University of Cambridge

Master's Thesis

User Authentication for Pico: When to unlock a security token

Author:
Cristian M. Toader

Supervisor:
Doctor 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

May 2014

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

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

Chapter 1

Introduction

Passwords are currently the most widely used electronic authentication mechanism. They rely on remembering a secret sequence of characters, and providing it as input for the authentication process. This originally offered a sufficiently secure authentication mechanism. However, as shown by Adams & Sasse [1], the fundamental concept of remembering a secret, makes passwords unsuited for the current technological context.

As Robert Morris [2] emphasises in his paper, there is a constant competition between attackers and security experts. The majority of users try to maximise usability of their passwords by choosing secrets which are easy to remember. These however are susceptible to a number of attacks such as brute force, dictionary, pre-compiled hashes, rainbow tables [3], and others. Security experts were able to slow down the attackers in the past without any impact to the user, but with a constant increase in computational power, passwords became easier to breach.

The main flaw of passwords is that when chosen freely they tend to be short and predictable. In order to maintain acceptable security guarantees, a number of requirements in their creation started to be enforced. Some password mechanisms may require 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 on a regular basis with something not too similar with the original.

As shown by Yan et al [4], without any additional advice to make the password more memorable, users choose weaker passwords. From a theoretical perspective, additional restrictions would make the mechanism more secure. The users are forced to pick a non-intuitive password, and fully utilise the available character set. This makes dictionary and brute force attacks harder to perform. In practice however, users need to memorise numerous, unique, and complex passwords. As shown by Adams & Sasse [1] maintaining all restrictions and security advices proves not to be feasible, leading to poor practices such as writing the passwords down.

The main problem with passwords is the basic principle of users remembering a secret. If the secret is memorable, than 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 solution is not scalable. For all these fundamental reasons, passwords prove not to be a reliable solution for the future and even present.

A large number of alternative 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 ignore the existence of available solutions. The main conclusion we may draw 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 which generates and manages user authentication credentials. This transforms the problem of knowing a secret into having it. Since anyone in possession of such a hardware token would have access to the owner's accounts, this type of authentication is not very secure. Therefore, Pico adds an additional layer of security by being usable only in the presence of its owner. In a sense a security chain is created where \who you are" unlocks \a secret you have" which is used for authentication.

In order to identify the presence of its owner, Pico communicates with small devices called Picosiblings [8]. These devices are embedded in everyday items that the user carries throughout the day (i.e. keys, necklace, rings). Each Picosibling transmits a secret sequence to the Pico. When all required secrets are gathered, Pico becomes unlocked and may be used by its owner.

Picosiblings are a sensible solution to unlocking Pico. However, they are purely based on proximity to the device. As suggested in the original Pico paper [7] anyone in possession of both Pico and the Picosiblings would have full access to the owner's accounts for a limited amount of time. Some additional security features are included, such as having a remote online server as a Picosibling. However, the main downside of this approach is the fact that Picosiblings do not reflect who the user is, but rather additional things the user has.

The purpose of this project is to design and implement 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 fit these requirements are biometric authentication mechanisms. For the purpose of this project we have explored the possibility of combining multiple biometrics and behavioural analysis as part of an unified solution. The output from each mechanism is combined to generate an overall confidence level, reflecting that the owner is still in possession of the Pico.

We will explore and evaluate the original Picosiblings solution as well as other token unlocking schemes. The evaluation will be performed using a framework derived from the work of Bonneau et al [5]. This will enable a formal analysis of the benefits and downsides of the new authentication scheme in comparison with existing mechanisms.

Contribution

In the process of designing and developing a new token unlocking mechanism, a number of contributions have been made. The following list presents a summary of these achievements, with further details in the following chapters.

- We create a framework derived from the work by Bonneau et al [5]. This is used to evaluate a few existing token unlocking mechanisms, including Picosiblings. The data is then used as a benchmark when evaluating the proposed solution.
- We design a new token unlocking mechanism. Although the solution may be used in any type of user authentication, it is presented in the context of unlocking the

Pico token. The design is analysed using the token unlocking evaluation framework. A comparison is made with the original Picosiblings solution. The aim of the dissertation is for the new scheme to achieve better results in at least some categories of the token unlocking framework.

- We develop an Android prototype. The implementation is meant to prove that the design is feasible for implementation using existing technologies. The prototype was not developed for performance purposes. However, power analyses as well as timings of di erent stages of the scheme were recorded to serve as an approximation of the limitations and downsides of the scheme.
- We analyse and determine the impact of the proposed token unlocking mechanism on the Pico. The analysis is performed based on the original framework by Bonneau et al [5]. One of the proposed goals when designing the solution was to make Pico better in terms of at least one property.

Chapter 2

Pico: no more passwords!

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

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

The motivation behind this project 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. As shown by Adams & Sasse [1], this has a negative impact on the users, which have limited memorising capability.

Another reason why passwords are no longer viable is the fact that they are not a scalable solution. Security experts recommend that passwords should be reused for multiple accounts. However, a large number of computer based services require password 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] performed over half a million users confirms the negative impact of scalability on password quality. Furthermore, passwords are often forgotten or reused across accounts.

When designing the Pico password replacement mechanism, Stajano decides to have a fresh start. He describes that an alternative for passwords needs to be at least memory-less and scalable, without reducing security. In the case of token based authentication, the solution also needs to be loss and theft resistant. The Pico token was therefore designed to satisfy these fundamental properties, as well as other bene ts emphasised in a paper by Bonneau et al [5] as well as the the original work by Stajano.

As a token authentication mechanism, Pico transforms \something you know" into \something you have". It o ers support for thousands of credentials which 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 credentials 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. 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” just as 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. It contains thousands of slots used for storing unique credentials used in the authentication process. 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 creating the account.

During account creation Pico scans a 2D visual code generated by an app. The image encodes a hash of the apps certi cate including the app name and public key. Pico starts the protocol with the app using the radio channel, and the app provides 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, and provides a temporary public key. This protects the identity of the owner, by only showing their public key after the app is authenticated. Only then Pico generates a key pair, sends the public key to the app and stores the key pair.

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 which we have not yet fully described. The token should become unlocked only in the presence of its owner. Currently this is achieved 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, rings, keys, chains, etc.

The Pico authentication credentials are encrypted using the Pico Master Key. The key is not available on the Pico 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 share is held by a Picosibling.

Using an initialisation protocol based on the resurrecting duckling [11], each Picosibling is securely paired with Pico. After the initialisation process, Pico sends periodic ping requests to which all registered Picosibling are expected to respond. During each successful ping, the Picosibling sends its k-out-of-n share back to Pico. If enough secrets are provided 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. The countdown value is decreased periodically. When it 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 the Pico.
- Remote server network connection used for locking the 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 cheaper alternative to prototype and test, which is something we will make use of in this project.

Chapter 3

Assessment framework

The purpose of this chapter is to create an assessment framework for token unlocking mechanisms. This framework will be used to evaluate existing solutions, including the Picosiblings scheme used by Pico. The analysis of the results can then be used to create an alternative unlocking mechanism for Pico. The project aims to achieve better results in some categories, without necessarily completely outperforming it.

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, the assessment scheme is not entirely compatible for token unlocking schemes. For example, properties such as \Browser-compatible" do not apply, while others need to be rede ned to t our context. The paper however presents a good starting point for our token unlocking evaluation framework. The remainder of this section will present a brief summary of the paper.

The motivation behind this paper is to gain insight about the di culty of replacing passwords. An assessment framework is created, and a number of web authentication mechanisms are evaluated. It is an 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. For this reason, it is abbreviated by the authors as the \UDS framework". An authentication scheme is evaluated by assessing whether each property is o ered or not. In the case where a scheme almost o ers a property, the authors mark it as quasi-o ered. To simplify the framework, properties which are not applicable are marked as o ered.

