University of Cambridge

Master's Thesis

User Authentication for Pico: When to unlock a security token

Author:
Cristian M. Toader

Supervisor:
Dr Frank Stajano

A thesis submitted in ful lment of the requirements for the degree of \Master of Philosophy"

in the

Computer Security Group
Computer Laboratory

June 2014

Declaration of Originality

I, Cristian Toader of Churchill College, being a candidate for the M.Phil. in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14943 (from page 1, excluding the appendix and bibliography).

Signed: _____

Date: _____

Abstract

The abstract needs to be written at the end.

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

Chapter 1

Introduction

Passwords are the most widely used electronic authentication mechanism. They are a secret sequence of characters used for proving the identity of the user, in order to gain access to a resource. This originally o ered a su ciently secure authentication mechanism. However, the fundamental concept of remembering a secret makes passwords unsuitable for the current technological context.

As Robert Morris [1] emphasises in his paper, there is a constant competition between attackers and security experts. The majority of users try to maximise the usability of passwords by choosing secrets that are easy to remember. This makes the mechanism more vulnerable to brute force attacks (e.g. dictionary, pre-compiled hashes, rainbow tables [2]). In the past, security experts were able to slow down attacks without any impact to the user. However, with a constant increase in computational power, passwords became easier to breach.

One aw of passwords is that when chosen freely they tend to be short and predictable. In order to maintain acceptable security guarantees, a number of creation requirements are enforced. Examples include having a minimum length, one or more numeric characters, and one or more special characters. Security experts recommend that each account needs to have an unique password. Furthermore, passwords sometimes require to be changed periodically with something not too similar with the original.

As shown by Yan et al [3], users choose weak passwords if not given any advice to make them memorable. Theoretically, additional restrictions make the mechanism secure. Having non-intuitive passwords that fully utilise the available character set makes brute

force attacks harder to perform. In practice however, users need to memorise numerous, unique, and complex passwords. As shown by Adams & Sasse [4] maintaining all restrictions and security advices proves not to be feasible. This often leads to poor practices such as writing the passwords on paper.

The main problem with passwords is the basic principle of users remembering a secret. If the secret is memorable, then an attacker may brute force it with more ease. If it is too complex, then the user may not remember it. Furthermore, since reusing passwords is not safe, and given the memory capacity people have, the mechanism is also not scalable. For these reasons, passwords prove not to be a reliable solution for the future and even present.

A large number of alternatives to passwords are available. However, as shown by Bonneau et al [5], the main advantage passwords have over other authentication mechanisms are in terms of deployability and usability. A study by Clarke et al [6] shows that although 81% of users agree to an alternative to password based phone unlocking, the majority ignores the existence of available solutions. The main conclusion is that although passwords are not secure, the cost of replacing them and familiarising with a new authentication mechanism is still too inconvenient.

The Pico project was designed by Frank Stajano [7] with the purpose of replacing password based mechanisms. Pico is a hardware token that generates and manages user authentication credentials. It has an additional layer of security by only being usable in the presence of its owner. Therefore, a security chain is created where \who you are" unlocks \a secret you have" which is used for authentication.

The current solution for unlocking Pico is by communicating with small auxiliary devices called Picosiblings [8]. They are designed to be embedded in everyday items that users can carry throughout the day (e.g. keys, necklace, rings). Each Picosibling transmits a secret sequence to Pico. When all required secrets are gathered, Pico becomes unlocked and can be used by its owner.

Picosiblings are a sensible solution to unlocking Pico. However, they are purely based on proximity to the device. As presented in the original Pico paper [7] anyone in possession of both Pico and its Picosiblings can have full access to the owner's accounts for a limited amount of time. This risk is lowered by additional security features. However, the main

vulnerability of Picosiblings is that they do not re ect who the user is, but additional things the user has.

The purpose of this dissertation is to design and prototype a better token unlocking mechanism for Pico. According to its design, the process should be memoryless, and enable continuous authentication. The token should lock and unlock automatically only in the presence of its owner. The solutions that seem to best t these requirements are biometric authentication mechanisms. Therefore, we have explored the possibility of combining multiple biometrics and behavioural analysis as part of an uni ed solution. The output from each mechanism is combined to generate an overall con dence level, re ecting that the owner is still in possession of the Pico.

A number of contributions have been made throughout this dissertation project. The following list presents a summary of these achievements, with further details in the following chapters.

- We have created a framework derived from the work by Bonneau et al [5]. This is used to evaluate a couple of existing token unlocking mechanisms, including Picosiblings. The results are used as a benchmark when evaluating the proposed solution.
- We have designed a new token unlocking mechanism. The solution may be used in any type of user authentication, but it is presented in the context of unlocking the Pico token.
- We have developed an Android prototype. The purpose of the implementation is to prove that the design can be developed using existing hardware. The prototype was not created for performance purposes. However, power analysis as well as timings of di erent authentication stages were recorded. These should serve as an approximation of the limitations and downsides of the scheme.
- The scheme is analysed by using the token unlocking evaluation framework. A comparison is made with Picosiblings in order to identify performance di erences. We aimed for the proposed scheme to achieve better results in at least some categories of the token unlocking framework.

- We have analysed and determined the impact of the proposed token unlocking mechanism on the Pico. The analysis is performed using the framework by Bonneau et al [5]. One of the goals when designing the solution was to make Pico better in terms of at least one property.

Chapter 2

Pico: no more passwords!

This dissertation project aims to design and implement a new Pico unlocking mechanism. A better understanding of the Pico design [7] is therefore necessary. This chapter aims to go into brief detail as to what Pico is, how it works, and what are its properties.

Pico is an user authentication hardware token, designed with the purpose of fully replacing passwords. Although other alternative mechanisms exist, they are generally focused on web based authentication. The solution described by Stajano addresses all instances of password authentication, both web based as well as local.

The motivation behind Pico is the fact that passwords are no longer viable in the current technological context. Computing power has grown, making simple passwords easy to break. Longer and more complex passwords are now required. However, as shown by Adams & Sasse [4], this has a negative effect on the users, which have a limited memorising capability.

Another reason why passwords are no longer viable is that they are not a scalable solution. Security experts recommend that passwords should not be reused for multiple accounts. However, a large number of computer based services require this type of authentication. In order to respect security recommendations, users would be forced to remember dozens of unique, complex passwords. A study by Florencio et al [9] confirms the negative impact of scalability on password quality.

When designing Pico, Stajano decides to have a fresh start. He describes that an alternative for passwords needs to be at least memoryless and scalable, without compromising

security. In the case of token based authentication, the solution also needs to be loss and theft resistant. Pico was therefore designed to satisfy these fundamental properties. It provides a number of additional bene ts also described in the work by Bonneau et al [5].

As a token authentication mechanism, Pico transforms \something you know" into \something you have". It o ers support for thousands of credentials that are kept encrypted on the Pico device. The encryption key is also known as the \Pico Master Key". If the Pico is not in the possession of its owner it becomes locked. In this state, the \Pico Master Key" is unavailable and the user cannot authenticate to any app[1].

Credentials are generated and managed automatically whenever the owner interacts with an app. Therefore, the responsibility of generating a strong and unique credential, as well as memorising it, is shifted from the user to the Pico. No additional e ort such as searching or typing is required.

Another important feature o ered by Pico is continuous authentication. Traditional password mechanisms provide authentication for an entire session. The user is responsible of managing and closing the session when it is no longer needed. Instead, Pico o ers the possibility of periodic re-authentication of its owner using short range radio communication. If either the Pico or the owner are no longer present, the authentication session is closed.

From a physical perspective, Pico is a small portable dedicated device. Its owner should be carrying it at all times, just as they would with a car key. It contains the following hardware components:

- Main button used for authenticating the owner to the app. This is the equivalent of typing the password.
- Pairing button used for registering a new account with an app.
- Small display used for noti cations.
- Short range bidirectional radio interface used as a primary communication channel with the app.

[1] For the purpose of brevity, any mechanism requiring user authentication will be called an \app". This naming convention was used in the original paper by Stajano.

- Camera used for receiving additional data from the app via 2-dimensional visual codes. This serves as a secondary communication channel.

As mentioned before, the Pico main memory is encrypted using the \Pico Master Key". The token contains thousands of slots used for storing unique credentials used in during authentication. Each credential consists of public-private key information generated during account creation in a key exchange protocol. The public key belongs to the corresponding app, while the private key was generated when registering the account.

During account creation Pico scans a 2D visual code generated by the app. The image encodes a hash of the app's certi cate that includes its name and public key. Pico starts the registration protocol through the radio channel, and the app responds with a public key used for communication. The key is validated using the hash from the visual code, and the protocol continues. Pico then initiates a challenge for the app to prove that it is in possession of the corresponding private key. It also provides to the app a temporary public key used for communication. Once the app is authenticated, Pico generates a key pair and sends the account public key to the app. To complete the registration it stores the generated private key and the app's public key.

The account authentication process starts when the user presses the main button and scans the app 2D code. The hash of the app's name and public key are extracted from the 2D image. This information is used to nd the corresponding credentials. An ephemeral public key encrypted with the app's public key is sent via the radio channel. The app is authenticated by using this key to require the corresponding (user id, credential) pair. Only after the app is authenticated Pico uses the public key generated during the registration process and authenticates itself to the app.

The locking process is an important aspect of Pico that was not yet fully described. Currently this is achieved by using bidirectional radio communication with small devices called Picosiblings [8]. These are meant to be embedded in everyday items that the owner carries around, such as earrings, keys, chains, and rings.

The Pico authentication credentials are encrypted using the \Pico Master Key". The key is not available on the token and can only be reconstructed using k-out-of-n secret sharing, as described by Shamir [10]. Except for two shares which will be discussed later, each k-out-of-n secret is held by a Picosibling.

Using an initialisation process based on the \resurrecting duckling" [11], each Picosibling is securely paired with Pico. The token sends periodic ping requests to which all initialised Picosiblings are expected to respond. During a successful ping, each Picosibling sends its k-out-of-n share to Pico. Given enough secrets, the \Pico Master Key" is reconstructed and Pico becomes unlocked.

Internally, Pico keeps a slot array for each paired Picosibling. Each slot contains a countdown value, and the key share provided by the Picosibling. When the countdown value expires, the share becomes deleted. Similarly, if k shares are not acquired before a prede ned time-out period, all shares are removed.

Except for the Picosiblings, two additional special shares with a larger time-out period are described by the paper:

- Biometric measurement used for authenticating the owner to Pico.
- A server network connection used for locking Pico remotely.

The possibility of using a smart phone as a Pico is brie y considered in the paper. This would have the advantage of not requiring any additional devices from the user. Modern smart phones provide all the necessary hardware required by Pico. However, this would be a security trade-o in exchange for usability. Mobile phones are an ecosystem for malware, and they present uncertainty regarding the privacy of encrypted data. This option may still be used as a cheap alternative to prototype and test.

Chapter 3

Assessment framework

The purpose of this chapter is to create an assessment framework for token unlocking mechanisms. This will be used to evaluate the Picosiblings scheme used by Pico. The results serve as a benchmark when comparing to the alternative mechanism proposed in this dissertation.

3.1 UDS assessment framework

Similar work to what we are trying to achieve in this chapter was performed by Bonneau et al [5]. The authors create a framework for evaluating web based authentication mechanisms. However, this is not entirely compatible for token unlocking schemes. Properties such as \Browser-compatible" do not apply, while others need to be rede ned to t our context. However, this paper is a good starting point for our token unlocking evaluation framework.

The motivation behind this paper is to gain insight to the di culty of replacing passwords. An assessment framework is created, and a number of web authentication mechanisms are evaluated. It is a useful tool in identifying key properties of web based authentication schemes. The framework is intended to provide a benchmark for future proposals.

The framework consists of 25 properties divided into three categories: usability, deployability, and security. Therefore, it is abbreviated by the authors as the \UDS framework",

and it will be referred to as such throughout this dissertation. An authentication scheme is evaluated by assessing whether each property is o ered. If a scheme almost o ers a property, it is marked as quasi-o ered. To simplify the evaluation, properties that are not applicable are marked as o ered.

After evaluating 35 schemes, the conclusion is that passwords are not completely dominated by any mechanism. They satisfy all the properties in the deployability category. They score reasonably well in terms of usability, excelling in properties such as: \nothing-to-carry", \e cient-to-use", and \easy-recovery-from-loss". However, from a security perspective passwords don't perform well. They only o er the \resilience-to-theft[1]", \no-trusted-third-party", \requiring-explicit-consent", and \unlinkable" properties.

Biometric mechanisms receive mixed scores on usability. None of them o er \infrequent-errors" due to false negative precision. They score poorly in deployability, partially because they often require additional hardware. In terms of security they perform worse than passwords. Replay attacks can be performed using pre-recording data of the user, making them not \resilient-to-targeted-impersonation" and not \resilient-to-theft". There is a one to one correlation between the owner and their biometric recording, and therefore they are not \unlinkable".

By analysing the framework results, we see that mechanisms such as security tokens o er \memory-e ortless" in exchange for \nothing-to-carry". The only schemes that o er both are biometric mechanisms. This is a consequence of replacing \something you know" with \something you are" instead of have. For di erent reasons no mechanism o ers both \memory-e ortless" and \resilient-to-theft".

When computing an aggregate score using the UDS framework, properties should have di erent weights depending on the purpose of the assessment. For example, when searching for the most secure authentication mechanism, security properties would have a larger weight in the overall evaluation. For this reason, the authors only provide the means for others to make an evaluation based on their needs. No aggregate scores or rankings are provided in the paper.

The authors mention the option of combining mechanisms as part of a two factor authentication. In terms of deployability and usability, the overall scheme o ers a property if it

[1] Not applicable to passwords

is o ered by both authentication mechanisms. In terms of security, only one of the two mechanisms needs to o er the property in order for the two factor combination to o er it as well. However, Wimberly & Liebrock [12] observe that combining passwords with a second authentication mechanism scheme leads to weaker credentials and implicitly less security.

3.2 Token unlocking framework

Token unlocking and web based authentication mechanisms are similar in concept. The di erence between the two is that on a token data is collected and processed locally. Therefore, a subset of the UDS framework is included in the token unlocking framework we have developed. Properties that do not apply, or would apply to every mechanism, were removed. Other properties needed to be adapted to the context of a token, and therefore have a di erent meaning.

The following list contains the subset of the UDS framework developed by Bonneau et al [5] that is relevant to token unlocking mechanisms. A short description is included to show how they are adapted to the new context.

Memorywise-effortless

Users do not need to remember any type of secret (e.g. passwords, physical signatures, drawings). The original property was quasi-o ered if one secret would be used with multiple accounts, but this will not apply for security tokens. As an example the RSA SecurID [2] is used in conjunction with a password in order to authenticate the user, and therefore does not o er this property.

Nothing-to-carry

The unlocking mechanism does not require any additional hardware except for the token. The property is quasi-satis ed in the case of hardware the user would have carried on a normal basis such as a mobile phone. An example mechanism that quasi-o ers the property is Picosiblings, which uses small devices embedded in everyday items. Biometric mechanisms that require additional sensors such as a ngerprint reader do not satisfy this property.

