70	70	50	50
100	100	80	70
100	100	90	100

Table 3. Results from testing Session 3, in %

Subject 1	Subject 2	Subject 3	Subject 4
0	0	10	0

Table 4. Results from comparison Session 1, in %

	Success Rate
AMLocker	80
AMLocker 2.0	100

Table 5. Results from comparison Session 2, in %

AMLocker	Sub-1	Sub-2	Sub-3	Sub-4
Thr 5	40	50	0	40
Thr 7	90	100	50	50
2.0	100	100	80	70

Sub = Subject

Firstly, the test subjects trained the gestures three times. As we can see from the first row, the success rate in unlocking devices is unsatisfactory and again leads to lower user convenience as in the case of the first prototype. In the next two lines we can see how the success rate improved when the number of training sessions increased. With just five training sessions we achieved 32.5% increase in success rate. The last setting with ten sessions was too time consuming with respect to the achieved increase in the success rate.

This test shows that number of training session has a positive effect on how successful the gesture is recognized, but after some limit, the increase is too small compared to the effort required. Five sessions seem to have the best training time/success rate ratio.

Rate of Tolerance for Various Thresholds

Initially, the threshold value for gesture similarity was set to 80%, or 0.8. In the next experiment, we investigated how the results with change with a more relaxed threshold of 0.7. Results are shown in Tab. 2.

As expected, when the threshold value is lower, the success rate of unlocking a device increases. However, when we compared the threshold values of 0.7 and 0.8, we found out that there was only a little increase in success rate. On the other hand, with lower threshold we were able to obtain almost 100% success rate in unlocking the device after 10 training sessions.

Authentication by Unauthorized User

In the last set of tests we investigated the resilience against unauthorized access. We set the number of training sessions to five and threshold value to 0.7 (based on previous experiments, these values provide best training time/success rate ratio). We allowed the attacker to watch test subjects to perform chosen gestures with their devices. Then he had to replicate these movements with the "stolen" device. Unlike the real world situation, we did not limit the time that the attacker was allowed to look on the gesture. Our results show, that the attacker has only a low chance to reproduce the gesture, as is shown in Tab. 3. To improve the security, threshold value can be increased, even if this can slightly inconvenience the user (by gesture retries).

Comparing Results with the Prototype

To compare the performance of the original prototype and improved version, we chose following criteria:

- Success rate of unlocking a device with recommended settings of the prototype.
- Success rate of unlocking a device with recommended settings for each version.
- Resilience against unauthorized access.

Success rate of unlocking a device with recommended settings from the first version - ten training sessions and threshold values of 6 and 0.6 respectively - is shown in the Tab. 4.

With five training sessions recommended in improved version, we again compared unlock success rate. In this

testing session we increased and decreased threshold values by one in prototype and used recommended value in version 2.0. The results are shown in the Tab. 5.

We can easily see, that improved version of our authentication method is more reliable even with lower threshold values than the originally proposed version.

Comparing resilience properties showed that we managed to improve this already impressive feature by a small margin in version 2.0. This means, that improved proposal is faster, more efficient and more secure that the original proposal.

Further improvements and fine tuning of parameters require a more extensive testing on a larger user base. Unfortunately, we do not have enough resources for such large scale experiments. However, we believe that our experiments have sufficiently demonstrated the feasibility and properties of the proposed method.

7 Conclusions

We have proposed a new authentication method suitable for mobile devices. The method is based on moving the device itself in a pre-specified secret pattern (gesture) by the user. This method combines secret chosen by the user (secret gesture pattern) with a biometric property, that it is difficult for another user to reliably reproduce the gesture, even if he learns the secret. The method is convenient for the user (gesture can be made a part of typical access action, such as taking mobile phone out of the pocket). Another advantage of the method is a difficulty of a realization of the brute-force attack, because each tried gesture need to be executed in a real world and time (if properly integrated within the system).