Since passwords are currently the most widely used authentication mechanism, the results are predictable. Evaluating 35 replacement schemes shows that no scheme completely dominates them. Passwords 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. The full evaluation can be found within the paper itself.

Biometric mechanisms receive mixed scores on usability. None of them o er the \infrequent-errors" property, due to false negative precision. More importantly if biometric data becomes compromised, the possibility of replay attacks makes the authentication mechanism unreliable. They score poorly in deployability partially because they require additional hardware. In terms of security they perform worse than passwords. Replay attacks can be performed by an attacker using a 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, therefore the \unlinkable" property is not o ered by these mechanisms.

By analysing the framework results, we see that some authentication schemes, such as security tokens, o er \memory-e ortless" in exchange for the \nothing-to-carry" property. 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" while being \resilient-to-theft".

When trying to compute an aggregate score using the framework, not all properties should be equal in importance. Di erent properties should have di erent weights depending on the purpose of the assessment. For example, if we would try to nd the most

[1]Not applicable to passwords

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 possibility of combining schemes as part of a two factor authentication. In terms of deployability and usability, the overall scheme o ers a property if it 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.

The following section will o er more details on the UDS framework properties which also apply to token unlocking. Further information about the framework are not mentioned in this dissertation for the purpose of brevity. The full list of properties, their description, and the evaluation of a number of mechanisms are provided in the original paper by Bonneau et al.

3.2 Token unlocking framework

Unlike web based authentication mechanisms, token unlocking schemes record and process data locally. For this reason, a subset of the UDS framework properties are also present in the framework we have developed. Those which do not apply, or would apply to every token unlocking mechanism were removed. Some properties needed to be adapted to the new context of a token, and therefore will have a slightly di erent meaning.

The following list contains all properties of the UDS framework developed by Bonneau et al [5] which we adapted and included in the token unlocking framework. A short description is included for the cases when the property is o ered or quasi-o ered, as well as a small example.

Memorywise-effortless

Users do not need to remember any type of secret. This includes passwords, physical signatures, or drawings. The property was originally quasi-o ered if one secret would be used with multiple accounts, but in the case of security tokens this does not apply. 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 of a mechanism that quasi-o ers the property is the Picosiblings scheme which uses small devices embedded in everyday items. Biometric mechanisms that require additional sensors such as a ngerprint reader do not satisfy this property.

Easy-to-learn

Users who use the unlocking mechanism would be able to learn it with ease. The original paper by Bonneau et al [5], assess Pico as not \easy-to-learn" due to the complexity of the Picosiblings management[3]. PINs or passwords however satisfy this property due to the users' familiarity 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 the input for the mechanism. The same applies for setting up the token unlocking mechanism, but with a larger time scale. In the case of PINs for example the input and processing time are very low, and therefore the scheme o ers the property. Mechanisms based on biometrics however may not, depending on the implementation.

Infrequent-errors

The rightful owner should generally be able to successfully authenticate to the token. Any sort of delays resulted from the unlocking mechanism such as typos during typing or biometric false negatives may contribute to the mechanism's inability to provide this property. As an example, PINs have a limited input length

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

[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 user therefore needs to choose the right combination of Picosiblings in order to unlock the Pico, which may prove difficult

and character set size which makes infrequent errors unlikely and therefore offer the property. Biometric mechanisms, based on the type and implementation may quasi-offer the property, although they generally do not.

Easy-recovery-from-loss

The meaning of this property was modified to reflect the context of token unlocking. It is offered if the user may easily recover from the loss of authentication credentials. Depending on the scheme, this may include the loss of auxiliary devices, forgotten credentials, difference in biometric features. As an example, forgotten PINs offer the property as they generally require a simple reset using an online service.

Accessible

The mechanism is usable by any user regardless of any disabilities or physical conditions. In the original paper, passwords are offered as an example of a scheme which offers this property. A gait recognition unlocking scheme would not offer this property.

Negligible-cost-per-user

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

Mature

A large number of users have successfully used the scheme. Any open source projects involving the mechanism, as well as any participation not involving its creators contribute to this property. For example, passwords are widely used and implemented and therefore offer 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 sort of secret.

Resilient-to-physical-observation

An attacker would not be able to impersonate the owner of the token after observing him authenticate. Based on the number of observations required for the attacker to unlock the token, the scheme may quasi-offer the property. The original paper suggests 10-20 times to be su cient, although it is just an approximation. Physical observation attacks are not restricted to shoulder sur ng, and may include video cameras, keystroke sounds, or thermal imaging of the PIN pad.

Resilient-to-targeted-impersonation

An attacker should not be able to impersonate the owner of the token by exploiting knowledge of personal details. This may include birthday, full name, family details, and other sensitive information. The scheme should also be resilient to pre-recordings of biometric information which may then be replayed to the authenticator.

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. As an example, PINs o er this property because 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. Even though the guessing process is only restricted by the attacker's computational power, the scheme would still not be bypassed within reasonable time. The original paper 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 original description will be adapted to a single account. Therefore the property is granted only if an attacker requires more than 2^{40} attempts.

Resilient-to-theft

The property applies to schemes which use additional hardware other than the token. If the additional hardware becomes in the possession of an attacker, it is not su cient to unlock the token. For example, auxiliary biometric devices used in the conjunction with the token o er this property. In this case the token would still not be unlocked using the hardware alone. Picosiblings however only quasi-o er the property. Although they generally rely on proximity to the Pico, the two special shares allow the owner to lock the token remotely.

Unlinkable

Using the authentication input with any verifier using the same authentication mechanism[4] does not compromise the identity of the token owner. As an example the link between a PIN and its owner is not strong enough to make a clear link between the two. However, biometrics are a prime example of schemes which do not offer this property.

We have augmented the subset from the original UDS framework with a number of properties relevant which are relevant to Picosiblings, PINs, as well as other token unlocking mechanisms. The following list is part of the project's contributions to the overall evaluation framework.

Continuous-authentication

The token unlocking scheme re-authenticates the user periodically. The process doesn't need to be hidden from the owner, but it is required to be effortless. The token should remain unlocked and usable in the presence of its owner. The scheme needs to detect when the owner is no longer in possession of the token, and lock the device accordingly. When locked, any open authentication session managed by the security token will 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 benefits of the Pico project. Using the UDS classification of the original framework, the property belongs to the Security category.

Multi-level-unlocking

The unlocking scheme provides quantifiable feedback, not just a locked or unlocked state. The mechanism offers the possibility of supporting multiple token security permissions. These would be granted based on the confidence level that the user trying to unlock the token is its owner. For example, a 70% confidence level that the owner is present may allow the user to access an email account, but not make any sort of payments or banking transactions. Passwords only provide a "yes" or "no" answer and therefore do not offer this property. Biometric mechanisms can offer this property. Their output is either a probability or some sort of distance metric that data was collected from the owner. Different confidence levels could

[4] The authentication mechanism is not necessarily used for token unlocking. Any sort of mechanism which requires user authentication is a valid option.

therefore enable different security permissions. Using the UDS classification 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. However, the focus here is that the token may only be used by its owner. This is an important property in enterprise situations where the security token should not be shared. Passwords and other schemes based on secrets do not offer this property as the owner could share it with another user without any difficulty. Biometric mechanisms however cannot be easily disclosed. Based on the UDS classification 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 such as traffic noise, different light intensities, or restricted movement space. The property is not related to physical disabilities preventing the user from using the scheme but only on contextual influences on data collection. As an example gait recognition would only function while moving on foot and therefore does not offer the property. A mechanism requiring a PIN on the other hand would work in any circumstance. Using the UDS classification of the original framework, the property belongs to the Usability category.

3.3 Example evaluation

We will demonstrate how the framework should be used by assessing three token based authentication mechanisms: Picosiblings, PIN, and Face-unlock. Each scheme represents a different type of authentication method. Picosiblings essentially are a secret the owner has, PINs are a secret the owner knows, and Face-unlock reflects who the owner is. Results in the Picosiblings section will be used in the following chapter as a benchmark for comparison with our proposed token unlocking scheme.