[2] http://www.emc.com/domains/rsa/index.htm?id=1156

Easy-to-learn

Users that want to use the unlocking mechanism would be able to learn it with ease. For example, Picosiblings do not o er this property because of the complexity of their management[3]. However, PINs or passwords do, as users are already familiar with this type of authentication.

Efficient-to-use

The amount of time the user needs to wait for the token to be unlocked is reasonably short. This includes the time required to provide authentication input, and set up the unlocking mechanism. As an example, the input and processing time for PINs is very low, and therefore the scheme o ers the property. However, mechanisms based on biometrics may not, depending on the implementation.

Infrequent-errors

The rightful owner should generally be able to successfully unlock the token. Any delays resulted from the scheme (e.g. typos during typing, biometric false negatives) contribute to the mechanism's inability to o er this property. For example, PINs have a limited input length and character set size. This makes frequent errors unlikely and therefore they o er the property. Biometric mechanisms, depending on the type and implementation may quasi-o er the property, but they generally do not.

Easy-recovery-from-loss

The meaning of this property was modi ed to re ect the context of token unlocking. It is o ered if the user may easily recover from the loss of authentication credentials. This may include the loss of auxiliary devices, forgotten credentials, di erence in biometric features. For example, forgotten PINs o er the property as they generally require a simple reset using an online service.

Accessible

The mechanism is usable regardless of any user disability or physical condition. As an example, passwords o er this property, while gait recognition unlocking does not.

[3] As discussed in the previous chapter 2, each Picosibling contains a k-out-of-n secret used to reconstruct the \Pico Master Key". The owner needs to choose the right combination of Picosiblings in order to unlock the Pico, which may prove di cult.

Negligible-cost-per-user

The total cost per user of using the scheme, enquired by both the user and the veri er, is negligible.

Mature

A large number of users have successfully used the scheme. Any participation not involving its creators, such as open source projects, that use or extend the mechanism contribute to this property. For example, passwords are widely used and implemented and therefore o er the property.

Non-proprietary

Anyone can implement the token unlocking scheme without having to make any payment, such as royalties. The technologies involved in the scheme are publicly known and do not rely on any secrets.

Resilient-to-physical-observation

An attacker is not able to impersonate the token owner after observing them authenticate. Based on the number of observations required for the attacker to unlock the token, the scheme may quasi-o er the property. The original paper suggests 10-20 times to be su cient. Physical observation attacks include shoulder sur ng, video cameras, keystroke sound analysis, and thermal imaging of the PIN pad.

Resilient-to-targeted-impersonation

An attacker is not able to impersonate the token owner by exploiting knowledge of personal details (i.e. birthday, full name, family details, and other sensitive information). In the case of biometrics, this property is satis ed if the scheme is resilient to replay attacks based on pre-recordings.

Resilient-to-throttled-guessing

The scheme is resilient to attacks with a guessing rate restricted by the mechanism. The process cannot be automated due to the lack of physical access to authentication data. This may be achieved using tamper resistant memory. For example, PINs o er this property as tokens such as SIM cards become locked after only three unsuccessful attempts.

Resilient-to-unthrottled-guessing

The scheme is resilient to attacks with a guessing rate unrestricted by the mechanism. The UDS framework suggests that if the attacker may process 2^{40} to 2^{64} guesses per account, they would only be able to compromise less than 1% of accounts. Since tokens are generally designed to have one owner, the property needs to be adapted. It is o ered only if an attacker requires more than 2^{40} attempts to unlock the token.

Resilient-to-theft

This property is relevant to schemes that use additional hardware other than the token. If the additional devices become in the possession of an attacker, they are not su cient to unlock the token. For example, auxiliary biometric devices used in the con]TJ 0 -20.324 Td [(counat)-343(n)-315(2)]TJ/F23 7.94o u(e)-1o-3154oqo c8(y)-743

should be closed. The concept is mentioned by Bonneau et al [5], but not included in the UDS framework. It is discussed in more detail by Stajano [7] as one of the bene ts of the Pico project. Using the UDS classi cation of the original framework, the property belongs to the Security category.

Multi-level-unlocking

The unlocking scheme provides quanti able feedback, not just a locked or unlocked state. It o ers support for multiple token security permission levels. These are granted based on the con dence that the user of the token is its owner. For example, a 70% con dence level that the owner is present may allow the user to access an email account, but not make payments or banking transactions. Passwords only provide a \yes" or \no" answer and therefore do not o er this property. Biometric mechanisms can o er this property. Their output is either a quanti able probability or a distance metric that data was collected from the owner. Di erent con dence levels could therefore enable di erent security permissions. Using the UDS classi cation of the original framework, the property belongs to the Security category.

Non-disclosability

The owner may not disclose authentication details neither intentionally or unintentionally. This is a broader version of the \resilient-to-phishing" and \Resilient-to-physical-observation" properties from the original UDS framework. The focus of this property is that the token may only be used by its owner. This is important in enterprise situations where the security token should not be shared. Passwords and other schemes based on secrets do not o er the property, as the owner can share it with another user with no di culty. Biometric mechanisms however are harder to disclose. Based on the UDS classi cation, the property belongs to the Security category.

Availability

The owner has the ability of using the scheme regardless of external factors. The ability to authenticate should not be impaired by the authentication context (i.e. tra c noise, di erent light intensities, restricted movement space, etc.). The property is not related to physical disabilities preventing the user from using the scheme but only on contextual in uences on data collection. For example, gait recognition

would only function while moving on foot and therefore does not o er the property. However, a mechanism requiring a PIN hand would work in any circumstance. Using the UDS classi cation of the original framework, the property belongs to the Usability category.

3.3 Picosiblings evaluation

We continue by evaluating the Picosiblings scheme using the token unlocking assessment framework. Results in this section will be used for comparison with the scheme designed in this dissertation project.

The scheme doesn't require remembering any secret and is therefore \memorywise-e ortless". Since it relies on devices embedded in everyday items, it quasi-o ers \nothing-to-carry". Bonneau et al [5] evaluate Pico as not \easy-to-learn" due to the Picosiblings unlocking mechanism. In the lack of user studies it only quasi-o ers \e cient-to-use" and \infrequent-errors". It does not o er the \easy-recovery-from-loss" property. The unlocking mechanism is invariable to external factors, therefore o ering the \availability" property.

The original paper marks Pico as not \accessible" due to the coordinated use of camera, display, and buttons. However, Picosiblings are \accessible" because they are embedded in everyday accessories that any user can wear. Pico doesn't aim to satisfy the \negligible-cost-per-user" property, and in the lack of a realistic Picosiblings cost estimate we will consider the property is not o ered. The scheme is at the stage of a prototype, with no external open source contributions, and little user testing. Therefore, it is not considered to be \mature". Frank Stajano [7] states that the only requirement for implementing the Pico design is to cite the paper, which makes the unlocking mechanism \non-proprietary".

Since the scheme does not rely on user input, it is \resilient-to-physical-observations". Based on the Picosiblings description given by Stajano [7] the scheme o ers \resilient-to-targeted-impersonation", \resilience-to-throttled-guessing", and \resilient-to-unthrottled-guessing". Any attacker can unlock Pico if they are also in possession its Picosiblings.

However due to the auxiliary shared secrets [5] the scheme quasi-o ers \resilient-to-theft". Each Picosibling only works with one veri er (i.e. its master Pico), and therefore is \unlinkable". The scheme was designed to provide \continuous-authentication". Because of the k-out-of-n master key reconstruction mechanism, Picosiblings can only have the locked and unlocked states. Therefore they do not o er \multi-level-unlocking". The scheme does not satisfy the \non-disclosability" property, because the owner is free to give away the Pico with its Picosiblings.

3.4 Conclusions

We have developed a token unlocking evaluation framework. The result is strongly related to similar work by Bonneau et al [5] which was summarised at the beginning of the chapter. Some properties needed to be adapted to t the context of a security token. In addition we have contributed with 4 original properties.

The Picosiblings scheme was evaluated using the token unlocking framework. This will serve as a benchmark when comparing to the new solution proposed in this dissertation. Two additional example mechanisms were assessed, with details in appendix B. A summary of the results is posted in table 3.1. Each property is highlighted with an appropriate colour in order to allow for quicker analysis.

As the table shows, none of the evaluated schemes completely dominates the others. They receive mixed scores in terms of availability and security. PINs dominate in terms of deployability, receiving a perfect score due to their similarity with passwords.

[5] Picosiblings also relies on two special shares. One is unlocked using biometric authentication, and the other is provided by an external server. Using these shares would only grant the thief a limited time window before the token is either locked remotely or the shares expire.

Property	PIN	Picosiblings	Face recognition
Memorywise-e ortless	Not-o ered	O ered	O ered
Nothing-to-carry	O ered	Quasi-o ered	O ered
Easy-to-learn	O ered	Not-o ered	O ered
E cient-to-use	O ered	Quasi-o ered	O ered
Infrequent-errors	Quasi-o ered	Quasi-o ered	Not-o ered
Easy-recovery-from-loss	O ered	Not-o ered	O ered
Availability	O ered	O ered	Not-o ered
Accessible	O ered	O ered	O ered
Negligible-cost-per-user	O ered	Not-o ered	O ered
Mature	O ered	Not-o ered	Quasi-o ered
Non-proprietary	O ered	O ered	Not-o ered
Resilient-to-physical-observations	Not-o ered	O ered	O ered
Resilient-to-targeted-impersonation	Quasi-o ered	O ered	Not-o ered
Resilient-to-throttled-guessing	O ered	O ered	O ered
Resilient-to-unthrottled-guessing	O ered	O ered	O ered
Resilient-to-theft	O ered	Quasi-o ered	O ered
Unlinkable	O ered	O ered	Not-o ered
Continuous-authentication	Not-o ered	O ered	Not-o ered
Multi-level-unlocking	Not-o ered	Not-o ered	Not-o ered
Non-disclosability	Not-o ered	Not-o ered	Quasi-o ered

TABLE 3.1: Initial assessment using the token unlocking framework.

Chapter 4

Design

4.1 Design requirements

The framework evaluation of Picosiblings provides insight as to how the scheme can be improved. We identify as a key downside that it does not guarantee the identity of the owner. This information is mainly inferred from the number of Picosibling shares in the proximity of the Pico. However, anyone may be in possession of the shares, therefore being temporarily granted full authentication privileges. This is re ected in the evaluation by failing to fully o er `resilient-to-theft" and \non-disclosability". Another improvement can be made by introducing \multi-level-unlocking", allowing for multiple levels of authentication depending on the con dence in the owner's presence.

The Pico design proposed by Stajano [7] claims two properties that need to be supported by the token unlocking mechanism: memory e ortless authentication, and continuous authentication [1].

A requirement when designing the new Pico unlocking mechanism is to fully satisfy the the properties presented in this section.

[1] Continuous authentication is de ned by the ability to re-authenticate the user without the need for any physical e ort.

4.2 Proposed solution

The idea explored in this dissertation is to simultaneously use multiple memoryless continuous authentication mechanisms. Each mechanism needs to provide a quanti able con dence level that will be used in calculating a combined score. This satis es the Pico design requirements. By combining mechanisms we achieve a higher con dence of correctly identifying the owner. Furthermore, given that each individual mechanism supports continuous authentication, using them simultaneously does not create any inconvenience for the owner.

Multi-level unlocking model

The Pico token should no longer enter a general locked or unlocked state. Its most important secret, the \Pico Master Key" should be kept in tamper resistant memory, and be accessible at all times. Using the overall score computed by the proposed mechanism, Pico should o er granular user authentication. Each app needs to be associated with a con dence level de ned during the registration process. If the overall con dence of the token exceeds the app's con dence level, then it becomes \unlocked" for that speci c app. All authentication sessions between Pico and apps need to be managed independently based on this model.

The scheme should achieve continuous authentication, while correctly identifying the owner of the token. Therefore, we have decided that authentication mechanisms combined in the scheme need to be based either on biometrics or behavioural analysis. Biometric features that can be used include iris, face, voice, and gait. Behavioural sources of data can be obtained from frequent GPS location, travel paths, wireless network connections, and others.

The solution o ered in this project is di erent from simply stating that Pico is using biometric data as an unlocking mechanism. The novelty in the design is based on how data is combined in order to compute the overall con dence level.

Decaying weights

Each mechanism of the scheme is assigned a prede ned initial weight based on the level of trust it o ers in identifying the owner. This doesn't necessarily need to be related to the precision of the mechanism, but it would be a good indicator for choosing the value.

Data samples captured for authentication are not always meaningful. For example, accelerometer values for gait recognition are only usable when the user is walking. Depending on how the sensors are integrated with the Pico, camera input for face recognition may not always capture a valid image. The con dence of each mechanism should therefore decrease in time from the last valid authentication sample. This introduces another original feature of this scheme, which is having a decaying weight. Each mechanism starts with a prede ned initial value that is decreases in time until a valid user data sample is recorded.

Let us take for example a voice recognition mechanism which samples data every minute. The current weight of the mechanism is 0 so its output is completely ignored. The next sample is recorded, and the voice recognition mechanism outputs a con dence of 70% that the owner is present. After the successful recording, the mechanism weight is updated to its prede ned starting value of 30. For the next 10 minutes the owner will be silently reading a book. Since the mechanism only identi es background noise, the weight value of 30 decreases in time. This will induce a smaller impact of the mechanism on the overall score. Each mechanism weight can decrease down to 0, at which point the mechanism is ignored. Computing the overall score will be explained in more detail later in the chapter.

4.2.1 Explicit authentication

We need to consider the case where the owner wants to use Pico to authenticate to a high security app, given a low con dence level from the authenticator. As an example, the Pico owner wants to access their bank account after sitting silent in a dark room for the past hour. Let us say the app requires a con dence level of 95%. Due to the lack of valid authentication data, the authenticator only outputs a 20% overall con dence that the owner is present. To solve this problem we have introduced the concept of explicit authentication mechanisms. When the con dence score drops below

the threshold required by an app, the user is given the chance to provide valid data samples to one or more mechanisms through an explicit request.

Combining explicit and continuous authentication can be performed consistently with the current design. Whenever explicit authentication is required, the only di erence is that the owner becomes aware of the authentication process. Given that prior to the explicit authentication request the unlocking mechanism didn't produce a high enough con dence, it is assumed that this will also happen prior to that. Therefore, explicit authentication requests need to have a slower decay rate. This will enable the continuous authentication process.

Authentication result

Each mechanism calculates the probability that the data sample belongs to the owner of the token. After each recording, this probability is updated using Bayes' Law. The process is also known as a Bayesian update, and is descried in the following equation:

$$P(H|E) = \frac{P(H) * P(E|H)}{P(E)} \tag{4.1}$$

In the equation above:

- E: Stands for evidence and in this case represents the data sample.
- H: Stands for hypothesis. In this case we refer to the hypothesis that the owner is present.
- $P(H|E)$: Represents the probability of hypothesis H after observing evidence E. This is the nal probability we are trying to compute after each sample. It is also known as the posterior probability.
- $P(H)$: Represents the probability of hypothesis H before observing evidence E. This is also known as the prior probability and is the probability computed at the previous step.
- $P(E|H)$: Represents the probability that the current evidence belongs to hypothesis H. It is the probability outputted by the mechanism given the sample data.