Our prototype implementation shows that the method is feasible, and can be implemented in existing Android mobile devices. On the other hand, existing gesture recognition algorithms are not as reliable as expected. Gesture recognition can be improved by adding input from gyroscope and by employing more advance recognition methods based on neural networks. A correct threshold for recognition must be specified in a way that will allow the device owner to authenticate with high probability while preventing the attacker even if he learned the secret gesture by observation. This threshold depends on concrete gesture recognition algorithm as well as the concrete sensors in each mobile device. Due to the stochastic nature of the method, we recommend to combine it with additional authentication method, which is applied after a limited number of failed attempts.

Acknowledgements

The authors would like to thank for the support of the grant VEGA 1/0173/13 and VEGA 1/0159/17.

References

[1] Android and iOS Continue to Dominate the Worldwide Smartphone Market with Android Shipments Just Shy of 800 Million 2013 According to IDC, [online]: http://www.idc.com/getdoc.jsp ?containerId=prUS24676414, 29.7.2014.

[2] Android Security Overview [online]: http://source.android.com/ tech/security/index.html, 29.7.2014.

[3] N. Aguilar, "How Thieves Unlock Passcodes on Stolen iPhones (And How to Protect Yourself Against it)", [online]: http://ios. wonderhowto. com/how-to/thieves-unlock-passcodes-stolen-iphones-and-protect-yourself-against-it-0139559/.

[4] J. Engler and P.Vines, "Electromechanical PIN Cracking Implementation and Practicality", iSEC Partners, Inc., Technical Report, 2013, pp. 6.

[5] M. Burnett 10,000 Top Passwords, [online]: https://xato.net/ passwords/ more-top-worst-passwords/more-269, 8.9.2014.

[6] A. J. Aviv, B. Sapp, M. Blaze and J. M. Smith, "Practicality of Accelerometer Side Channels on Smartphones", *ACSAC '12 Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 41-50.

[7] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze and J. M. Smith, "Smudge Attacks on Smartphone Touch Screens", *WOOT'10 Proceedings of the 4th USENIX conference on Offensive technologies Article no. 1-7*, 2010, pp. 10.

[8] E. Kremic and A. Subasi, "The Implementation of Face Security for Authentication Implemented on Mobile Phone", *Information Technology Interfaces (ITI)*, Proceedings of the ITI 2012 34th International Conference on, 2012, pp. 435-440.

[9] B. -K. Yi, H. V. Jagadish and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping", *ICDE '98 Proceedings of the Fourteenth International Conference on Data Engineering*, 1998, pp .201-208.

[10] S. Salvador and P. Chan, "FastDTW: Toward Accurate Dynamic Time Warping Linear Time and Space", *Intelligent Data Analysis*, vol. 11, no. 5, October 2007, 2007, pp. 561-580.

[11] O. Grosek, "On a Reconstruction of a Markov Chain", *Journal of Combinatorics*, Information & System Sciences 20 (1995), no. 1-4, 1995, pp. 85-93.

[12] D. Wilson and A. Wilson, "Gesture Recognition Using the Xwand, Assistive Intelligent Environments Group", *Robotics Institute*, Carnegie Mellon University and Microsoft Research, Technical report, 2004, pp. 10.

[13] A. Y, J. A. Benbasat and Paradiso, "An Inertial Measurement Framework for Gesture Recognition and Applications", *Gesture Workshop*, 2001, 2001, pp. 12.

[14] R. Nesselrath, *TaKG Ein Toolkit zur automatischen Klassifikation von Gesten*, Saarland University, Master Thesis, 2008, pp. 76.

[15] R. Nesselrath, "Android Gesture Recognition Tool", [online]: http://www. dfki. de/rnessel/tools/gesture_recognition/gesture _recognition. htm, 8.9.2014.

[16] Move'n Launch [online]: https://play.google.com/store/apps/ details?id=com.probayes. movenlaunch.full, 17.9.2016.

[17] SensorLock [online]: https://play.google.com/store/apps/ details?id=com.saturn. sensorpatterndemo&hl=sk, 17.9.2016.

[18] MagusFree 3D GestureLauncher [online] https://play.google.com /store/apps/details?id=com. stmp. magusfree 17.9.2016.

[19] Heaton Research: Encog Machine Learning Framework [online] http://www.heatonresearch.com/encog/, 17.9.2016.

Received 31 March 2017