3.3.1 Picosiblings

Unlocking the Pico token requires k-out-of-n secrets used to reconstruct the Pico Master Key. Each Picosibling contains a secret that is transmitted to Pico using a secure connection via a radio channel. Given enough secrets the master key is reconstructed, and Pico becomes unlocked.

The scheme doesn't require from its owner any known secret and therefore is \memorywise-e ortless". Since it relies on devices embedded in everyday items the \nothing-to-carry" property is quasi-satis ed. The original paper by Bonneau marks Pico as not \easy-to-learn" due to Picosiblings management, which is a characteristic of the unlocking mechanism. It is quasi-\e cient-to-use" and has quasi-\infrequent-errors" until proven otherwise. It does not o er the \easy-recovery-from-loss" property. The unlocking mechanism relies only on radio communication with the token. This makes it 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, the 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 since no realistic Picosiblings cost estimate exists 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. For these reasons it is not considered to be \mature". Frank Stajano states in his paper [7] that the design of Pico and Picosiblings are not patented, and no royalties are due. The only requirement for implementing the design is to cite the paper, which makes the unlocking mechanism \non-proprietary".

Since the scheme does not rely on any user input it is \resilient-to-physical-observations". Based on the description of Picosiblings given by Stajano [7] the scheme o ers the \resilient-to-targeted-impersonation", \resilience-to-throttled-guessing", and \resilient-to-unthrottled-guessing" properties. Any attacker which comes in possession of the Picosiblings may unlock the Pico. However due to the auxiliary shared secrets[5] the scheme is quasi \resilient-to-theft". Each Picosibling only works with one veri er (its

[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.

master Pico), and therefore o ers the \unlinkable" property. The scheme was designed to provide \continuous-authentication". Because of the k-out-of-n master key reconstruction mechanism, Picosiblings only have the locked or unlocked states and therefore do not o er \multi-level-unlocking". The scheme does not satisfy the \non-disclosability" property. The owner is free to disclose authentication credentials simply by giving their Picosiblings.

3.3.2 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 has failed. As an example, typing 3 wrong PINs on a mobile phone would lock your 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[6] the scheme only quasi-o ers \infrequent-errors". PINs are generally easily reset by the manufacturer using online services, granting them \easy-recovery-from-loss"[7]. 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". The have virtually no cost, satisfying the \negligible-cost-per-user property". Being a subset of passwords, the mechanism is considered 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. Similarly to passwords, PINs are often written down in plain

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

[7] 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

sight. However, in the lack of relevant studies[8] we will mark the scheme to quasi-o er the \resilient-to-targeted-impersonation" property. 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" due to explicit requests. 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.

3.3.3 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 unlock mechanism.

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 (i.e. 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.

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

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". The scheme relies on proprietary software and therefore is not \non-proprietary".

Observing the owner authenticate using the scheme does not provide any advantage to an attacker. The scheme 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 does not o er the \resilient-to-targeted-impersonation property". 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 \linkable" 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.

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. We have also contributed with 4 original properties.

The framework was applied for three sample token unlocking mechanisms. 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. These will serve as a benchmark for the proposed solution.

As the table shows, none of the example 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.

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: Token unlocking framework sample assessment.

Chapter 4

Design

4.1 Proposed design

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 the user has. However, anyone may be in the possession of the shares, therefore being granted full temporary authentication privileges. This is re ected in the evaluation by failing to fully o er the `resilient-to-theft" and \non-disclosability" properties. A further 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 also need to be supported by the token unlocking mechanism: the authentication process is memory e ortless; and the unlocking scheme needs to support continuous authentication[1]. These features need to be satis ed when designing the new token unlocking mechanism.

The idea explored in this dissertation project is to simultaneously use multiple continuous authentication mechanisms. Each mechanism needs to provide a quanti able con dence level which will be used in calculating a combined score. This satis es the memoryless and continuous authentication properties required by Pico. By combining

[1]Continuous authentication is defined by the ability to re-authenticate the user without the need for any physical effort.

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.

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 user account needs to be associated with a con dence level de ned by the app during the registration process. If the overall con dence level of the unlocking mechanism exceeds the app's con dence level, then the token 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. For this reason 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 with this scheme 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. All mechanisms are assigned a di erent 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 a value.

Data samples captured for the owner authentication process are not always meaningful. For example, accelerometer values for gait recognition are only usable when the user is travelling on foot. 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, re ecting the weight of the mechanism in the overall unlocking process. This value is decreases in time until a valid user data sample is provided to the mechanism for authentication.

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 mechanisms 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 voice recognition mechanism on the overall score. The con dence of each mechanism can decrease up to 0, at which point the mechanism is ignored. Computing the overall score will be explained in more detail later in the chapter.

Each mechanism outputs a value, which is the probability that the sample data belongs to the owner of the token. Upon each recording, this probability is updated using Bayes' Law. This process is also known as a Bayesian update. The equation is described below:

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

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 biometric 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 using the \Law of total probability", which is the following:

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

Using equation 4.2 the Bayes' Law equation 4.1 becomes:

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

Our model however, 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 hypothesis that the data belongs to the owner and $P(H)$ that the data belongs to someone else. Obviously the value of $P(\neg H)$ is $1 - P(H)$ and $P(E|\neg H) = 1 - P(E|H)$ Introducing these values 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, just as explained above.

Now that we have shown how mechanism probability is calculated, and know that each mechanism has a decaying weight based on the last recording time we can continue to calculate the overall con dence of the Pico. This is performed quite trivially using a weighted sum. The following equation shows the process:

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

The result is then compared with the minimum threshold required by Pico. If the requirement is satis ed, the user is granted access for the current app authentication. Due to the continuous authentication property, the Pico token will continue to ask its authenticator whether the con dence level is still satis ed. Based on the decay rate

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

of the weights and the input data available of the authenticator's mechanisms this will constantly be recalculated.

At some point the con dence level required by Pico might be too high for the authenticator to grant access. As an example the owner will want to access it's bank account after being silent in a dark room for the past hour. Let us say this would require a con dence level of 95%, while the authenticator may only output a 20% con dence that the user is still present. Given the circumstances, an explicit authentication mechanism may be required from the user in order to increase the current con dence level.

Combining explicit authentication with the current design can be performed consistently with the continuous authentication mechanisms. Whenever an explicit authentication is required, the only di erence will be the fact that the user becomes aware of the authentication process. They are prompted to pass an authentication challenge (i.e. facial recognition, voice recognition). This would guarantee valid input for the authenticator which may then proceed to compute an accurate score.

4.2 Framework evaluation

We will continue by evaluating the new proposed scheme with the token unlocking framework de ned in the previous chapter.

Memory-effortless: Satisfied

None of the authentication mechanisms require any sort of known secret. Authentication is granted based on biometrics and behavioural analysis.

Nothing-to-carry: Quasi-satisfied

This property is only quasi-satis ed due to the fact that it relies on the implementation of the design. Ideally all authentication data should be gathered from an uni ed device containing the Pico. Alternatively however, the scheme can be implemented using individual sensors which the owner would have to carry, which is why the property is not fully granted.

Easy-to-learn: Satisfied

In order to satisfy Pico's property of continuous authentication, all mechanisms part of the scheme I developed also need to have this property. Therefore the authentication process is non-transparent to the user, and therefore there is nothing to learn.

Efficient-to-use: Satisfied

The authentication data is collected either at xed time intervals, or is red during special events. The authentication process however, does not fully depend on recent data. A response may be generated without any recent authentication data. Therefore the time spent by the mechanism to generate a response is immediate.

Infrequent-errors: Quasi-satisfied

Given that the scheme depends on biometric mechanisms, the quality of the errors is as good as the underlying biometrics. If the scheme cannot generate a high enough con dence an explicit biometric challenge will be issued for the user to satisfy. Since the original biometric mechanisms do not have this property, to some extent neither will the scheme I have designed. However, the scheme is combining multiple biometrics results with di erent score weights based on importance and accuracy. This is much more likely to be accurate, which is why I will mark this as Quasi-satis ed. For a more accurate response, the design needs testing with a high quality prototype.