- $P(E)$: This is the model evidence, and has a constant value for all hypothesis.

Although $P(E)$ is constant we need its value in order to calculate $P(H|E)$. We can compute it by using the \Law of total probability":

$$P(E) = \sum_n P(E|H_n) * P(H_n) \tag{4.2}$$

Using equation 4.2, Bayes' Law 4.1 can be written as:

$$P(H|E) = \frac{P(H) * P(E|H)}{\sum_n P(E|H_n) * P(H_n)} \tag{4.3}$$

Our model contains only two hypothesis[2]: the recording of the data either belongs to the owner, or not. We can therefore consider $P(H)$ to be the probability that the data belongs to the owner and $P(\neg H)$ the probability that the data belongs to someone else. This means the value of $P(\neg H)$ is $1 - P(H)$ and $P(E|\neg H) = 1 - P(E|H)$ Introducing this in equation 4.3, the rule for updating the mechanism's probability becomes:

$$P(H|E) = \frac{P(H) * P(E|H)}{P(H) * P(E|H) + P(\neg H) * P(E|\neg H)} \tag{4.4}$$

Equation 4.4 represents the nal probability that the owner is present given the sampled data. All the variables in this equation are known, for reasons explained above.

We have de ned how individual scores are calculated, and that each mechanism has a decaying weight. Using this data we can calculate the overall score of the scheme. This is performed by using the following modi ed weighted sum:

$$P_{Total} = \frac{\sum_{i=1}^{n}(w_{id} * P_i(H|E_i))}{\sum_{i=1}^{n} w_i} \tag{4.5}$$

In equation 4.5, w_{id} represents the decayed weight of mechanism i, and w_i is its original weight. We have chosen this model because in a scenario where the token has no sample data to collect, all mechanisms would decrease their weights simultaneously. Using a

[2] Arguably there is a third case where the data sample is not a valid recording of an user. This case is ignored and no probability is computed. The only result in this scenario is the resuming of the decay process in the weight of the mechanism.

simple weighted sum, this would misleadingly provide a high overall result, even though all decayed weights would be low.

4.3 Related work

Clarke et al [13] present statistics con rming the need for an unlocking scheme di erent from PINs. They conduct a couple of surveys trying to evaluate the reliability of a PIN in unlocking a mobile phone. The paper reveals a high number of bad practices involved in PIN authentication: reusing the PIN with other authenticators, forgetting the PIN, sharing the PIN with someone else, 45% of owners never change the default factory code, 42% only change it once after buying the device.

A promising result showed in the paper is that 83% of users are willing to accept a biometric authentication mechanism to unlock their devices. The following biometric mechanisms were included in the study: ngerprint analysis, voice recognition, iris recognition, hand recognition, keystroke analysis [14], and face recognition. The paper also shows that 61% of users would accept an unobtrusive biometric continuous authentication mechanism. Using multiple biometrics for continuous authentication is mentioned brie y, but each mechanism is used individually based on what the user is doing. As an example, when the user walks he is authenticated using gait recognition, and while he is speaking on the phone, voice recognition.

In a di erent paper, Clarke et al [6] study PIN alternatives for mobile phone unlocking. The authors conduct a survey with interesting results. A remarkable 11% of participants were not aware of PIN authentication. An average of 81% of participants agree that PINs should be replaced with a mechanism that provides better security. Although they report the need and desire for a di erent type of phone unlocking, many of them do not use currently available alternatives.

Gregory Williamson [15] writes in his PhD dissertation about the need for an enhanced security authentication mechanism for on-line banking. He proposes a multi-factor authentication model, and presents two interesting options: the traditional one where both mechanisms are required in the multi-factor model (blanket authentication), and one where the second authentication mechanism is only requested from the user if the transaction appears to be risky (risk mode authentication). A risky situation is de ned

as either an important transaction like withdrawing money, or a transaction made under unusual circumstances such as using an unknown device.

Risk mode authentication is similar in concept with the explicit authentication request used in our proposed token unlocking scheme. Furthermore, Williamson shows that 75% of users agree with having biometric authentication as a second factor authentication for passwords. This shows promising results in adopting our scheme for token unlocking purposes.

Elena Vildjiounaite et al describe in their paper [16] a similar authentication mechanism based on combining biometric authentication data on mobile phone devices. The authors explore an alternative to PINs based on a two stage \risk mode authentication". The first stage combines biometric data in order to achieve continuous authentication. This is achieved by training a cascade classifier to a target false acceptance rate (FAR)[3]. Data from mechanisms is merged using a weighted sum fusion rule. Mechanism weights are chosen based on their error rates. The second stage is only enabled if the cascade classifier does not identify the owner as being present. In low noise scenarios, continuous authentication is achieved without the need for an explicit challenge 80% of the time. In noisy situations (city and car noise), the percentage drops ranging from 40 to 60%. The cascade classifier was trained with a FAR of 1%, with results showing a false rejection rate (FRR)[4] of only 3 to 7%.

The paper by Elena Vildjiounaite et al [16] is similar with the solution proposed in this dissertation. It also combines multiple authentication mechanisms, each being assigned different weights. Differences between the two are the fact that weights are maintained static over time. The overall sum is computed differently, and there is no mention of Bayesian updates or probabilities. Furthermore, the authors use a classifier instead of producing a confidence level, which cannot be used for granting different levels of security. The results presented by this paper are however encouraging, showing that continuous authentication presents good results using multiple biometric authentication mechanisms.

[3] The false acceptance rate is the equivalent of false positive precision. It is the probability of incorrectly granting authentication privileges to an user

[4] The false rejection rate is the probability of incorrectly denying access to the rightful owner.

4.4 Conclusions

We have designed a new Pico unlocking mechanism that supports Pico's claims for continuous and memory effortless authentication. The scheme is guaranteed to improve on the existing Picosiblings solution at least by offering a better way of correctly identifying its owner.

An evaluation of the scheme is not yet offered because mechanisms such as \Negligible-cost-per-user" are implementation dependent. The next chapter will present a prototype solution. This offers a better definition of the scheme, that can be evaluated using the token unlocking assessment framework. The results will be compared with the current Picosiblings implementation allowing for further analysis and conclusions.

Chapter 5

Implementation Prototype

In this chapter we have developed a prototype for the scheme proposed in section 4.2. We have chosen as an implementation platform the Android Nexus 5 smart phone. The device o ers enough sensors to perform biometric and behavioural analysis [1]. These resources will be used to demonstrate that the scheme can be implemented using similar dedicated hardware that may o er more security features.

We have included a brief overview of the Android development model and the platform's security features in appendix A. This information provides an introduction for understanding the principles used in the implementation of the prototype.

5.1 Implementation overview

The Android token unlocking scheme is designed to work as a bound service. It is implemented in the \UAService" [2] class. Feedback is provided to clients either after an explicit request or through periodic broadcasts.

Each authentication mechanism that participates in the scheme may have di erent requirements for sampling and processing data. As an example, voice recognition can gather optimal data during a phone call [3], while face recognition when the phone screen

[1] The full range of sensors supported by the Android platform can be found here: http://developer.android.com/guide/topics/sensors/sensors_overview.html (accessed on 28.05.2014)

[2] The name of the class stands for User Authentication Service

[3] Call events can be intercepted by registering a listener for the PHONE_STATE event

is unlocked. Therefore, to enable more exibility in the individual mechanisms' implementation, they are developed as independent services.

\UAService" communicates with the authentication mechanisms by binding to their service. On prede ned time intervals it requests the con dence level and weight of each mechanism. Using this data it then calculates the overall result according to the design in section 4.2.1. Feedback is sent back to each registered client for interpretation.

5.2 Implementation details

This section presents the implementation of the design proposed in section 4.2. The full source code for the prototype can be downloaded from github [4].

The structure of the Android application is presented in gure 5.1. Each colour represents an Android component (e.g. activity, service). This should provide a better understanding of the overall design. A full detailed UML diagram could not be included with this dissertation, but can be downloaded from github [5].

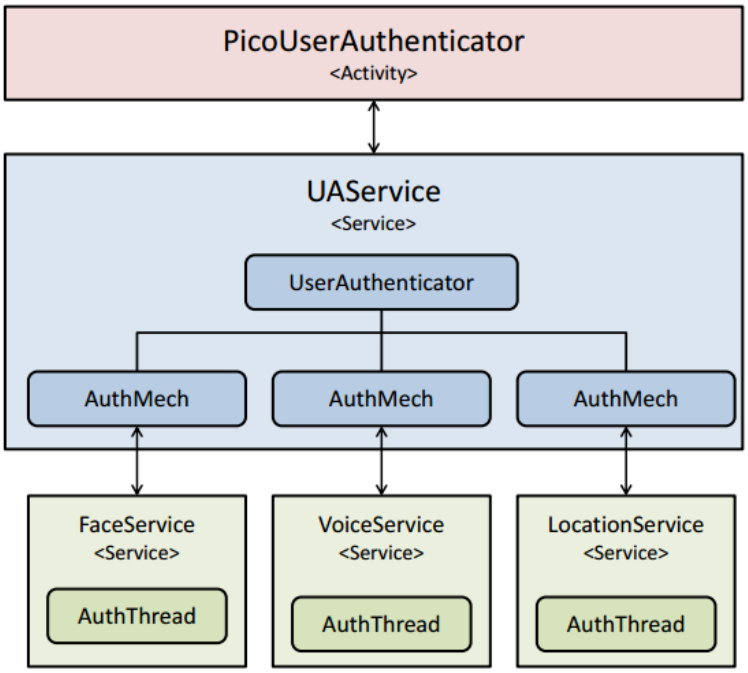

FIGURE 5.1: Authenticator design overview

[4] https://github.com/cristiantoader/fyp-pico
[5] https://github.com/cristiantoader/fyp-pico/blob/master/PicoUserAuthenticator/dissertation/Pictures/detailed-uml.png

5.2.1 UAService

The token unlocking mechanism is started using the \PicoMainActivity" class, which acts as a Pico client. The scheme itself is developed to be used by binding the \UAService" component. An UML diagram of the di erent components related to \UAService" is attached in gure 5.2.

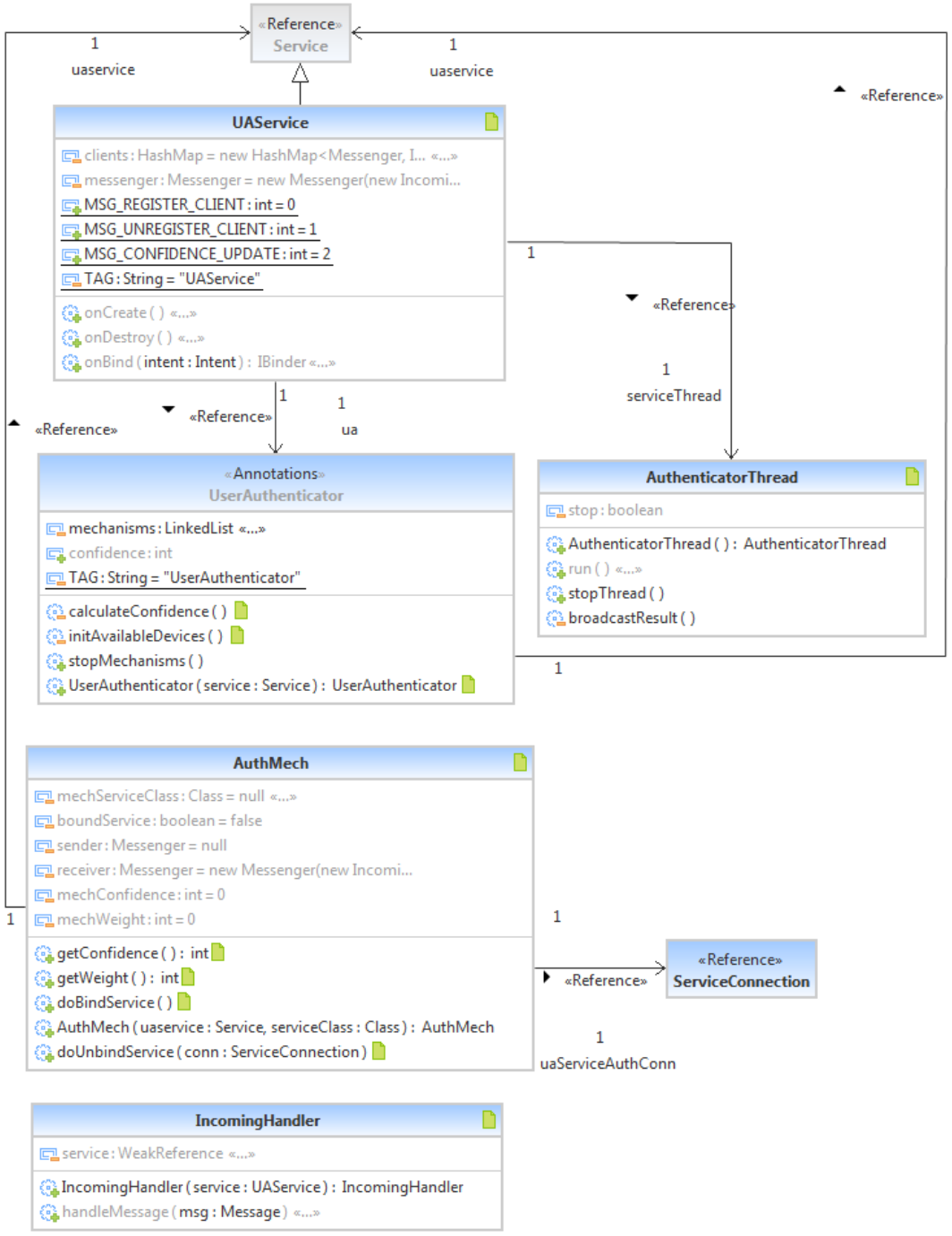

FIGURE 5.2: UAService components

\PicoMainActivity" starts \UAService" using the \Context.startService()" method. Communication is enabled by binding to \UAService" using \Context.bindService()". This approach protects the lifetime of the authenticator. When \PicoMainActivity" gets sent to background and loses control of the screen, \UAService" is not explicitly unbound. This guarantees that the service will continue running in the background, and should also prevent malicious components from stopping it.

Clients need to bind \UAService" to receive authentication updates. When bound, communication is enabled by exchanging \Messenger" objects using the IBinder interface. A \Messenger" allows another component to send \Message" objects, and de nes how they are handled by the receiver through an \IncomingHandler". The \Messenger" queues all requests on a single thread, and therefore the application does not require to be thread safe.

When bound by a client, the \IncomingHandler" used by \UAService" exposes the following API de ned by the \what" parameter of a received \Message":

MSG_REGISTER_CLIENT

Used for registering a client for periodic broadcasts of the current authentication con dence level. Feedback is provided at a xed time interval of 1000ms [6].

MSG_UNREGISTER_CLIENT

Used for any application component to unregister as a listener from \UAService".