Easy-recovery-from-loss: Not-satisfied

Token based mechanisms in general do not have this property due to the inconvenience of replacing the token. In our case, the property is also not satis ed. The user would have to re-acquire a new token and recon gure the owner's biometric data. Furthermore based on the mechanism, such as location settings or gait recognition, the token is likely to require an adaptation period.

Availability: Satisfied

Some mechanisms are not always available even though enabled, especially due to the continuous authentication property. As an example gait recognition while sitting in an o ce. However, the scheme may use a multitude of mechanisms with the unlikeness that all of them are unavailable. For instance location history may predict with a certain con dence that the owner still in possession of the token. This propery is aided by the explicit authentication mechanism which requires explicit input from the user.

Accessible: Satisfied

Due to the fact that the scheme is based on multiple biometrics and location settings, I consider this property to be Satis ed or as a very least Quasi-satis ed. The scheme functions based on available biometrics, without having any prede ned solutions. It is highly unlikely that the owner cannot generate any of the available biometric inputs, especially for some such as \face recognition".

Negligible-cost-per-user: Quasi-satisfied

This property depends on the way in which the scheme is implemented. If the implementation is based on high quality sensors embedded in items of clothing and such, then the property is not satis ed. If the implementation reuses sensors that the user already possesses, the the property is fully satis ed as the cost is 0. An example of such an implementation would be an Android application/service possibly using the future Google Glass hardware.

Mature: Not satisfied

This property is not satis ed as the project is at the level of a work in progress prototype. The design is quite fresh and was not implemented by any third party. Neither was is reviewed by the open source community or has had any user feedback.

Non-proprietary: Satisfied

Anyone can implement the scheme without any restrictions such as royalty checks or any other sort of payment to anyone else.

Resilient-to-physical-observation: Satisfied

Since the mechanism is based on biometric data, simple observations from an attacker cannot lead to compromising the user's authentication to the token. The attacker would have no way of reproducing the input through simple observation.

Resilient-to-targeted-impersonation: Quasi-satisfied

Saying that the scheme Quasi-satis es this property is a bit generous. Each of the mechanisms is vulnerable to a replay attack. An attacker may record one of the user's biometric and replay it as a token input. However, given that the token uses multiple mechanisms, some of which being location based, this is a highly unlikely occurrence. The only vulnerable point would be the explicit authentication mechanisms, which carry a lot of weight.

Resilient-to-throttled-guessing: Satisfied

The amount of throttled guessing required for the user to break one of the biometric mechanisms is far too large for this to actually be a threat.

Resilient-to-unthrottled-guessing: Satisfied

Given that the Resilient-to-throttled-guessing property is satis ed, this property is also satis ed.

Resilient-to-internal-observation: Satisfied

This property does not apply to this scheme.

Unlinkable: Not-satisfied

Just as any of the biometric mechanisms, this property is not satis ed by the mechanism. The authentication data maps uniquely to the owner of the token.

Continuous-authentication: Satisfied

The mechanism was designed with continuous authentication in mind. Data is collected periodically with a con dence weight decaying over time. This allows for the token to be used at any time based on current existing data. The only exception breaking the model would be the explicit authentication mechanisms, but these could only be triggered at the beginning of an authentication process using the token.

Multi-level-unlocking: Satisfied

This property is fully satis ed by the authentication mechanism. It allows the token to grant access to di erent authentication accounts based on the precomputed level of con dence that the owner is present.

Let us continue by comparing the results of our proposed scheme with the original Picosiblings solution. The results are summarised in the following table. In the \Proposed scheme" column, properties which are highlighted in order to facilitate the comparison with the Picosiblings solution. The colour green means that the proposed scheme is better, red worse, and no colour means that both properties have the same value.

As the table shows, the proposed solution does not completely dominate the Picosiblings solution, and this is only because of the \Unlinkable" property. Given that our solution is fundamentally based on biometric data, this property could never be achieved. However,

Property	Picosiblings	Proposed scheme
Memory-e ortless	Satis ed	Satis ed
Nothing-to-carry	Not-satis ed	Quasi-satis ed
Easy-to-learn	Satis ed	Satis ed
E cient-to-use	Quasi-satis ed	Satis ed
Infrequent-errors	Quasi-satis ed	Quasi-satis ed
Easy-recovery-from-loss	Not-satis ed	Not-satis ed
Availability	Satis ed	Satis ed
Accessible	Not-satis ed	Satis ed
Negligible-cost-per-user	Not-satis ed	Quasi-satis ed
Mature	Not-satis ed	Not-satis ed
Non-proprietary	Satis ed	Satis ed
Resilient-to-physical-observation	Satis ed	Satis ed
Resilient-to-targeted-impersonation	Satis ed	Satis ed
Resilient-to-throttled-guessing	Satis ed	Satis ed
Resilient-to-unthrottled-guessing	Satis ed	Satis ed
Resilient-to-internal-observation	Satis ed	Satis ed
Unlinkable	Satis ed	Not-satis ed
Continuous-authentication	Satis ed	Satis ed
Multi-level-unlocking	Not-satis ed	Satis ed

our solution performs better than Picosiblings in 5 other properties. Important points of improvement are accessibility, which makes the proposed scheme viable for a larger number of people. The Multi-level-unlocking property is another good improvement, allowing for an enhanced security model.

4.3 Conceptual design threat Model

An accurate threat model on the proposed unlock mechanism must start by analysing the set of assumptions made about the mechanism. From there we can identify available threats and how the scheme can be exploited in order to unlock the Pico without owner permission. Throughout the threat model we will explain how relaxing the initial set of assumptions may change the security outcome. Each model is analysed from an Availability, Integrity, and Con dentiality.

It is important to note that con dentiality is an important category in this evaluation. This is because the device will store sensitive biometric data which is directly linkable to the user. Losing this data, especially in plain-text, would disable the user from ever using the biometric device for which the data was leaked. This is due to the fact that the leaked data could always be replayed, successfully tricking the biometric mechanism.

In each subsection, the model will obviously only introduce issues with the mechanism. Therefore when reading a subsection, the issues are not only those currently presented, but also those from previous subsections that lead up to that point.

4.3.1 Dedicated device with dedicated sensors

We will start from the assumption that the unlock mechanism is integrated on the same device with the Pico. The device is assumed to be dedicated and runs no other software. Furthermore, the set of available sensors will also be integrated within the device. Alternatively there may also be peripheral sensors, with no way for an attacker to tamper with the communication to the authenticator.

Availability

From an availability point of view, an outside attacker cannot create a denial of service scenario. Interactions with the device are performed physically, so therefore the device cannot be made unavailable while in the possession of its owner. If the Pico would temporarily lose ownership, from a software perspective it would lock up due to mismatching biometric and location data, but would become available again in the presence of the owner.

Only hardware modi cation would a ect data availability. Simply disconnecting the sensor would not a ect the scheme's ability to generate viable results due to the fact that multiple biometrics are used. However an attacker could modify a sensor to output wrong data, tricking it into saying the user is never the owner. This would create a successful denial of service attack path where a few sensors output that the owner is never present.

Integrity

Communication paths are not accessible from the outside and therefore cannot be tampered with in order to modify data. Furthermore the device is not running any other software and is therefore safe from any malware attacks.

Only physical tampering with the device would change data integrity. Modifying one of the sensor's and changing its output to some random data would be undetectable by the mechanism.

Con dentiality

No software access as well as no communication with the outside (i.e. wired communication) means that data is safe as long as the device is with its owner.

If the device were to be lost, Storage data should be kept encrypted, similar to the way Ironkey [] protects its data. Unfortunately an attack path may already be identi ed which is due to the fact that using this model the decryption key needs to be stored is nothing the scheme could do to prevent this other than keep the sensors constantly locked for itself. However since the model is built on the concept of shared sensors, this might not be a feasible solution.

Furthermore, communication paths are no longer dedicated. Weather the communication channel is radio or pure software, this introduces a new attack path. A \man in the middle" type of attack may be performed where information data from the sensors is dropped and replaced with bad data. This would create a scenario similar to the one in the previous section, but without the need for physically modifying the sensors.

Integrity

Having shared communication paths with the sensors means that data integrity may be compromised from outside. This goal would be achieved in the previous model only by physically modifying the sensors. Furthermore if the sensors are on the same device as the Pico, malware may modify output data leading to unsuccessful mechanism authentication.

Since Pico and the authenticating mechanism are fully compartmentalised from the outside, their communication is still secure. This compartmentalisation however needs to include all types of storage and communication.

Confidentiality

Unfortunately having shared sensors introduces quite a big con dentiality issue. Given that the sensor data required for authentication is shared, nothing would stop an attacker from collecting just as the Pico unlocking mechanism would. This data could then be replayed to the authenticator in order to unlock the Pico.

This is quite a critical issue. An example of feasible attack pattern would be. A peace of malware analyses when the sensors are locked, and makes assumptions as to when the Pico authenticator is locking them. Based on these assumptions the malware then captures sensor data immediately after the lock was released therefore capturing a possibly valid sample of data.

A more elaborate peace of malware could detect patterns such as time intervals or events that trigger sensor locking. Knowing these patterns it could therefore lock the sensors and gather data just before the Pico authenticator would, and then trick the authenticator by sending it a replay or possibly modi ed data.

Yet another scenario in these circumstances would be to send the Pico authenticator constant bad data and anticipate the trigger of an explicit authentication request to the user. By locking the sensors at that key time the peace of malware could acquire a high quality data sample. Since most of the mechanisms used by the scheme are biometrics, that data sample would represent permanent damage to the user, as an authentication mechanism using that type of biometric could be replayed in any circumstance.

Since the Pico unlocking mechanism is fully compartmentalised, access its storage is secure and therefore any stored credentials are fully protected.

4.3.3 Insecure communication with Pico

This is a special case model which assumes that Pico and the authenticator we have developed are communicating over an insecure channel. The only element we need to consider is the communication between the two participants.

Availability

To do.

Integrity

To do.

Confidentiality

To do.

4.3.4 Shared device with shared components

We will relax the model even more in order to better t reality constraints when implementing the mechanism. In this model, Pico and its authentication mechanism reside in a computing model with shared storage resources. The security of Pico and its authenticator may only be as good as the underlying OS. In order to have a meaningful use-case scenario.

Availability

To do.

Integrity

To do.

Confidentiality

To do.

4.3.5 Proposed secure implementation

A secure proposed implementation is viable using an Android telephone running a TrustZone enabled ARM processor available in ARMv6KZ [] and later models. This device would essentially be divided into two \worlds": the normal world running the untrusted Android OS, and the trusted world running a small operating system written for TrustZone. Both operating systems are booted at power up. In addition the TrustZone OS loads a public/private key pair which is inaccessible from Android.

Ideally Pico would be implemented with its authenticator within TrustZone. This would essentially guarantee complete separation from a memory perspective leaving any sort of malware attack impossible via memory.

Persistent memory is however required in order to store data for each individual biometric mechanism used in the authentication scheme. Unfortunately this type of memory is not protected by the TrustZone OS and constitutes a way for a third party to attack the scheme. However, we could use the TrustZone OS key pair in order to encrypt biometric data on disk. Even though this data is available from Android it would be fully con dential. If properly stored within Android, the OS may even protect its integrity from outside attacks.

Let us consider however that the Android OS has been completely compromised by the attacker and is therefore \hostile". Under these circumstances data con dentiality can still be fully guaranteed. The TrustZone public key could still be used in order to encrypt the biometric data before writing it to disk. Attacks from a memory perspective may only be performed by modifying data stored on disk. This may only lead to a denial of service for the owner, but not a con dentiality breach.

Let us brie y discuss any issues using the availability-integrity-con dentiality framework.

Availability

Only plausible attacks are denial of service through deleting biometric cache les from disk. This would require constant recon guration for the Pico scheme, making the Pico unavailable.

Integrity

Data integrity may only be altered from cache les on disk.

Confidentiality

No known attacks on data con dentiality other than capturing sensor data just as the authenticator would. However this would be possible with or without the Pico being present.

4.4 Related work

Clarke et al write in their paper [13] a few interesting concepts strongly related to the design proposed in this dissertation. They conduct a couple of surveys trying to assess the reliability of a PIN as an authentication mechanism for a mobile phone. In a study involving 297 participants, they assess the use of mobile phone devices in day to day life, existing authentication mechanisms, and the users' attitude towards further security options. The paper reveals a number of bad practices with PIN authentication such as 45% never changing the default factory code, 42% only changing it once after buying the device, weakness due to reusing the PIN in other authentications, forgetting the pin, and sharing the PIN with someone else.

The paper [13] however shows that 83% of users are willing to accept some sort of biometric authentication mechanism in order to unlock their devices. The mechanisms included in the study ranked by popularity by an IBG study [] are: ngerprint analysis, voice recognition, iris recognition, hand recognition, keystroke analysis [14], and face recognition. The paper also talks about continuous authentication, showing that 61% of users would accept a non intrusive biometric continuous authentication mechanism. Combining multiple biometric for continuous authentication is mentioned brie y, but from the perspective of having each active sequentially based on the current user task, which is a divergence point from what we are trying to achieve in this dissertation.

A similar paper [6] written by Clarke et al studies the need for mobile phone authentication mechanisms alternative to the PIN. The authors conduct a survey with interesting results. A remarkable 11% of participants were not even aware of the PIN authentication method used for unlocking a mobile phone. An average of 81% of participants agree that di erent mechanisms should be used, which provide more security. Subscribes have reported both the need and desire for using alternative authentication mechanisms, but at the same time many of them do not use available alternatives available today. More details regarding the study can be found in the original paper [6].

Gregory Williamson writes in his PhD dissertation [15] 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 authentications are required in the multi-factor model (blanket authentication), and one where the second authentication mechanism is only requested from the user if the transactions appears to be risky (risk mode authentication). A risky situation is de ned as either an important transaction such as withdrawing money, or a transaction made under unusual circumstances such as from an unknown device.

A similar approach to the risk mode authentication presented by Williamson [15] is proposed in this project. Our scheme yields a confidence level which may or may not be sufficient to unlock the Pico based on the current active transactions. Similarly, if the confidence level is not high enough, an explicit authentication mechanism will prompt the user for input. As the dissertation by Williamson shows, 75% of users questioned in his study agree with having biometric authentication as a secondary mechanism. This shows promising results in adopting our scheme for token unlocking purposes.

Elena Vildjiounaite et al describe in their paper [16] a similar mechanism of combining biometric authentication data on mobile phone devices. The authors identify the security downside of granting authentication for a long time after a single verification challenge, which is the case for password based systems. They explore an alternative based a two stage risk mode authentication [15]. 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) using as data a weighted sum fusion rule. Mechanism weights are chosen based on total error rates. The second stage is only enabled if the cascade classifier does not identify the owner as being present. In low noise scenarios 80% of the time continuous authentication is achieved without the need for an explicit challenge. In noisy situations (city and car noise), 40 to 60% of authentication is obtained in a unobtrusive way. The cascade classifier was trained with a FAR of 1%, with results showing a false rejection rate (FRR) of only 3 - 7%.

The paper by Elena Vildjiounaite et al [16] is similar in with the solution proposed in this dissertation through the fact that it also combines multiple authentication mechanisms, each being assigned different weights. Differences between the two are in the fact that weights are maintained static in time. The weights of the sums are computed differently, and there is no mention regarding bayesian updates or probabilities. Furthermore, the authors use a classifier instead of producing a confidence level which may be used for granting different levels of security. The results presented by this paper are however encouraging, showing that continuous authentication is feasible using multiple authentication mechanisms.