MSG_GET_STATUS

Used by a client for requesting an explicit authentication status update.

The \UAService" service wraps an \UserAuthenticator" proxy object that implements most of its functionality. The \UserAuthenticator" is responsible for collecting data from authentication services, and computing the nal con dence level. The fact that the mechanisms are independent services is hidden from the \UserAuthenticator" using

[6] An alternative implementation explored in the project was to have each client also register a con dence level using the \arg1" parameter of a \Message". In this case, the authenticator would only provide each client with a locked/unlocked result. However, this would shift the meaning of client to that of an authentication session, with state managed by the unlocking scheme. A client would therefore have multiple connections, requiring more ICC. Since all \Messenger" requests made to \UAService" are queued to a single thread, this would slow down the feedback process and possibly lead to a denial of service attack. Therefore we have chosen to reduce the communication overhead, and have each client manage the status of its authentication sessions based on the con dence level provided by the unlocking scheme.

\AuthMech" observer objects. Each \AuthMech" starts and binds an authentication mechanism, listens for con dence level updates, and keeps track of the most recent value.

Each authentication mechanism service extends the \AuthMechService" abstract class. This de nes them as bound services with the same \IncomingHandler" implementation. The communication with \AuthMech" is therefore standardized, providing the following \Messenger" API:

AUTH_MECH_REGISTER

Used for registering the \UAService" client to the \AuthMechService".

AUTH_MECH_UNREGISTER

Used for unregistering the \UAService" client from the \AuthMechService".

5.2.2 Authentication mechanisms

In order to create a functional prototype, we have implemented a number user authentication mechanisms. The result quality of each mechanisms is outside the scope of this project. Their sole purpose is to demonstrate that sensor data o ered by smart phones can be used to implement biometric and behavioural analysis.

When developing an individual authentication mechanism, the following abstract requirements need to be satis ed:

1. The result needs to be quanti able in the form of a percentage ranging from 0 to 100, where 100 means that the mechanism has 100% con dence that the owner of the token is present.
2. The mechanism needs to support continuous authentication.
3. The authentication process needs to be e ortless and preferably unobtrusive for the user.

A list of authentication mechanism examples that can be implemented on the Android platform is presented in appendix C.

Each mechanism developed for this scheme extends the \AuthMechService" abstract class. As mentioned, this class de nes the mechanism as a bound service. The communication with \UAService" is standardized by implementing \Service.onBind()" to register the same \IncomingHandler" implementation. Furthermore, \AuthMechService" de nes the decay implementation of a mechanism's weight. This is developed using a \Handler" object that schedules a \Runnable" to execute at a prede ned time interval. The \Runnable" is responsible for decreasing the mechanism's weight and sending the result to the corresponding \AuthMech". Figure 5.3 provides an overview of the components interacting with \AuthMechService".

Although it is not enforced, each authentication mechanism has the same general design. The mechanism's \Service.onCreate()" method initialises the weight of the mechanism, and starts a thread responsible for periodically collecting sensor data using an access object (DAO). The samples are analysed using a class that mediates interaction with other components and libraries. After each successful analysis, the result is sent back to the corresponding \AuthMech" and the decay process is started using \AuthMechService.startDecay()".

The biometric libraries used in the prototype provide feedback as an Euclidean distance. To convert it to a percentage con dence level, we de ne for each mechanism an acceptable threshold. Any result above the threshold is considered too high and is truncated to its value. Using equation 5.1 we convert the Euclidean distance to a con dence level. Dividing the distance over the threshold yields a value between 0 and 1, where 1 is a very large distance and hence a bad result. By using one minus this value we invert the meaning. Values will range between 0 and 1, and 1 corresponds to a con dence level of 100%. This result is $P(E|H)$ from equation 4.4.

$$P(E|H) = 1 - \frac{distance}{THRESHOLD} \tag{5.1}$$

Having known $P(E|H)$ we continue to calculate $P(H|E)$ by using the Bayesian update formula de ned in equation 4.4. When calculating the nal value of the mechanism, we multiply $P(H|E)$ with the current decayed weight.

The following mechanisms have been implemented as part of the prototype: voice recognition, face recognition, location analysis, and a dummy mechanism used for testing.

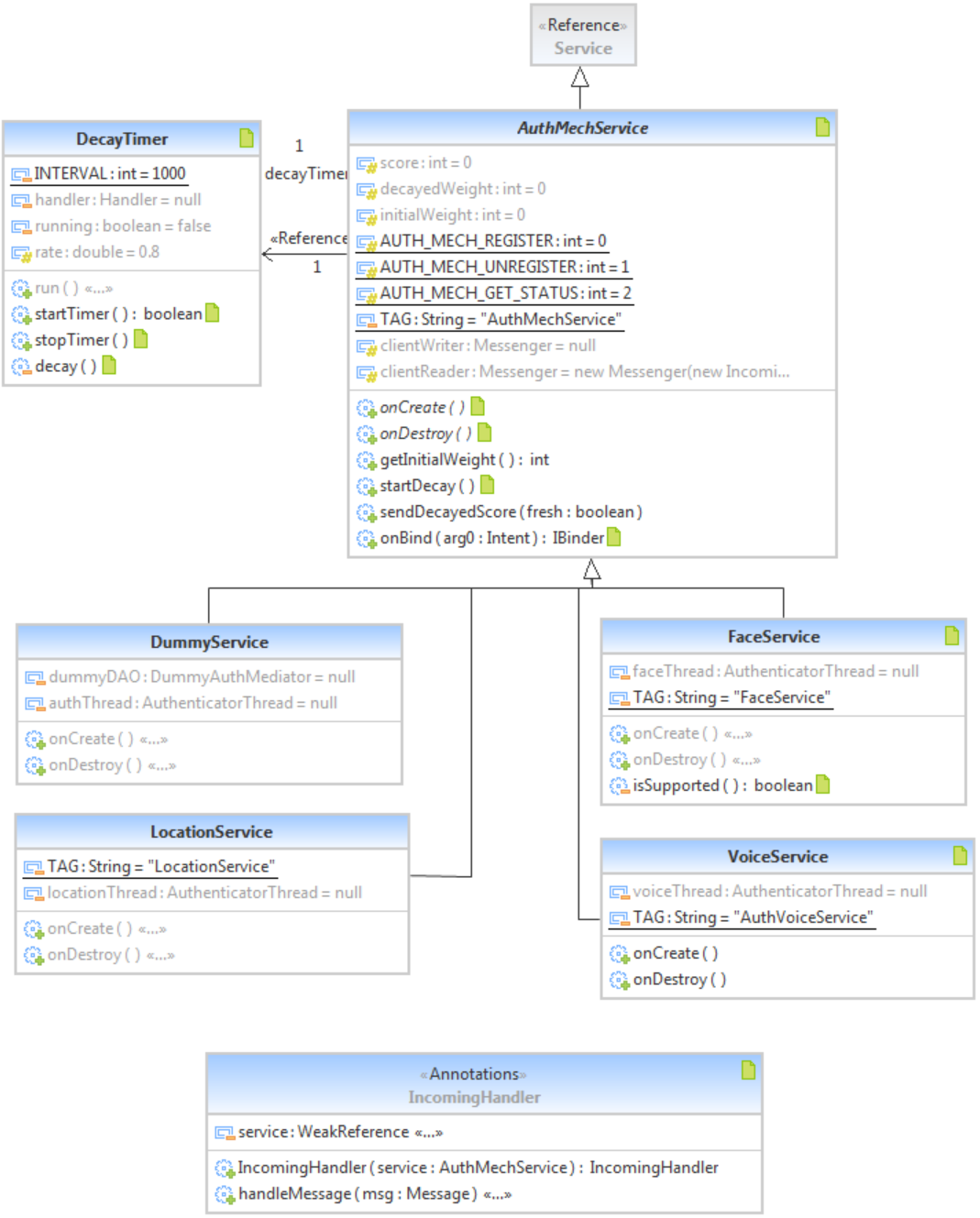

FIGURE 5.3: AuthMechService components overview.

The next sections will provide details regarding their functionality and implementation process.

5.2.2.1 Dummy mechanism

A dummy authentication mechanism was developed for testing the overall scheme. It produces random con dence levels within a prede ned range, which provides a controlled testing environment. An overview of the components involved in the dummy mechanism are shown in gure 5.4

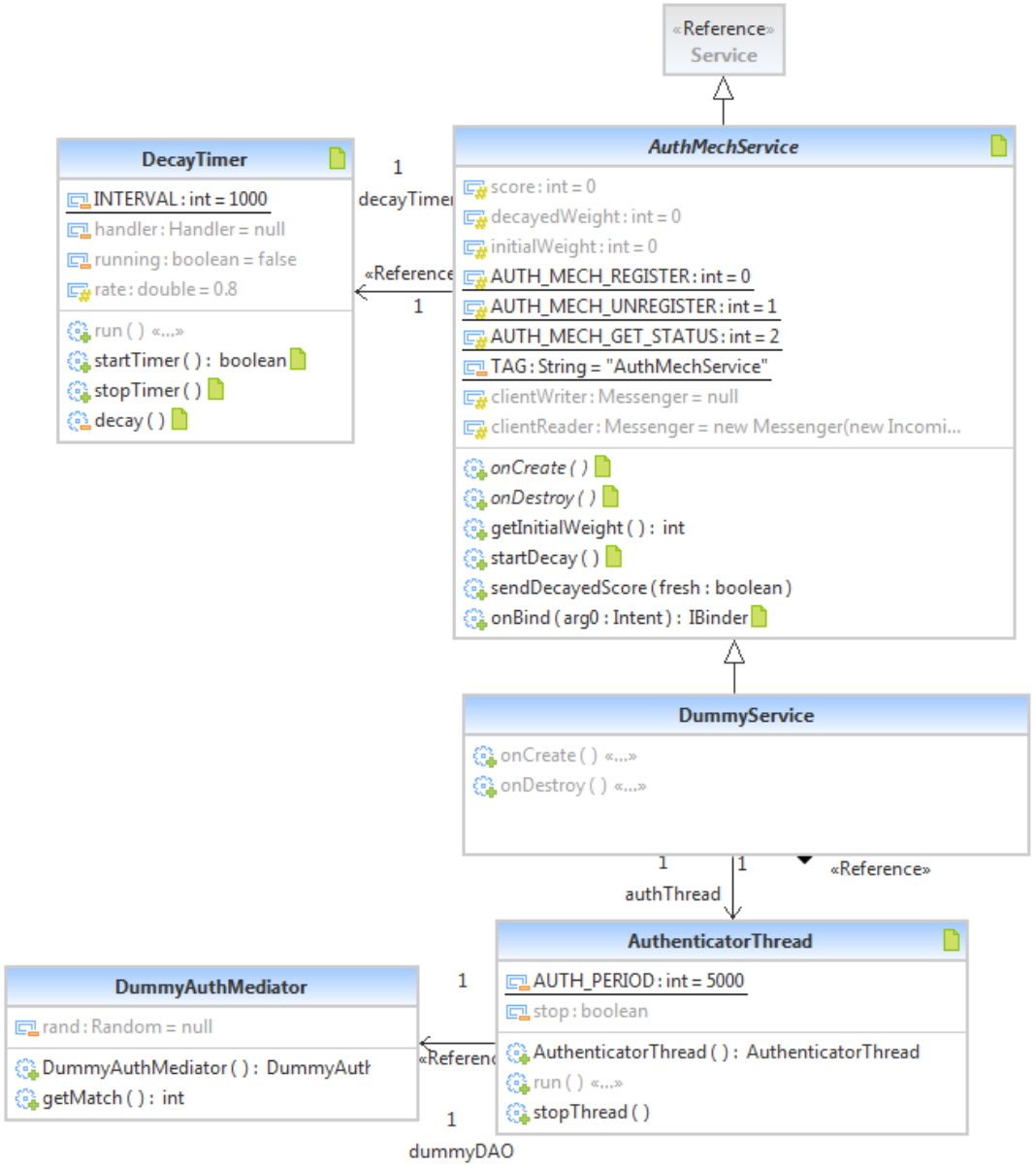

FIGURE 5.4: Dummy mechanism

The mechanism was developed in the \DummyService" class and was designed consistently with the application model. Its \DummyService.onCreate()" method creates an authentication thread that periodically generates random values and sends them to

\UAService". It does not use any DAO, in order not to over complicate its implementation. The random values are generated using a \DummyAuthMediator" object. Once the mechanism's con dence level is calculated, the weight decay process is started using \AuthMechService.startDecay()" and result updates are sent back to \UAService".

5.2.2.2 Voice recognition

The voice recognition mechanism is implemented in the \VoiceService" class and extends the \AuthMechService" abstract class. The \VoiceService.onCreate()" method starts a thread that periodically gathers data from the microphone, performs biometric authentication, and produces a con dence level. An overview of the components involved in this mechanism is presented in gure 5.5.

The library used for biometric voice recognition is called Recognito [7], and was developed by Amaury Crickx. It uses a text independent speaker recognition algorithm developed in Java (SE). Its author claims very good results in scenarios with minimal background noise [8].

Porting Recognito for Android required no changes. However, in order to package the library, a subset of the \rt.jar" Java (SE) library is needed for sound le formats. Including the full \rt.jar" is not possible due to a package name collision with Android \javax.*" system libraries. Therefore, we have included only the \javax.sound.*" package using a custom jar. This was purely done to trick the Android Java compiler to build the application. Using \javax.sound" features would generate a runtime error. Therefore, we only use Recognito functions which require direct data input, without any knowledge of sound le formats.