Chapter 5

Implementation Prototype

Thus far we have developed a new Pico authentication scheme and assessed it using our own token unlocking framework. We then have performed a threat model from an availability, integrity, and confidentiality perspective and have suggested the safest implementation which would be as feasible as possible for the user to adopt.

In this chapter we will described design and implementation details for the prototype of the proposed scheme. The implementation platform will be the Android OS, which uses a Java based SDK for application development.

5.1 Authenticator design

The user authenticator for Android is designed to work as a bound service called UAService. Periodically the service outputs to registered Pico clients the status of the authentication process. Any application may be a client as long as it registers with the service. Furthermore, explicit authentication update requests may be performed by the Pico client.

Since different authentication mechanisms require different update periods, we have chosen each mechanism to be represented by an independent service. This allows for more flexibility such as periodic sampling with different intervals. Another feasible use case for example would be performing voice recognition based on the first few seconds of an outgoing or incoming call. This would require a service that is triggered by a PHONE_STATE intent.

Each authentication mechanism service is started and managed by the UAService. Communication between the UAService and each authentication mechanism is enabled through intents. Using this communication link, requests can be made from each individual authentication mechanism in order to get the current con dence level. This value is equal to the probability that the owner is present, multiplied by the weight carried by the mechanism. Given that each mechanism runs as an independent service, weight decay may easily be performed using an AlarmManager or simply a function which is called periodically within the authentication thread.

Either periodically UAService gets the con dence level and weight from each mechanism. It then calculates the overall result. If the result is above the threshold requested by the Pico client, a \Message" is passed back saying that Pico should unlock. Otherwise a negative result is returned, letting the Pico know it should be locked.

5.2 Implementation details

5.2.1 Main application and services

The user authenticator for Android is designed to work as a bound service. According to the Android documentation a bound service exposes functionality to other application components and as well as external applications. It is developed as a regular service which implements the onBind() callback method to return an IBinder. The service lives only as long as a component is bound to it. The service implementation class is called UAService.

The UAService is a central node in the application. It is a bound service for any Pico client which wishes to register for events. Furthermore, it binds any authentication mechanism that is available, enabling it for authentication.

The UAService periodically broadcasts intents to registered clients saying if the Pico should be locked or unlocked. The following interface is exposed to available Pico applications through the \what" parameter of the \Message" class:

MSG_REGISTER_CLIENT

Used for registering a client. The \Message" should have as the \arg1" parameter the level of con dence required for unlocking. This value should range from 0 to 100. Any values outside these limits will be truncated within the range.

MSG_REGISTER_CLIENT

Used for any application to unregister as a listener for this service. No additional parameters required.

MSG_GET_STATUS

Used by any application when an authentication request is needed. Although the service periodically broadcasts to its registered clients what is the authentication status, explicit requests may also be performed using this \Message".

UAService interacts with AuthMech objects in order to communicate with an authentication mechanism. Each object is responsible for interfacing the communication with an authentication mechanism. A valid authentication mechanism service needs to extends the AuthMechService abstract class which de nes a standard way of communication with the UAService.

Each AuthMechService is programmed as a bound service. UAService binds these services through AuthMech objects. Each AuthMechService exposes the following message passing interface:

AUTH_MECH_REGISTER

Used for registering the UAService service as a client to the AuthMechService.

AUTH_MECH_UNREGISTER

Used for unregistering the UAService service as a client to the AuthMechService.

AUTH_MECH_GET_STATUS

Used by the UAService in order to request the authentication con dence from the AuthMechService. The value will be returned in the arg1 parameter of the Message passed.

5.2.2 Authentication mechanisms

In order to create a functional prototype, we implemented a couple of mechanisms. The focus of the project is not the quality of the biometric mechanisms involved in the prototype, their sole purpose being to demonstrate a proof of concept. Android devices o er a wide range of sensor data such as GPS, accelerometer, camera, and microphone.

Based on the sensor data o ered by Android devices, a wide range of biometric mechanisms can be developed. A non extensive list may include face recognition, voice recognition, iris scanning, keystroke analysis, gait recognition, and many others. The scheme however, requires a clear prede ned list of mechanisms o ering continuous authentication as well as explicit.

A number of continuous authentication mechanisms may be developed using solely the standard sensors o ered by Android devices. The following non-extensive list was achieved, with details regarding what each mechanism means and how it should be implemented:

Face recognition

This mechanism was also implemented for the purpose of the project, and further details are o ered in the following sections. The idea is that based on user behaviour, sampling of the user's face can be performed without any explicit requests. For instance when an user is unlocking the phone it is highly likely that he will be looking at the screen. This creates a good opportunity for the authentication mechanism service to capture an image and determine the con dence level that the unlocker is the actual user.

Voice recognition

This mechanism was also implemented for the purpose of the project, and further details are o ered in the following sections. Note that voice sampling does not necessarily imply a voice password of any kind. Voice can be analysed from a feature's perspective, regardless of the words being spoken. Voice sampling can be performed at any time. With a frequent enough sampling rate, the owner of the device is likely to be present in most voice recordings. For even better con dence the mechanism should be implemented to start recording when a call is either made or received. On Android this can be achieved by implementing a listener for a PHONE_STATE intent.

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 face 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, making the con dence level of the mechanism relatively low.

Keystroke analysis

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: the amount of time the user takes to type letter sequences, words, or individual letters, words per minute, frequent used words, and many others. Based on these features a con dence level can be generated (not carrying a considerable amount of weight). This is harder to implement using solely the Android SDK. A good starting point would be to have a keyboard application developed for the user, which also communicates with the authentication mechanism. Obviously if the keyboard is disabled by an attacker this should still be considered, especially if the authenticator was originally con gured to listen for input.

Gait recognition

This mechanism is based on the concept of analysing individual walking patterns. Di erent people walk in di erent ways, which even though may not be entirely unique for every individual, would still provide some con dence level regarding the user of the device. In the lack of an existing reliable library, e orts have been made to implement this mechanism, unfortunately unsuccessful. The implementation requires accelerometer data from the device, which needs to be normalised from the sensor's perspective. Android o ers activity recognition for walking, driving, or standing still. This is achieved by registering a sensor callback for the TYPE_STEP_DETECTOR composite sensor.

Ear shape analysis

Studies have shown [] that the shape of the human ear contains enough unique features in order to perform biometric authentication. Taking advantage on user behaviour when using a phone, accurate images can be captured in order to perform such analyses. Within

erals are attached, the user is going to move the phone towards his ear. Based solely on timing and/or accelerometer data, accurate pictures could be taken of the user's ear before the camera gets too close. Images captured by such a mechanism could then be used to calculate an accurate confidence whether the owner is the person who is answering the phone.

Service utilisation

This proposed mechanism is not biometric based. It exploits patterns in the Android phone's service and app utilisation. Based on current running applications, services, and the time they were started my create a model where some confidence is given as to whether the owner has changed. This mechanism would only be effective in detecting sudden changes, but may easily be obstructed either by removing the Pico authenticator. Furthermore sudden changes in ownership are not promptly detected which is why the mechanism would have a low weight in the overall scheme.

Proximity devices

A mechanism may be developed which tries to connect with other devices also running the authenticator. The two owners don't necessarily need to know one another for the acknowledgement to be performed. Based on day to day activities, users tend to interact or at least be around a lot of the same people. Weather regular travel schedules, or as a better scenario, working in an office, constantly being in the presence of other known devices should give a confidence as to whether the device is in the presence of the user. This mechanism could only be circumvented by co-workers or friends unlocking the Pico, which is why it should never have sufficient weight to unlock the Pico on its own. In combination with other mechanisms however, it would provide a good sense regarding the owner of the device. It the device is \in good company" there is a good chance the owner is also present. This should be enhanced with time data as to when other trusted device are recognized. Furthermore, based on the ID of the devices the owner comes in proximity to, the mechanism may have different weights for different devices. As an example, even though travelling with your family on holiday and most of the devices there are unknown, given that a number of frequent IDs are in the proximity of the device, the mechanism should still consider to some extent that it is in the possession of its owner. This would work similarly with the Picosiblings idea, but each Picosibling is a device running this authentication mechanism which is frequently in the proximity of the owner.

Location data

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

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.

Some of the continuous authentication mechanisms may also be used for explicit authentication. Based on the non-extensive list mentioned above, the user may be noti ed to provide accurate information for the following mechanism: face recognition, voice recognition, iris scanning, keystroke analysis, gait recognition, and ear shape analysis. By notifying the user that he has to provide more accurate authentication data, the mechanisms get a better chance of providing valid results. The decay rate after explicit authentication will be slower in order to maintain the continuous authentication property of the Pico for the duration of the authentication session.

In addition to the mechanisms mentioned above, a number of explicit authentication mechanisms which do not satisfy the continuous authentication property of the Pico may be implemented using the Android SDK. It is important to note that any other mechanisms not included in this list need to satisfy the memory property of the Pico, according to which the user doesn't need to remember any known secret. A non-extensive list of mechanisms includes the following:

Fingerprint scanner

Devices which may have a ngerprint scanner incorporated, such as the IPhone 5S may use this sensor in order to gather biometric data used for authentication. This mechanism cannot actively be used for continuous authentication due to the fact that the user doesn't come in contact with the sensors on a regular basis.

A mechanism can therefore request explicit fingerprint data, which would then be compared with the owner's biometric model, outputting a confidence for the authentication. This confidence will be combined in the calculation of the overall scheme confidence just as any other mechanism, the only difference being 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 memoryless property, since the user doesn't need to remember any sort of secret. The handwriting would be analysed with a preconfigured set of handwriting samples in order to determine the confidence level that the owner produced the input.

Lip movement analysis

According to the paper [] by TODO, lip movement during speaking may be used to uniquely identify individuals. Lip movement analysis would be performed similarly as described in the paper. The confidence level that the owner produced the input would then be combined in the authentication scheme. This may also be implemented as a continuous authentication mechanism, with with lower success rate expectations due to the way users tend to hold mobile phones, which usually doesn't expose the mouth to the camera.

5.2.2.1 Dummy mechanism

In order to perform tests for different confidence levels, a dummy authentication mechanism was implemented using the AuthDummyService class. It extends the AuthMechService abstract class, which makes it an independent service just to maintain the application model consistent.

The service contains a data access object (DAO) which in this case only produces random confidence levels within a given range. A thread running within the service makes periodic requests to the DAO in order to mimic an authentication mechanism which periodically samples for data. The service is updated based on the produced value.

When the UAService wants to update its overall confidence, it makes a AUTH_MECH_GET_STATUS request to the AuthDummyService service, which returns the most recent confidence level multiplied by the current decay factor. The result is combined with the result from the remaining authentication mechanism services.

5.2.2.2 Voice recognition

The voice recognition mechanism is implemented as a VoiceService class extending the AuthMechService abstract class. When the services onCreate() method is called, it starts an authenticator thread which periodically samples data from the device's microphone.

The library used for voice recognition is called Recognito developed by Amaury Crickx. It is a text independent speaker recognition library developed in Java. It is by no means one of the best voice recognition libraries, but it was best suited for the purpose of this prototype. The library required minimal additional changes. It claims very good results in scenarios with minimal background noise, such as TED talks [] which it was originally tested on by its author.

In order for the application to compile properly, a subset of the rt.jar was required. This is due to \javax.sound." packages included with the library which are not available on Android. Unfortunately \javax" is a core library also available in Android, but without any of the sound features. For this reason, although the required packages are included with the application this is purely done to trick the compiler that everything is in place. In reality none of the functionality o ered by these packages is used by the application. This is avoided by only using the raw features of the library which require direct sound input without any details regarding sound les and formats.

In order to gather samples compatible with the library and manage them properly, we have created the \VoiceRecord" class. This class is responsible of gathering microphone input using a prede ned compatible con guration listed in the following listing:

- Sample rate: 44100
- Channel con guration: AudioFormat.CHANNEL_IN_MONO
- Audio format: AudioFormat.ENCODING_PCM_16BIT

The minimum bu er size required by the class is device dependant and pre-calculated in the constructor. The class wraps an AudioRecord object used for gathering microphone data. Due to the nature of the SDK, the recording is saved as a file which then is loaded into memory whenever needed. Although this is not an efficient approach due explicit loads from disk, it serves the purposes of the prototype.

In order to have a better interface to the Recognito library, a DAO class was created. When initialised, the DAO object loads the owner configuration together with the predefined set of background noises. It instantiates a Recognito object and trains it using the loaded data. This is performed using the \createVocalPrint" public function made available by the library.

Using the audio data stored as a \double[]" array and the sampling rate stored in the \VoiceRecord" class, we can then call the \recognize" functionality of the library in order to get the closest match to either the owner, or one of the background noises used for training. The library then returns the closest match given its training data, together with the Euclidean distance to that match.

In order to convert the Euclidean distance to a percentage confidence level, an acceptable Euclidean distance threshold is used. Any result above the threshold is considered too high and is truncated to the threshold level. Using the following formula we then convert the value to a confidence level, which is the equivalent of $P(E|M)$, the probability that the evidence belongs to the model.

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

Dividing the distance over the threshold yields a confidence 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, yielding values between 0 and 1 where 1 corresponds to 100% confidence level.

Having calculated $P(E|H)$ we then proceed by calculating $P(H|E)$ using the formula mentioned in the design section of this dissertation. Whenever calculating the current confidence level, we use the value of $P(H|E)$ multiplied by the current decay rate, a number which is periodically decreased within the service. Due to the message passing mechanism using intents, this value needs to be an integer and is therefore multiplied by 100. The overall result is stored in the service and updated whenever the decaying weight is modi ed. When a request is made by UAService, the value is returned using the IBinder message passing mechanism o ered by Android.

5.2.2.3 Face recognition

The face recognition mechanism was implemented in the \FaceService" class, which extends the AuthMechService abstract class. It is a service running a thread which periodically collects data from the camera. Each sample is analysed using a face recognition library, and a con dence level is outputted for the current sample. Just as the scheme proposes, this con dence level is multiplied by a weight which is a decaying factor.

The library used for face recognition is a port of the Javafaces library []. This was the closest functional library found that was compatible with the Android API. Javafaces is a library written entirely in Java, but which unfortunately makes use of the \javax.imageio." package which is not available in the standard Android SDK. Since a considerable amount of code needed to be changed, we have created a new library [] based on the original for the Android OS.

I will brie y present the changes made when porting the Javafaces library. The \Bu ered-Image" class had to be replaced by its Android counterpart, Bitmap. All Bu eredImage references and initialization had to be changed. The API was modi ed to support direct Bitmap input in order to add more exibility and lighten the main code of the authenticator. Original data formats for black and white images were assumed to have a single colour channel representing the grey value. This had to be changed within the code in order to re ect the Bitmap convention where all 3 colour channels are present but have the same value. Additional modi cation were required such as data type mismatches as well as other smaller issues.

Unfortunately, this library combined with the Android SDK does not provide accurate results. This is due to the fact that the library requires a rectangle bitmap perfectly containing the face of an individual. Unfortunately the Android SDK although o ers face detection, it only provides the location of the midway coordinate between the eyes, and the distance between the eyes. Using this data alone, an accurate crop cannot be made. As a solution, yet another library would need to be used in order to detect faces and provide more accurate data regarding their location and boundaries.

Every xed time interval, a thread running within the \FaceService" object samples data from the camera at a xed time interval. This is performed using an instance of the Camera class. Additional con guration is required based on the orientation of the phone. On the device the prototype was developed [], when the phone is held normally a 90 degree rotation of the image is required.