In order to gather and manage samples compatible with the Recognito library we have created the \VoiceDAO" class. Microphone input is gathered using the following predened con guration:

- Sample rate: 44100
- Channel con guration: AudioFormat.CHANNEL_IN_MONO

[7] The library can be downloaded using github from the following link: https://github.com/amaurycrickx/recognito

[8] It was tested by its author on TED talks such as: https://www.ted.com/talks/browse (visited on 06.01.2014)

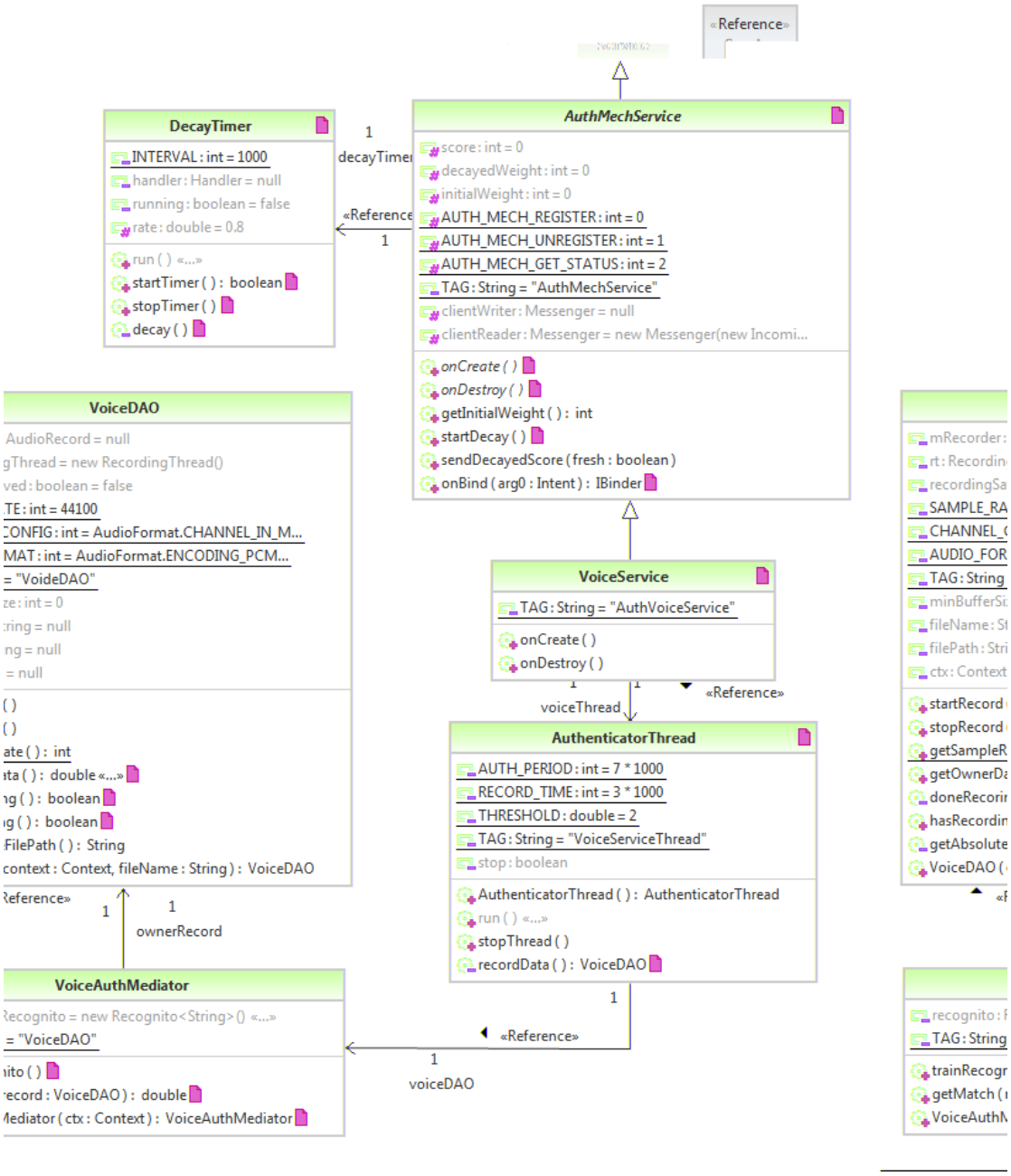

FIGURE 5.5: Voice recognition overview.

- Audio format: AudioFormat.ENCODING_PCM_16BIT

The minimum bu er size required by `VoiceDAO" is device dependant and pre-calculated in the constructor. The class wraps an Android \AudioRecord" object used for gathering microphone data. The recording is saved as a le and loaded into memory when needed. The implementation is not e cient, but serves the purpose of the prototype.

The \VoiceAuthMediator" class was created to mediate calls to the Recognito library. When initialised, it loads the owner con guration, and a prede ned set of background noises. It then creates a \Recognito" object and trains it using the data. This is performed using the \Recognito.createVocalPrint()" method.

Every prede ned time interval, the \VoiceService" authentication thread records data in \double[]" format using a \VoiceDAO" object. It uses the \VoiceAuthMediator" to analyse the sample. This returns the Euclidean distance to the closest match, which is either the owner, or one of the background noises used for training. We convert the Euclidean distance from \VoiceAuthMediator" to a percentage using equation 5.1. The nal con dence is computed, stored in the service, and the decay process is started. Whenever the weight is modi ed, a \Message" is sent to the corresponding \AuthMech" and updates its value.

5.2.2.3 Face recognition

The face recognition mechanism was implemented in the \FaceService" class and extends \AuthMechService". When created, the service starts an authentication thread that periodically collects data from the camera, performs biometric face recognition, and produces a con dence level. An overview of the components involved in the face recognition mechanism is shown in gure 5.6.

Face recognition is implemented using a modi ed version of the \Javafaces" library [9]. It is written entirely using Java (SE), but unfortunately uses the \javax.imageio." package that is not available in the Android API. A considerable amount of code needed to be ported for the Android platform. Although not currently optimised for public use, the new library is available at the following link: https://github.com/cristiantoader/JavafacesLib.

We will brie y present the changes made when porting the \Javafaces" library. The \Bu eredImage" class had to be replaced by its closest Android equivalent, which is \Bitmap". This required a number of adaptations due to di erences between the two classes. For example, \Bu eredImage" grey-scale images use a single colour channel for the grey intensity value. This had to be changed to the Bitmap format that uses all 3 channels. Additional modi cations were required due to data type mismatches, as well

[9] The \JavaFaces" library is maintained at the following address: https://code.google.com/p/javafaces/

FIGURE 5.6: Face recognition overview.

as other related issues. Furthermore, The API was modi ed to support direct \Bitmap" input in order to add more exibility and lighten the main code of the authenticator.

Each prede ned time interval the \FaceService" authentication thread collects a picture from the camera. Its validity is determined using the Android \FaceDetector" class. If the image contains a face it continues to be processed, otherwise the weight decays until a new data sample is collected. The image collection and validation was simpli ed by developing the \CameraDAO" class.

The \FaceAuthMediator" class was implemented to mediate calls to the \Javafaces" library. When the authentication thread is started, a \FaceAuthMediator" is created and used for training the biometric recognizer. It then analyses face data sampled from \CameraDAO" in order to produce a result. The return value represents the Euclidean distance between the face captured from the camera and the registered owner face. This distance is transformed into a percentage con dence level using equation 5.1, and the decay process is restarted.

The Android API does not easily allow for a Camera picture to be taken without any sort of noti cation to the user. Both a shutter sound and a visual preview display should be present. The sound can be disabled by not providing a shutter callback function when calling the Camera.takePicture() method.

Disabling the user preview of the camera was more di cult to achieve. The solution was to exploit an Android feature that allows to render the preview in a \SurfaceTexture" object. This satis es the API's requirement to have a visual display preview for the camera, while the \SurfaceTexture" itself does not need to be displayed on screen. Therefore a picture can be taken from a background service without any interruption to the user.

Another problem encountered by the face recognition service is data sizes. When the \Javafaces" library performed face recognition, the device was running out of memory. This caused the app to be closed by the Android OS. To x this issue, all images collected from the camera are resized to 50% before they are processed.

The library combined with the Android SDK does not provide accurate results. The reason is that it requires an input image perfectly embedding the face of the user. Unfortunately, although the Android SDK o ers face detection, it only provides the location of the midway coordinate, and distance between the eyes. Using this data alone, an accurate crop cannot be made. As a solution, yet another library is needed to properly detect face regions. This would provide better input data and increase the precision of the mechanism.

5.2.2.4 Location analysis

This mechanism is based on gathering location data and using it to generate a probability that the owner is present. This is implemented in the \LocationService" class and extends the \AuthMechService" abstract class. Data is collected periodically using the \LocationManager" provided by the Android API. An overview of the components involved in the location analysis mechanism is shown in gure 5.7.

A DAO object is used to mediate Android API calls and manage the existing owner con guration. It is implemented in the \LocationDAO" class. It o ers functionality for gathering and saving location updates. It is developed to use the most accurate data provider. The Android API o ers the following sources of collecting \Location" data:

- GPS_PROVIDER: Collects data from the GPS.
- NETWORK_PROVIDER: Collects data from cell tower and WiFi access points.
- PASSIVE_PROVIDER: Passively collects data from other applications which receive \Location" updates.

External libraries were not used for the authentication process. We have developed a primitive location analysis algorithm in the \LocationAuthMediator" class. During the con guration stage, which is a process managed by \LocationActivity", location data is sampled every 5 minutes and saved in internal storage. After the process has ended, each time a \Location" is sent for authentication it is compared with all the locations saved during the con guration process. The nal result is the minimum number of meters between the current \Location" and any other saved \Location".

When the \LocationService" mechanism is started, its \onCreate()" method spawns an authentication thread. This thread periodically requests the current location using the \LocationDAO". Data is returned in a \Location" object and is provided as input to the \LocationAuthMediator". Although the result is represented in meters, and is not an Euclidean distance, it can be converted to a percentage using equation 5.1. Once the nal probability is calculated, the weight decay process is restarted.

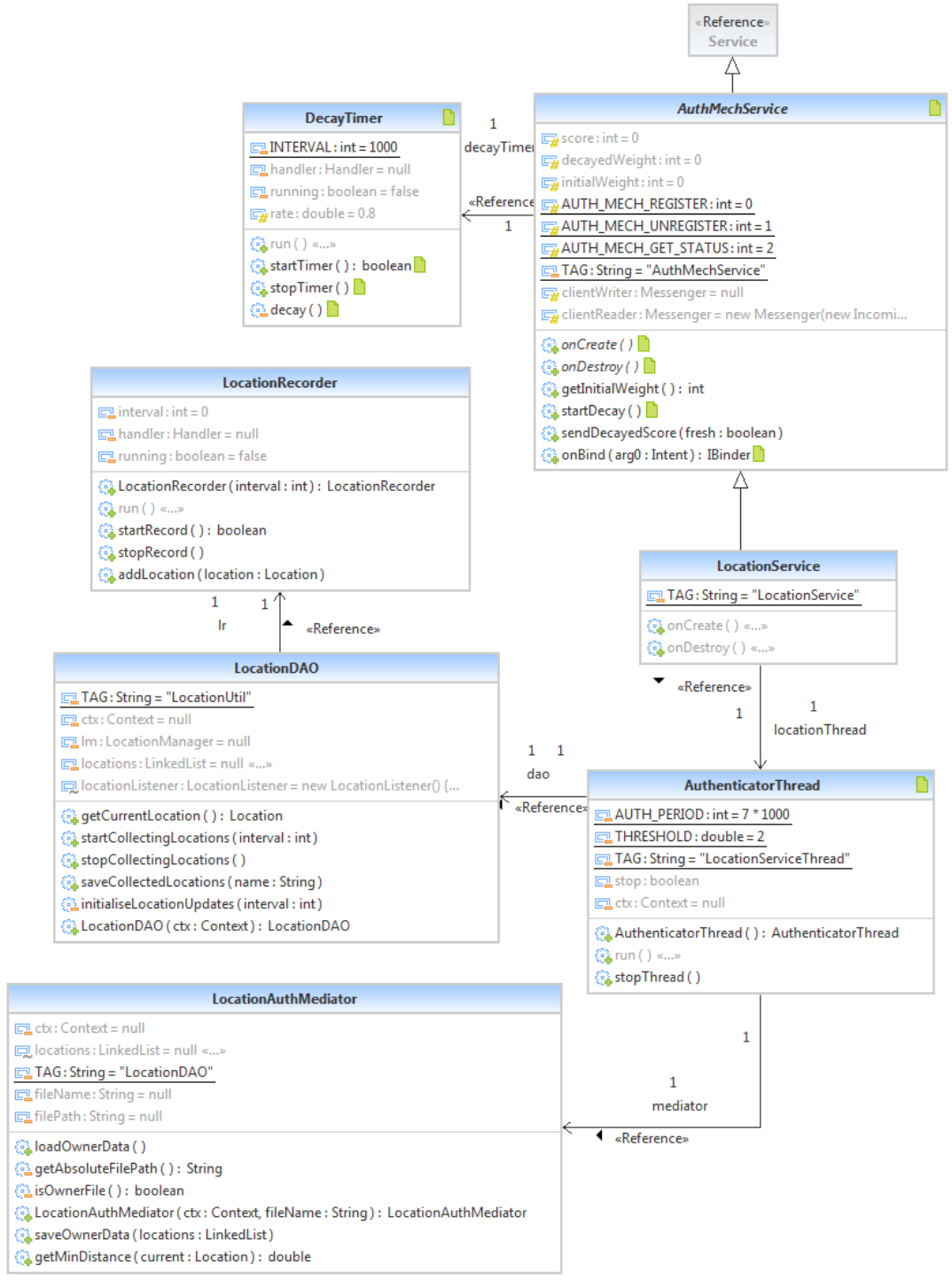

FIGURE 5.7: Location analysis overview.

5.2.3 Owner con guration

There are a number of components that are used in the con guration of the prototype. Each authentication mechanism has a corresponding Activity that can be started from

the main Activity called \PicoUserAuthenticator". These are used to register owner biometrics needed by the mechanisms.

Each con guration Activity uses the same DAO class as its corresponding authentication mechanism. The DAO is used for collecting and storing owner data. Given that the overall size is relatively small, les are kept in internal storage.

5.2.4 Cryptographic protection

All biometric data registered by the owner and used by the mechanisms is stored in internal memory. This is protected by the Linux permissions model, and provides su cient security from other applications. However, if an attacker acquires \root" privileges, the biometric les would be completely exposed. To add an additional layer of security, we have included cryptographic protection of owner data.

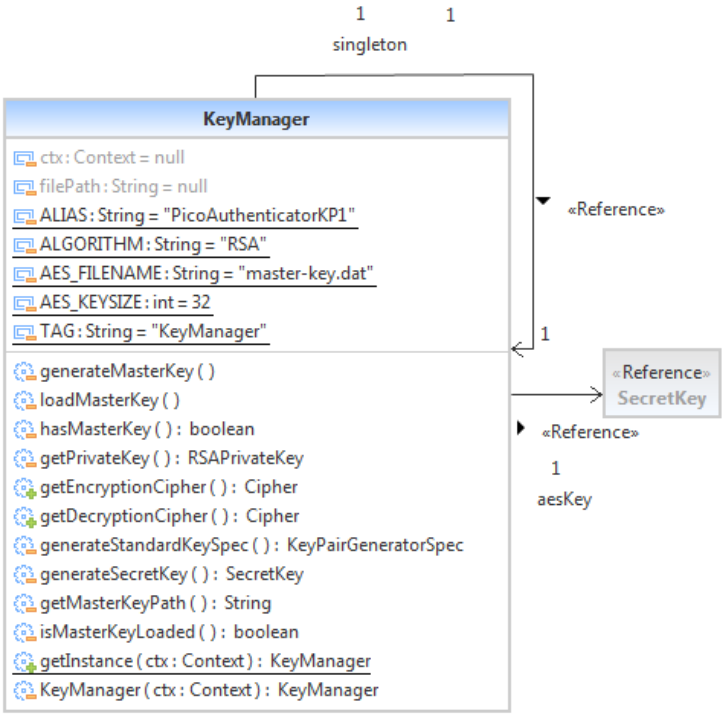

FIGURE 5.8: KeyManager overview.

The cryptographic layer is implemented in the \KeyManager" class. An overview of the class structure can be seen in gure 5.8. It uses the \KeyStore" API in order to keep an RSA key pair securely stored. Starting with Android 4.3, credentials stored in the Android \KeyStore" are not extractable because they are hardware secured using:

Secure Element, TPM, or TrustZone. The securely stored RSA key pair is used to decrypt an AES master key that is kept encrypted in the app's internal memory. Once it is retrieved, the AES master key is used for encryption/decryption of con guration les.

All mechanisms except for face recognition use this cryptographic layer to keep data encrypted in internal memory. The face recognition mechanism does not support this feature due to the \Javafaces" library, which is implemented to process image les independently with no cryptographic support.

5.3 Conclusion

We have described the Activity and Service components developed for the prototype, as well as their communication ow. We have ported two biometric libraries and developed a location analysis mechanism. DAO objects facilitate cryptographic access to owner con guration les, and additional classes mediate the calls to external libraries. An overview of the app design is available on github [10].

One of the limitations of the prototype is the lack of explicit authentication mechanisms. Another issue is the precision of the biometric mechanisms, in the lack of better libraries. However, due to the modular design of the application, existing mechanisms can be improved simply by importing a new library and modifying the corresponding mediator class. The existing set of mechanisms can be increased by creating a new class that extends \AuthMechService" and implements the algorithm's logic. In order to be managed by \UAService", the new mechanism needs to be included in the \UserAuthenticator.initAvailableDevices()" method.

5.4 Related work

Liang Cai et al [17] analyse ways of protecting users from mobile phone sensor sni ng attacks. The authors design a framework used for protecting sensor data from being leaked. From a security perspective the user should not to be trusted with granting

[10] https://github.com/cristiantoader/fyp-pico/blob/master/PicoUserAuthenticator/dissertation/Pictures/detailed-uml.png

permissions to di erent applications. A solution provided in the paper is for sensors to become locked once they are used. A downside to this is that malware may deny service to legitimate applications by creating a race condition for acquiring a sensor lock. This can be solved by using an user noti cation, allowing for the owner to decide which application acquires the lock.

The paper by Derawi et al [18] presents the feasibility of implementing gait authentication on Android as an unobtrusive unlocking mechanism. According to the de nition o ered by the authors \gait recognition describes a biometric method which allows an automatic veri cation of the identity of a person by the way he walks". The Android implementation developed by the authors has an equal error rate (EER) of 20%. Dedicated devices have an EER of only 12.9%, and the main cause for this is the sampling rate available at that time (2010). The authors have used a Google G1 phone with approximately 40-50 samples per second. This is much inferior to dedicated accelerometers that sample data at 100 samples per second. However, by conducting personal experiments with the accelerometer of a Google Nexus 5 phone, the rates of the highest sampling setting (SENSOR_DELAY_FASTEST) are over 100 samples per second. Therefore the current performance of the prototype developed in this paper should be closer to 12.9%.

Ming et al [19] present in their paper how to improve speaker recognition accuracy on mobile phone devices in noisy conditions. This approach uses a model training technique based on which missing features may be used to identify noise. The focus of the paper is designing and implementing the biometric mechanism.

Another way of performing speaker recognition involves using voiceprints. These are a set of features extracted from the speaker sample data. Kersta [20] explains the mechanism in more detail. The bene t of having feature extraction, as opposed to a di erent voice recognition mechanism, is that voiceprints do not require any secrets. This increases the usability of the mechanism in di erent scenarios required by the Pico authenticator. However, a downside to this approach is that it makes replay attacks easier to perform. Any recording of the user is su cient for an attacker to trick the biometric mechanism.

A popular face recognition paper was written by Turk and Pentland [21]. The biometric authentication process is based on the concept of \eigenfaces". This is a name given for the eigenvectors that are used to characterise the features of a face. These values

are projected onto the feature space. Using Euclidean distances in the feature space, classi cation can be performed in order to identify users. An implementation of this mechanism was used with the Pico unlocking scheme prototype.

An unconventional authenticating mechanism is presented by Clarke and Furnell [22]. They use keystroke analysis in order to make predictions regarding the user of a phone. This mechanism is unobtrusive and gathers data during normal user interactions such as typing a text message or phone number. It is based on a neural network classi er, reporting an EER of 12.8%. Input data used for classi cation is composed out of timings between successive keystrokes, and the hold time of a pressed key.

Chapter 6

Evaluation

This chapter presents an evaluation of the proposed token unlocking mechanism. We start by performing a threat model of the Android prototype. This should reveal any security limitations of the implementation, as well as the overall scheme. We continue by analysing the performance of the prototype and discuss how it can be improved.

Given a well de ned implementation (5), we assess the scheme using the token unlocking framework (3.2), and compare the results with the Picosiblings solution. In order to check for overall improvements, we use the UDS framework to evaluate a Pico token that uses the proposed token unlocking scheme, and compare the results with the original work by Bonneau et al [5].

6.1 Threat model

The threat analysis is performed from an availability, integrity, and con dentiality perspective. We consider the security mechanisms of the Android platform presented in appendix B as prede ned assumptions used in this model. Attack paths are analysed in di erent scenarios based on whether an attacker has physical access to the token or not. Although we are making a security assessment for a prototype developed on the commercial Android platform, similar issues may arise for a future implementation that uses dedicated hardware.

Availability

Breaking the scheme's availability while the device is in the possession of the attacker is relatively trivial. The application can be uninstalled, or the application data cache can be cleared, therefore removing the owner biometric models used by the individual mechanisms. Furthermore, in this scenario the owner is no longer in possession of their Pico, so basically the device is already unavailable.

Let us continue and study what denial of service (DoS) exploits can be achieved by a remote attacker. Removing the owner con guration data from internal storage would make the authenticator unusable. This can only be achieved if the attacker (or a malware application designed by the attacker) manages to get root access on the device. Given the Linux permissions model, there would be no way to protect this data from deletion. However, without root access the application data cannot be accessed or modi ed.

Based on each device platform, multiple apps recording data from a sensor may not be possible. This can enable a DoS attack on the prototype by having malware locking sensors before the authenticator. This would make data collection impossible, and therefore the mechanisms' weights would gradually decrease to 0. The overall con dence level would be lowered, preventing the user from authenticating. After performing experiments, we can con rm this problem for the Google Nexus 5 smart phone when two applications try to record microphone data using di erent sampling rates [1].

The current prototype is susceptible to a DoS attack caused by too many clients registered with the authenticator. Given that no permission is required to register to the \UAService" component, an unlimited number of connections can be made. Therefore, broadcasting the authentication status to each client may cause considerable delays. Furthermore, in order to unburden the developer from developing thread-safe code, ICC is performed on a single thread. This means that by spamming \UAService" with requests, the attacker can achieve a DoS attack for legitimate Pico clients. This can be xed if we only allow access to application developed by the same author as the authenticator app.

[1] The apps were trying to record microphone data using the AudioRecord class; application one was using a sampling rate of 44100 and application two 22050.

Integrity

The prototype stores data in internal storage, and cannot be accessed by other applications due to the Linux permissions mechanism. However, just as mentioned in the previous section, if the attacker gains root privileges it may modify any data on the device. The attack can be performed regardless of physical access.

From a data flow point of view, ICC is performed using the "/dev/binder" node driver. According to the official Android source code [2], although the node is readable and writeable by any application, communication is performed using IOCTL calls. Data is transferred from one component to the other without the possibility to intercept or modify. Given that Android ICC is secure, data either from the sensors or from app components cannot be tampered.

Confidentiality

Android apps may share private resources only if developed by the same author. This is determined by verifying the signature of the app, which is performed using a private key specific to each developer. Therefore, an attack where owner configuration data is leaked due to private resource sharing would only be possible if the attacker manages to acquire the private key that was used for signing the authenticator app. We will consider this to be a scenario outside the scope of the project.

Another case where owner authentication data can be accessed is having malware run with root privileges. This would allow an attacker the rights to read any application's data. However, the owner's biometric files would not be compromised, as they are kept encrypted using the Android Keychain API. Although the data can be read from internal storage, it cannot be interpreted in a meaningful way. Starting with Android 4.3, the Keychain API has hardware support, making the encryption keys non-extractable.

On Android versions earlier than 4.3, the following confidentiality attack path can be performed. Given root access, the attacker may retrieve the AES master key used by the keychain manager to store credentials. Using this key they can then retrieve

[2] For convenience, a link to the binder driver is found here: https://android.googlesource.com/kernel/common.git/+/android-3.0/drivers/staging/android/binder.c (visited on 06.02.2014).

the authenticator's application key used for encrypting owner con guration les. By retrieving this nal key, the attacker may decode sensitive data (i.e. biometric data) and leak it outside of the system, therefore compromising con dentiality.

A solution to this problem is not keeping the key used for decryption on the device. It should be generated on the app's rst run, and communicated securely back to a credentials server. Whenever the authenticator app starts, it would request the key remotely via a secure connection, use it to decrypt owner authentication les, and discard it without saving.

From a data ow perspective, ICC should o er full con dentiality. As previously mentioned, the \/dev/binder" device node used for ICC is managed by a driver which listens for \ioctl" requests. Data cannot be compromised as it is transferred from one component to the other. Only \UAService is an \exported" component that may be accessed by other apps. It only provides nal authentication feedback from the mechanisms, and only exposes a limited API through the IBinder interface object used for \Message" passing.

The communication between the authenticator and Pico can be secured, preventing other applications to register for updates. This can be achieved using runtime permission label checks. Both Pico and its authenticator would need to de ne these permission in their manifest les. Additional checks would need to be added for ICC in \UAService".

Liang Cai et al [17] presents the problem of sensor sni ng. A malware application may collect all relevant data on its own from the user, using the same functionality as the prototype. This would allow for a powerful replay attack in the future. Adrienne Porter Felt et al [23] show that when installing an app only 17% of users pay attention to the Android permissions dialogue, and only 3% understand what each permission represents.

Design model attacks

Pico needs to be unlocked only in the presence of its owner. In order to do so, the token unlocking scheme needs to gather valid sample data. Let us consider a few scenarios and assess any design issues.

The most unfavourable scenario is to have the owner silently sitting at work with their smart phone on the table. Voice and face recognition mechanisms would not gather any

valid samples. Location data can be collected, o ering some con dence that the owner is present. If authentication is required for a high security transaction, such as logging in to an online banking account, the overall score outputted by the scheme would not be high enough to grant access. The scheme would therefore make an explicit authentication request, o ering the user the possibility to generate valid data.

The main problem with the scheme is not denying service to the owner, but falsely granting it. Given the same situation as before, let us assume the owner forgets the smart phone on their desk and leaves the o ce for a break. Just as before, only location data can be collected, providing some con dence that the owner is present. Ideally as the owner leaves, most authentication sessions managed by the Pico should be closed. This should happen once the con dence level becomes too low, and the explicit authentication mechanisms are ignored.

From the authenticator's perspective there is no di erence between the two scenarios presented above. The rst suggests that non biometric mechanisms should provide su cient con dence to provide authentication when the owner \goes silent". The second scenario requires the opposite; when the owner can no longer provide biometric data, they are likely no longer with the token and Pico should lock.

A compromise solution is needed for the two scenarios. Non-biometric mechanisms need to provide a con dence level that is almost su cient to grant access to most medium-level security accounts. Periodically, explicit authentication requests will be made by the mechanism in order to provide a su ciently high score. Given the decaying weights, the con dence level will gradually drop until another explicit authentication is required. The weights and decay rates need to be con gured in such a way that the time interval between two explicit authentication requests is acceptable for the user, without compromising security. We suggest the time interval of 1 minute, but an user study would be more appropriate to determine this value.

An alternative solution to the problem presented above is to have an auxiliary biometric sensor that the owner would carry at all times. A good example is a heartbeat monitor. This can be embedded in an every day item such as a watch. The heartbeat authentication mechanism combined with the existing location analysis mechanism should provide a su ciently high con dence level to unlock Pico for any medium-security authentication session.

6.2 Functional evaluation

TODO: timings, and power consumption..

6.3 Token unlocking framework evaluation

We will continue by evaluating the proposed scheme with the token unlocking framework de ned in section 3.2.

The scheme is \memorywise-e ortless" because it doesn't require any secrets to provide authentication. The sensors used for authentication are embedded in the token, therefore o ering \nothing-to-carry". It is also \easy-to-learn", as user authentication is performed non-obtrusively. As shown in section 6.2, although user authentication is performed in a timely manner, setting up the scheme may take some time. Therefore, the \e cient-to-use" property is only quasi-o ered. The scheme only quasi-o ers \infrequent errors" because of the underlying biometric and behavioural authentication. Any di erences in biometric features that may occur can be resolved by re-con guring the authenticator. The prototype does not have a well de ned secure process for this task. In the lack of additional details we mark the scheme to only quasi-o er \easy-recovery-from-loss". As brie y shown in section 6.1, even in unfavourable scenarios the scheme may still provide authentication to the token. The \availability" property is therefore satis ed.

Given that multiple continuous authentication mechanisms are combined, the scheme o ers \accessibility" to any user, regardless of disabilities. Since it is implemented as an Android app, it has a \negligible-cost-per-user" both for the owner and the developer. It is not \mature" since it has only been prototyped. The \non-proprietary" property is o ered, as long as the individual mechanisms are developed using free to use algorithms and libraries.

From a security perspective the scheme is \resilient-to-physical-observations". If an attacker would have valid pre-recordings of the owner for all biometric mechanisms, they would still need to perform these in a location considered safe by the authenticator. Furthermore, the replays would have to be performed periodically in order to keep the authentication session alive. Therefore, due to the di culty to perform a replay attack,

Property	Picosiblings	Proposed scheme
Memorywise-e ortless	O ered	O ered
Nothing-to-carry	Quasi-o ered	O ered
Easy-to-learn	Not-o ered	O ered
E cient-to-use	Quasi-o ered	Quasi-o ered
Infrequent-errors	Quasi-o ered	Quasi-o ered
Easy-recovery-from-loss	Not-o ered	Quasi-o ered
Availability	O ered	O ered
Accessible	O ered	O ered
Negligible-cost-per-user	Not-o ered	O ered
Mature	Not-o ered	Not-o ered
Non-proprietary	O ered	O ered
Resilient-to-physical-observations	O ered	O ered
Resilient-to-targeted-impersonation	O ered	O ered
Resilient-to-throttled-guessing	O ered	O ered
Resilient-to-unthrottled-guessing	O ered	O ered
Resilient-to-theft	Quasi-o ered	O ered
Unlinkable	O ered	Not-o ered
Continuous-authentication	O ered	O ered
Multi-level-unlocking	Not-o ered	O ered
Non-disclosability	Not-o ered	O ered

TABLE 6.1: Token unlocking framework results compared with Picosiblings.

the scheme is considered to be \resilient-to-targeted-impersonation". The \resilient-to-throttled-guessing", \resilient-to-unthrottled-guessing", and \resilient-to-theft" properties do not apply. The scheme is not \unlinkable" because biometric data is unique for each individual. In order to satisfy the Pico requirements, all mechanism involved in the token unlocking process support \continuous-authentication". The authenticator provides as nal feedback a con dence level, which allows for \multi-level-unlocking". Intentional disclosure of authentication credentials would pose the same di culties as a replay attack, and therefore the scheme o ers \non-disclosability".

The results are summarised in table 6.1. Column properties are highlighted to facilitate the comparison with the Picosiblings solution [3].

The proposed solution does not completely dominate Picosiblings. This is only because the scheme is not \unlinkable". It performs better by o ering \nothing-to-carry" and quasi-o ering \easy-recovery-from-loss". The prototype also has a \negligible-cost-per-user", which is something Picosiblings do not aim to achieve. In terms of security

[3] The colours have the following meanings based on the result: green - o ered, red - not o ered, and yellow - quasi o ered.

	Property	Picosiblings	Proposed scheme
Usability	Memorywise-e ortless	O ered	O ered
	Scalable-for-users	O ered	O ered
	Nothing-to-carry	Not-o ered	Not-o ered
	Physically-e ortless	O ered	O ered
	Easy-to-learn	Not-o ered	O ered
	E cient-to-use	Quasi-o ered	Quasi-o ered
	Infrequent-errors	Quasi-o ered	Quasi-o ered
	Easy-recovery-from-loss	Not-o ered	Not-o ered
Deployability	Accessible	Not-o ered	Not-o ered
	Negligible-cost-per-user	Not-o ered	Not-o ered
	Server-compatible	Not-o ered	Not-o ered
	Browser-compatible	Not-o ered	Not-o ered
	Mature	Not-o ered	Not-o ered
	Non-proprietary	O ered	O ered
Security	Resilient-to-physical-observations	O ered	O ered
	Resilient-to-targeted-impersonation	O ered	O ered
	Resilient-to-throttled-guessing	O ered	O ered
	Resilient-to-unthrottled-guessing	O ered	O ered
	Resilient-to-internal-observaions	O ered	O ered
	Resilient-to-leaks-from-other-veri ers	O ered	O ered
	Resilient-to-phising	O ered	O ered
	Resilient-to-theft	Quasi-o ered	O ered
	No-trusted-third-party	O ered	O ered
	Requiring-explicit-consent	O ered	O ered
	Unlinkable	O ered	Not-o ered

TABLE 6.2: UDS framework assessment.

it is also better by o ering the \resilient-to-theft", \multi-level-unlocking", and \non-disclosability properties.

In conclusion, we achieve our proposed goal of providing a solution that is better than Picosiblings in at least one property.

6.4 UDS framework evaluation

We now perform the reassessment of a Pico that uses our proposed token unlocking mechanism. The evaluation is performed using the UDS framework developed by Bonneau et al [5]. We will compare the result with the original Pico assessment in order to check for improvements. A summary is presented in table 6.2.

The UDS framework assessment shows similar results to the token unlocking framework. By using the new scheme, Pico achieves an overall better score. It now o ers \easy-to-learn", as it no longer requires Picosibling secret share management. In the lack of a cost analysis, we will consider that even with the new scheme the \negligible-cost-per-user" property is not o ered. By relying on more than auxiliary devices, a Pico that uses the proposed scheme does o er \resilient-to-theft".

The only property where Picosiblings outperforms the scheme presented in this dissertation is \unlinkable". Unfortunately, this trade-o cannot be xed as the mechanisms combined in the scheme need to rely on biometrics and behavioural analysis, which are unique for each individual.

In conclusion, by using the new token unlocking mechanism, overall Pico would improve 2 properties in exchange for one.

6.5 Future work

The token unlocking mechanism proposed in this project o ers a new perspective to Pico unlocking. The assessment presented in this chapter shows that it o ers a reliable alternative to Picosiblings. However, improvements can be made both to its design and implementation.

The Android prototype was developed as a proof of concept. Further experiments need to be performed using di erent weights and decay functions. An user study is required in order to determine the acceptable time interval between consecutive explicit authentication requests, and the implementation needs to be adapted accordingly. Furthermore, explicit authentication mechanisms need to be developed for the Android prototype.

The set of individual mechanisms used with the scheme's prototype can be improved. Better biometric libraries should be either developed or imported in order to increase the accuracy of the implementation. Furthermore, additional mechanisms should be developed for the platform. A number of viable suggestions are made in appendix C.

With the current prototype, the voice and face recognition mechanisms sample data at xed time intervals. This should be change by taking advantage of user behaviour and Android events. Examples for this were given in section 5.1.

The face recognition mechanism can be improved either by introducing another library that performs face detection, or by using a di erent face recognition library that o ers both features. Cryptographic support needs to be added for this mechanism. It can be performed through additional modi cations of the \Javafaces" library that would allow it to use raw data during the training process.

A safer prototype would be to develop a root system service using the Android NDK C compiler. The binary has to be included in the system partition of the boot image in order to be accessible by the \init" process during start up. The \init.rc" con guration le used by \init" also needs to be con gured to start the service. This implementation requires modi cations to the \/system" partition. The process does not resume to simply gaining root privileges. The root directory \/." is mounted as ramdisk, and therefore any modi cations will be reverted once the device is rebooted. In order to make persistent changes, the user needs to modify the boot image, and re- ash it on the device.

6.6 Conclusions

The purpose of this dissertation was to create a new scheme for unlocking the Pico token. We have adapted the UDS framework developed by Bonneau et al [5] to create a token unlocking framework. Both frameworks were used in evaluating the solution proposed in this dissertation. Results have shown that although the new scheme does not completely outperform Picosiblings, they o er a larger number of bene ts. The new token unlocking mechanism was prototyped on an Android smart phone, proving that the design can be implemented using existing hardware.

Appendix A

Android development and security

To gain a better understanding of di erent design decisions and limitations of our implementation, we will present a brief literature review of the Android development platform. Mechanisms and components will be described with an emphasis on security.

William Enck et al [24] o er a good introduction to Android application development. They focus on the security aspects of the development platform. It is a relatively old paper (2008), from the same year of the Android initial release. However, the fundamental design principles and security concepts that are discussed did not change considerably. The platform's open standards were made public in November 2007. This allowed researchers such as the authors of this paper to perform a pre-release analysis of the system.

Android uses as a core operating system a port of the Linux kernel. This introduces to the platform some of the Linux security mechanisms (i.e. le permissions, access control policies). On top of the kernel there is an application middleware layer composed out of the Java Dalvik virtual machine, core Java application libraries, as well as libraries which o er support for storage, sensors, display, and other device features. Applications are supported by the middleware and developed using the Android Java SDK.

The Android development model is based on building an application from multiple components. Depending on their purpose, the SDK de nes four types: activity, service,

content provider, and broadcast receiver. For the purpose of brevity we will not discuss each individual component[1]. To allow meaningful interaction, Inter Component Communication (ICC) is enabled using special objects called Intents.

The application we are developing needs to perform most of its processing in the background. It does not require any explicit user interaction. According to the Android model, this should be achieved using Services. To enable convenient component interaction, services can be bound engaging in a client-server communication. An important note made in the paper is that while a Service is bound, it cannot be terminated by an explicit stop action. According to the Android development API guide [2] there are two independent scenarios describing the lifetime of a bound service:

1. If the service was not previously running, and a \bindService()" command is issued by a component, the service is kept alive for as long as clients are still bound. A client becomes unbound by calling \unbindService()".

2. If the service is started using \onStartCommand()" it can only be stopped if it has no bound clients and an explicit request is made either via \stopSelf()" or \stopService". Unlike the previous case, its lifetime persists even with no bound components.

The paper discusses two types of Android security enforcements: ICC, and system level. System level security is based on the Linux permission model. When installed, each app is allocated an UID and GID. This allows internal storage access control restrictions, keeping application data sandboxed from other apps.

ICC security is the main focus of the paper. Intent communication is based on commands sent to the \/dev/binder" device node. The node needs to be world readable and writeable by any application. Therefore, Android cannot mediate ICC using the Linux permissions model. Security relies on a Mandatory Access Control (MAC) framework enforced by a reference monitor. This protection is implemented by the driver responsible for processing IOCTL calls for the \/dev/binder" node.

[1] More details on the role of each component can be found on the Android website: http://developer.android.com/guide/components/fundamentals.html

[2] http://developer.android.com/guide/components/bound-services.html

During development, each application needs to de ne a manifest le[3]. Some of the security con gurations de ned in this le are: declared components and their capabilities, permissions required by the app, and permissions other apps need to have in order to interact with app components. These entries are used as labels for the MAC framework.

Using the app manifest le, each component can be de ned as either public or private. This re nement is con gured by the \exported" eld. It de nes whether or not another application may launch or interact with one of its components. When this paper was written, the \exported" eld was defaulted to \true". However, as shown by Ste en and Mathias [25] in 2013, starting with Android 4.2 the default of this value was changed to \false", and now conforms to the \principle of least privilege".

Components listening for Intents need to have an intent- lter registered in the application manifest le. This allows them to export only a limited set of intents to other applications. Further restrictions to Intent objects are o ered by the SDK using permission labels. This mechanism provides runtime security checks for the application. It is an additional prevention mechanism for data leaks through ICC. An application may broadcast an event throughout the system. By using permission labels, only apps that have the respective permission may process the event. Furthermore, Services may check for permissions when they are bound by another component. This allows them to expose di erent APIs depending on the binder.

Ste en and Mathias [25] focus on deeper issues of the Android platform. They show how problems are solved from one Android version to the other. Unfortunately, OEMs tend not to update the software of their devices once they have shipped, which creates a high security risk.

The starting point of understanding Android security is learning how it is bootstrapped during the ve step booting process:

1. Initial bootloader (IBL) is loaded from ROM.
2. IBL checks the signature of the bootloader (BL) and loads it into RAM.
3. BL checks the signature of the linux kernel (LK) and loads it into RAM.

[3]Full details regarding the manifest le can be found on the Android website: http://developer.android.com/guide/topics/manifest/manifest-intro.html

4. LK initialises all existing hardware and starts the linux \init" process.

5. The init process reads a con guration le and boots the rest of LA.

The android security model is split by the paper in two categories: system security, and application security.

Android provides a keychain API used for storing sensitive material such as certi cates and other credentials. These are encrypted using a master key, which is stored using AES encryption. Security needs to begin somewhere. An assumption has to be made about a state being secure from which multiple security extensions can be made. In this case, the master key is considered to be that point of security. However, given a rooted device, the master key itself can be retrieved from the system and therefore compromising all other credentials. The Android base system (libraries, app framework, and app runtime) is located in the \system" partition. Although it is writeable only by the root user, as mentioned before, exploits which grant this privilege exist.

From the user's perspective, an interesting \feature" which may a ect the ow of information within Android is the fact that applications from the same author may share private resources. When installing an app the user needs to accept its prede ned set of permissions. Due to resource sharing, a situation may present itself where an application that has permissions for the owner's contacts may communicate with an application that has permissions for internet in order to leak con dential data. A developer may therefore construct pairs of legitimate applications in order to mask a data ow attack.

The Android OS o ers a number of memory corruption mitigations in order to avoid bu er over ow attacks, or return oriented programming. The following list presents these low level security mechanisms:

- Implements mmap_min_addr which restricts mmap memory mapping calls. This prevents NULL pointer related attacks.
- Implements XN (execute never) bit to mark memory as non-executable. The mechanism prevents attackers from executing remote code passed as data.
- Address space layout randomisation(ASLR) was implemented starting with Android 4.0. This is a rst step to preventing return oriented programming attacks.

The memory location of the binary library itself is however static. After a number of attempts using trial and error, the attacker may succeed using return oriented programming.

- Position independent and randomised linker (PIE) is implemented starting with Android 4.1 to support ASLP. This makes the memory location of binary libraries to be randomised.
- Read only relocation and immediate binding space (RELro) was implemented starting with Android 4.1. It solves an ASLR issue where an attacker could modify the global o set table (GOT) used when resolving a function from a dynamically linked library. Before this update an attacker may insert his own code to be executed using the GOT table.

A number of application security mechanisms are in place to make Android a safer environment for its users. A device program also known as the \Bouncer" prevents malware to be distributed from the Android App store (Google Play). The purpose of the bouncer is to verify apps prior to installation by checking for malware signatures and patterns.

Secure USB debugging was introduced starting with Android 4.4.2. This only allows hosts registered with the device to have USB debugging permissions. The mechanism is circumvented if the user does not have a screen lock.

According to the paper, the Android OS is responsible for 96% of mobile phone malware. The authors claim that this is the case due to 4 big issues of the Android platform:

1. Security updates are delayed or never deployed. This is due to a number of approvals that an update needs to receive prior to deployment. This introduces an additional cost to the manufacturer (OEM), that does not generate any revenue. The majority of teams working on the Android platform are focusing on current releases. In most cases there are simply not enough resources to merge Google security updates to the OEM repository. Furthermore, the consequences of a failed OS update may cause \bricking" of the device, which is a huge risk for the manufacturer. All these issues lead to very few security updates. Therefore, important features such as RELro are never deployed, making older Android releases vulnerable.

2. OEMs weaken the security of Android by introducing custom modi cations before they roll out a device.

3. The Android permission model is defective. According to Kelley et al [26], most users do not understand the permission dialogue when installing an application. Furthermore, even if they could understand the dialogue, most of the time it is ignored in order to use the exciting new app. According to the same study, most applications are over-privileged. This is due to developers not understanding what each privilege grants. Furthermore, as previously pointed out, apps developed by the same owner may share resources and implicitly privileges.

4. Google Play has a low barrier for malware. A developer distribution agreement (DDA) and a developer program policy (DPP) need to be agreed to and signed by the developer before submitting the application to the Android market. However, Google Play does not check upfront if an application adheres to DDA and DDP. The application is only reviewed if it becomes suspect of breaking the agreements. Furthermore, according to [27] there are ways of circumventing the Bouncer program[4].

We have brie y presented the Android development model, existing mechanisms, and the security of the platform. This information should be su cient to understand the principles involved in the design of the prototype developed for this dissertation project.

[4] An example of such an application is presented in an article written in Tech Republic: http://www.techrepublic.com/blog/google-in-the-enterprise/malware-in-the-google-play-store-enemy-inside-the-gates/# (visited on 29.05.2014).

Appendix B

Token Unlocking Framework evaluation examples

The following sections present examples of how the token unlocking framework should be used. We will be assessing PINs, and biometric face unlock. Together with the Picosiblings evaluation in section 3.3, each scheme represents a di erent type of authentication method. Picosiblings essentially are a secret the owner has, PINs are a secret the owner knows, and Face-unlock re ects who the owner is.

B.1 PIN

PINs are token authentication mechanisms similar to passwords. The di erence between the two is that they use a smaller set of input characters. Additional protection comes from steep security measures when the authentication challenge has failed. As an example, typing 3 wrong PINs on a mobile phone would lock the owner's SIM card. A lot of the PIN properties should however be similar with those o ered by passwords.

The scheme relies on knowing a secret, which is not \memorywise-e ortless". It does however o er the \nothing-to-carry" property. Because of its similarity with passwords users nd it \easy-to-learn". The small character set allows for fast user input and validation making PINs \e cient-to-use". Mistakes however may still occasionally occur, and due to the lack of visual feedback [1] the scheme only quasi-o ers \infrequent-errors".

[1] If existent, visual feedback for PINs generally consists of `*' characters.

PINs are generally easily reset by the manufacturer using online services, therefore having \easy-recovery-from-loss" [2]. The scheme o ers the \availability" property, as the authentication process cannot be impaired by external factors.

Just as passwords PINs score all points in deployability. They can be used regardless of disabilities, making them \accessible". They have virtually no cost, satisfying the \negligible-cost-per-user" property. Being a subset of passwords, we consider the mechanism to be \mature" and \non-proprietary".

From a security perspective PINs score poorly. They are not \resilient-to-physical-observation". Anyone can eavesdrop the input of a PIN either by shoulder sur ng or recording with a camera. Just as passwords, PINs are often written down in plain sight. However, in the lack of relevant studies[3] we will mark the scheme to quasi-o er \resilient-to-targeted-impersonation". The restricted character set makes PINs adopt harsher security policies when provided invalid input. They are generally locked after three bad attempts, making them \resilient-to-throttled-guessing". The \resilient-to-unthrottled-guessing" property is implementation dependent. However, security tokens are dedicated devices that generally have tamper resistant memory, making unthrottled guessing not possible. Any hardware PINs may require does not compromise the mechanism, therefore o ering \resilient-to-theft". Users have the freedom of choosing any PIN. Even in situations when reused with multiple tokens, credentials are generally salted and therefore \unlinkable". The scheme does not o er \continuous-authentication" because the process is not e ortless for the user. They can only provide locked or unlocked feedback, and therefore do not o er \multi-level-unlocking". The owner may disclose their PIN at any time, making the \non-disclosability" property unsatis ed.

B.2 Face unlock

Although not currently used as a security token unlocking mechanism, face recognition is a viable biometric authentication scheme. It can be ported for a token such as Pico, which is designed to have a camera. With a variety of possible implementations, for accessibility reasons we will analyse the Android face unlocking mechanism.

[2] An example of this is the RSA SecurID. An example reset procedure is described at the following link: http://uk.emc.com/collateral/15-min-guide/h12278-am8-help-desk-administrator-guide.pdf

[3] Just as Bonneau et al suggest [5], a relevant study would assess acquaintances' ability to guess the PIN of a subject.

Face unlock is \memorywise-e ortless", as any other biometric scheme. It o ers the \nothing-to-carry property", the camera being embedded as part of the token. The mechanism is \easy-to-learn", since it only needs the user to look at the camera. The authentication process is performed almost instantly, making the scheme \e cient-to-use". The scheme is dependent on camera positioning, obstructing objects (e.g. glasses, earrings), and face mimic. In conjunction with the UDS framework assessment of biometrics in general, the scheme does not o er \infrequent-errors". If the scheme no longer functions as a result of change in facial traits, Android has a backup unlocking mechanism. This may also be used to disable or recalibrate the scheme, therefore offering \easy-recovery-from-loss". The \availability" property is not satis ed due to the dependence on external factors such as light or obstacles.

Android face recognition is \accessible" for anyone regardless of disabilities. It o ers the \negligible-cost-per-user" property, given that the hardware was already present in devices without face recognition features. Due to limited user exposure it is only quasi-\mature". On Android, the scheme is implemented as not \non-proprietary".

Observing the owner authenticate does not provide any advantage to an attacker. It therefore o ers the \resilient-to-physical-observations" property. Targeted impersonation is an issue with any biometric mechanism. The scheme is vulnerable to replay attacks (i.e. a picture of the owner's face) and therefore does not o er \resilient-to-targeted-impersonation". The "resilient-to-throttled-guessing\ and \resilient-to-unthrottled-guessing" properties do not apply. Given the Android implementation, neither does \resilient-to-theft". The same authentication data is used with any veri er, and therefore the \unlinkable" property is not o ered. The scheme is implemented without \continuous-authentication" or \multi-level-unlocking" although both can be supported by biometric mechanisms. Given the possibility of deliberately providing data for a replay attack, the scheme only quasi-o ers the \non-disclosability" property.

Appendix C

Examples of supported Android authentication mechanisms

Android provides an extensive sensor API that can support the token unlocking scheme proposed in section 4.2. This can be used to develop a number of continuous authentication mechanisms. We have listed the following non-exhaustive set of examples:

Face recognition

The mechanism is based on capturing an image of the user's face and performing face recognition. Sampling valid face images can be performed without explicit requests by predicting user behaviour. We will use as an example an user that owns a phone with a front-facing camera. When the owner is unlocking the phone, there is a high probability that they will be looking towards the screen. This provides a good opportunity for the face recognition service to capture a valid sample. Using the Android API, this can be achieved by registering a \BroadcastReceiver" to listen for the one of the following events: ACTION_SCREEN_ON, ACTION_SCREEN_OFF, or ACTION_USER_PRESENT. The mechanism may continue to perform face recognition based on collected data and a previously recorded sample of the owner. A simple face recognition mechanism was also implemented as part of the prototype.

Voice recognition

A voice recognition mechanism can record data either periodically, or based on Android events. It may then perform voice recognition and provide a con dence

level of the owner being present. Voice sampling does not necessarily imply a voice password. An analysis can be performed using feature extraction. This facilitates the sampling process, which may be performed at any time. With a frequent sampling period, the owner of the device is likely to be recorded while speaking, which would provide a valid data sample. For even better con dence the mechanism can be implemented to start recording when a call is either made or received. On Android this can be achieved by listening for a PHONE_STATE event. A simple voice recognition mechanism was implemented as part of the prototype.

Iris scanning

Similar to face recognition, this can be implemented by taking advantage of user behaviour while using the phone. When the phone is unlocked, the user is very likely to face the front camera, allowing for a good capture. The only problem with this mechanism is the quality of pictures o ered by most phones. If the sampling quality is not su ciently good, meaningful features from the iris may not be extracted. This would make the con dence level of the mechanism relatively low, but may change in the future as devices become increasingly performant.

Keystroke analysis

This mechanism was inspired from a paper by Clarke et al [22]. The principle of keystroke analysis is based on the patterns in which the user types on his mobile phone. Di erent features can be extracted here, such as: letter sequence timings, words per minute, letters per minute, frequent used words, and others. Using this data a con dence level can be generated.

This mechanism is harder to implement using solely the Android SDK. A good starting point would be to have a keyboard app developed for the user that also communicates with the authentication mechanism. If the keyboard is disabled by an attacker this should be considered, especially if the authenticator was originally con gured to listen for input.

Gait recognition

This mechanism is based on analysing individual walking patterns. According to data presented by Derawi et al [18], error rates[1] may vary between 5% to

[1] The performance indicator used in biometric analysis is the Equal Error Rate (EER).

20%. Android o ers native recognition support for walking, driving, or standing still. Applications can register a sensor callback for the TYPE_STEP_DETECTOR composite sensor. Whenever such an event is detected, data can be recorded from the accelerometer and validated using an algorithm similar to the one described by Derawi et al [18].

Ear shape analysis

Research shows (i.e. Burge et al [28], Mu et al [29]) that the shape of the human ear contains enough unique features to perform biometric authentication. Taking advantage of user behaviour, valid data can be captured and analysed using a smart phone. We suggest that a picture is taken a few seconds after a phone call event is detected. If no peripherals are attached, the user is likely to move the device towards the ear. Images captured by such a mechanism could then be used to calculate an accurate con dence level of the user's identity. This method was not tested, so therefore we cannot ensure whether the auto-focus of the camera is su ciently fast to obtain a valid image.

Proximity devices

This is an original idea based on providing a con dence level depending on the presence of known devices. The mechanism should connect with other devices that are also running the authenticator. The two owners don't necessarily need to know one another for the acknowledgement to be performed. Whether regular travel schedules, or working in an o ce, users are constantly being in the presence of other known devices. This should provide a con dence as to whether the device is in the presence of its owner.

The authentication works by seeking connections with other devices. Whenever a device is identi ed, its ID is recorded. The mechanism needs to keep track of the number of times it has connected with another device. Some connections may be established for the rst time, and should not bring any con dence. Other connections, such as the Pico of a co-worker, would probably have a high number of connections, and therefore the mechanism should output a higher con dence level in its presence. This mechanism is similar to the Picosiblings solution, but with no k-out-of-n secrets. Each Pico is essentially a Picosibling for another Pico, with each device having a di erent weight based on familiarity.

As an example, when travelling with your family on holiday most of the devices there are unknown. However, given that a number of frequent IDs are in the proximity of the authenticator, the mechanism should still consider to some extent that it is in the possession of its owner.

The mechanism can be circumvented in the scenario where co-workers or friends try to unlock the Pico. Due to this downside, it should never have su cient weight to unlock the token on its own. However, in combination with other mechanisms it would provide a good approximation of whether it is in the possession of its owner. If the device is in good company there is a good chance the owner is also present.

Location data

This mechanism is similar to \Proximity devices" and much easier to implement. Based on Android GPS and network location data, the phone may detect whether it is in an usual location or not. Just as \Proximity devices" this should not carry a high weight in the scheme, especially since it would not provide accurate results in scenarios such as holidays.

Service utilisation

This mechanism exploits patterns in the Android phone's service and app utilisation. Based on current running applications, services, and the time they were started we may create a model where some con dence is given regarding the ownership of the device. This mechanism would only be e ective in detecting sudden changes. It would have a low weight in the overall scheme due to its lack in precision.

Picosiblings The original Picosiblings mechanism may also be used with this scheme. Although not part of the standard set of Android device sensors, if available, a Picosiblings implementation may be included as one of the authentication mechanisms.

A number of continuous authentication mechanisms may also be used for explicit authentication. The user can be noti ed to provide accurate information for the following mechanisms: face recognition, voice recognition, iris scanning, keystroke analysis, gait recognition, and ear shape analysis. This creates the opportunity for a valid data sample to be collected.

A number of explicit authentication mechanisms which do not satisfy the continuous authentication property of Pico may be implemented for the Android platform. It is important to note that additional mechanisms not included in this list need to satisfy the memorywise-e ortless property of the token unlocking framework (3.2). We suggest the following mechanisms for implementation:

Fingerprint scanner

Devices that incorporate a ngerprint scanner (such as the IPhone 5S) can use the sensor as an explicit authentication mechanism. It cannot be used for continuous authentication because the user doesn't come in contact with the sensors on a regular basis. A mechanism can therefore request explicit ngerprint data, which would then be compared with the owner's biometric model, outputting a con dence for the authentication. The result will be combined in the overall scheme just as any other mechanism. The the only di erence will be in terms of weight and decay rate.

Hand writing recognition

The user may be prompted to use the touch screen in order to write a word of his choice. This would guarantee the memorywise-e ortless property because the user doesn't need to remember any secret. The handwriting would be analysed with a precon gured set of handwriting samples in order to compute the con dence level that the owner produced the input.

Lip movement analysis

According to Faraj and Bigun [30], analysing lip movement while speaking can be used for authentication. The user would be prompted to provide a data sample such as reading a word provided by the authenticator. Using lip movement authentication, a quanti able con dence level would be produced. This mechanism can also be implemented as a continuous authentication mechanism. However, data sampling would likely have a low success rate as users tend not to have their mouth within the camera's eld of view.

Bibliography

[1] Robert Morris and Ken Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594{597, 1979.

[2] Philippe Oechslin. Making a faster cryptanalytic time-memory trade-o . In *Advances in Cryptology-CRYPTO 2003*, pages 617{630. Springer, 2003.

[3] Je Jianxin Yan, Alan F Blackwell, Ross J Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security & privacy*, 2(5):25{31, 2004.

[4] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40{46, 1999.

[5] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 553{567. IEEE, 2012.

[6] Nathan L Clarke, Steven M Furnell, Phihp M Rodwell, and Paul L. Reynolds. Acceptance of subscriber authentication methods for mobile telephony devices. *Computers & Security*, 21(3):220{228, 2002.

[7] Frank Stajano. Pico: No more passwords! In *Security Protocols XIX*, pages 49{81. Springer, 2011.

[8] Oliver Stannard and Frank Stajano. Am i in good company? a privacy-protecting protocol for cooperating ubiquitous computing devices. In *Security Protocols XX*, pages 223{230. Springer, 2012.

[9] Dinei Florencio and Cormac Herley. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web*, pages 657{666. ACM, 2007.

[10] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612{613, 1979.

[11] Frank Stajano. The resurrecting duckling. In *Security Protocols*, pages 183{194. Springer, 2000.

[12] Hugh Wimberly and Lorie M Liebrock. Using ngerprint authentication to reduce system security: An empirical study. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 32{46. IEEE, 2011.

[13] Nathan L Clarke and Steven M Furnell. Authentication of users on mobile telephones{a survey of attitudes and practices. *Computers & Security*, 24(7):519{527, 2005.

[14] Nathan L Clarke, SM Furnell, Benn M. Lines, and Paul L Reynolds. Using keystroke analysis as a mechanism for subscriber authentication on mobile handsets. In *Security and Privacy in the Age of Uncertainty*, pages 97{108. Springer, 2003.

[15] Gregory D Williamson and GE Money-America's. *Enhanced authentication in online banking*. PhD thesis, Utica College, 2006.

[16] Elena Vildjiounaite, S-M Makela, Mikko Lindholm, Vesa Kyllonen, and Heikki Ailisto. Increasing security of mobile devices by decreasing user e ort in veri - cation. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, pages 80{80. IEEE, 2007.

[17] Liang Cai, Sridhar Machiraju, and Hao Chen. Defending against sensor-sni ng attacks on mobile phones. In *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, pages 31{36. ACM, 2009.

[18] Mohammad Omar Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on*, pages 306{311. IEEE, 2010.

[19] Ji Ming, Timothy J Hazen, James R Glass, and Douglas A Reynolds. Robust speaker recognition in noisy conditions. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(5):1711{1723, 2007.

[20] Lawrence George Kersta. Voiceprint identi cation. *The Journal of the Acoustical Society of America*, 34(5):725{725, 2005.

[21] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586{591. IEEE, 1991.

[22] Nathan L Clarke and SM Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1{14, 2007.

[23] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[24] William Enck, Machigar Ongtang, Patrick Drew McDaniel, et al. Understanding android security. *IEEE Security & Privacy*, 7(1):50{57, 2009.

[25] Ste en Liebergeld and Matthias Lange. Android security, pitfalls and lessons learned. In *Information Sciences and Systems 2013*, pages 409{417. Springer, 2013.

[26] Patrick Gage Kelley, Sunny Consolvo, Lorrie Faith Cranor, Jaeyeon Jung, Norman Sadeh, and David Wetherall. A conundrum of permissions: Installing applications on an android smartphone. In *Financial Cryptography and Data Security*, pages 68{79. Springer, 2012.

[27] NJ Percoco and S Schulte. Adventures in bouncerland: Failures of automated malware detection within mobile application markets. *Black Hat USA 2012*, 2012.

[28] Mark Burge and Wilhelm Burger. Ear biometrics. In *Biometrics*, pages 273{285. Springer, 1996.

[29] Zhichun Mu, Li Yuan, Zhengguang Xu, Dechun Xi, and Shuai Qi. Shape and structural feature based ear recognition. In *Advances in biometric person authentication*, pages 663{670. Springer, 2005.

[30] Maycel Isaac Faraj and Josef Bigun. Motion features from lip movement for person authentication. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1059{1062. IEEE, 2006.