By default, the Android API does not easily allow for a Camera picture to be taken without any sort of noti cation to the user. By default both a shutter sound and a visual preview display should be present. The shutter sound can easily be disabled by simply not implementing any shutter callback function. The preview display however proves to be a bit more di cult. The solution used with this prototype was to exploit Android's option to render the preview image to a \SurfaceTexture" object. This satis es the API's request to have a visual display preview for the camera, while the \SurfaceTexture" itself doesn't need to be displayed on screen. Therefore an picture can be taken from a background service without the user being aware of this event.

A DAO class called \FaceDAO" was developed in order to interface with the Javafaces library port. The authentication thread running within the \FaceService" object periodically captures an image from the camera. The image is then validated using the \FaceDAO" object. The value returned from the Javafaces library is the Euclidean distance to the closest registered user, which in our case is the owner of the device. This distance is handled in exactly the same way as the voice recognition mechanism.

Another problem encountered by face recognition mechanism is the size of the data involved in performing the face recognition. With standard pictures, the application runs out of memory and is closed by the Android OS. In order to x this issue, Bitmaps collected from the camera are scaled to 50

5.2.3 Owner con guration

In order to con gure the biometric authentication mechanisms in a exible, controlled manner a couple of Android activities were developed. There are used to set the initial owner biometrics based on which the mechanisms will output their con dence levels. These activities use the same DAO classes in order to store the data once it was collected.

Due to the size of the data, which is relatively small, the les can be stored in the application's internal memory, making it inaccessible by other applications.

5.3 Threat model

Even though the scheme implementation is a proof of concept, we will continue by analysing di erent threat models. This will reveal any aws behind the concept, allowing for a more robust future implementation.

The purpose of the Pico token is to provide a robust authentication mechanism, without the use of any secrets for the owner to remember. Where the Pico unlocking scheme ts, is correctly identifying the owner of the Pico. Attacks may be performed in the form of malware installed on the device while still within the possession of its owner. The main threat however comes from an attacker having physical access to the user's Pico.

It is important to note that since this is a purely software implementation, physical access may mean either that the attacker is in possession of the phone, or that it may replicate the secretes of the victim's Pico on a separate device. Replicating the Pico secretes would clearly create much more damage for the user from a cost perspective. A total reset of authentication credentials would be necessary for all accounts registered with the Pico device.

5.3.1 Literature review: Android security

In order to perform a valid threat modelling of the scheme, we need to have a better understanding of the Android security model, and the \features" o ered by di erent mechanisms such as Intents and lters based on these intents.

An interesting paper by William Enck et al [17] o ers a good description of the Android OS, with a focus on the security aspects of the platform. It is a relatively old paper from 2008 which is the same year of the Android initial release. The initial set of Android open standards was however released sooner, in November 2007, allowing researchers such as William Enck to perform an initial analysis of the system.

Android runs on a port of the Linux kernel. This means that many of the linux security mechanisms such as le permissions, are also part of Android. On top of the kernel lies an

App middleware layer which the Java SDK can use in order to extends the functionality of the hand held device.

Each android application is split into multiple components, with no xed entry point such as a main() routine. Based on its purpose, the SDK de nes 4 types of components: activity, service, content provider, and broadcast receivers. More details regarding the purpose of each component can be found on the Android website [18]. Another important architectural note is that components, and even applications, may communicate between each other using Intents. This type of communication is abbreviated as Inter Component Communication (ICC).

An important note in the paper is that when a Service becomes bound by another component it cannot be terminated by an explicit stop action. This provides an useful guarantee regarding the lifetime of a service.

Th two types of security enforcements can be split into two categories: ICC and system level. System level security is mainly de ned in terms of the user id (UID) and group id (GID) permissions. According to the Android OS, each is allocated an UID and GID. The security guarantees of this mechanism are the same as those of a linux based system. An interesting example would be a vulnerability of the T-mobile G1 phone browser, which due to this system level security enforcement did not a ect any other applications.

The focus of the paper however is ICC security enforcements. These are based on I/O control command of the \/dev/binder" special node. Since the node needs to be world readable and writeable, linux cannot mediate ICC. This is performed using labels using a MAC framework enforced by a reference monitor. This de nes how apps are allowed to access di erent components.

Components may be de ned as either public or private. This re nement is con gured by the \exported" eld de ned in a manifest le. It de nes whether or not another application may launch or interact with a component from another application. At the time the paper was written, the \exported" eld was defaulted to \true". However, as shown in a more recent paper [18] written by Ste en and Mathias in 2013, starting with Linux Android (LA) 4.2 the default of this value was changed to \false", therefore conforming to the \principle of least privilege".

Components listening for Intents need to have a registered filter in the manifest file. This is both convenient and secure for the developers of the system. If however the developer of an application wishes to restrict access to intent objects, the SDK provides user definable Intent permission labels as well as Service hooks. These provide runtime security checks for the application and prevent data leakage through ICC. Using permission labels, the developer may broadcast events to which only components which registered that permission may access. The same principle applies to service hooks, but in this case a component holding some permissions tries to bind on a service which checks these permissions and exposes different APIs based on the permissions held by the binding component.

Another more recent paper was written by Steffen and Mathias [18]. The authors focus on deeper issues of the LA, and how they were solved from one update to the other. However, it is shown that OEMs tend not to update the software of their devices once they have shipped, which poses a number of security issues.

The starting point of understanding Android security and how it is bootstrapped is the five 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.
4. LK initialises all existing hardware and starts the linux "init" process [?].
5. The init process reads a configuration file and boots the rest of LA.

The android security model described in this paper [18] is split in two categories: system security, and application security.

The android base system (libraries, app framework, and app runtime) is located in the "system" partition. Although this is writeable only by the root user, a number of exploits which allow root access, such as those shown in [18], have been possible. Android provides a keychain API used for storing sensitive material such as certificates and credentials. These credentials are encrypted using a master key, which is also stored in AES encrypted format. In this case, security needs to begin somewhere, an assumption has to be made about a state being secure which would allow for multiple security extensions. In this case, the master key is considered to be that point of security. However, given a rooted device, where due to an exploitation a process may be granted root privileges, the master key itself may be retrieved from the system therefore compromising all other credentials.

From the application's perspective, an interesting \feature" which may a ect the ow of information within LA is the fact that applications from the same author may share private resources. When installing an application the user needs to accept its prede ned set of permissions. Due to resource sharing, a situation may present itself when an application which has permissions for the owner's contacts may communicate with an application which 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 LA system o ers a number of memory corruption mitigations in order to avoid bu er over ow attacks, or return oriented programming attacks. The following list includes mechanisms enabled in time based on the LA version:

- 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.
- Address space layout randomisation(ASLR) implemented starting with LA 4.0. This is a rst step to preventing return oriented programming attacks. The position of the binary library itself is however static, meaning that after a number of attempts, using trial and error, the attacker may succeed using return oriented programming.
- Position independent and randomised linked (PIE) is implemented starting with LA 4.1 in in support of ASLP. This makes position of binary libraries themselves randomised.
- Read only relocation and immediate binding space(RELro) was implemented starting with LA 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 through the GOT table.

A number of security enhancement mechanisms are in place in order to make Android a safer environment for its users. In order to prevent malware within the Android App store (Google Play), a program also known as a \on device Bouncer". The purpose of the bouncer is to verify apps prior to installation for any malware signature or patterns. Secure USB debugging was introduced starting with LA 4.4.2, which only allows hosts registered with the device to have USB debugging permissions. This mechanism is circumvented if the user does not have a screen lock.

As pointed out by Ste en and Mathias, the Android OS is responsible for 96% of mobile phone malware according to a study from 2012 []. The authors claim that this is the case due to 4 big issues of the Android concept:

1. Security updates are delayed or never deployed. This is due to a number of approvals that an update needs to get throug in order to be pused to devices. This implies an additional cost to the manufacturer(OEM) which does not generate any revenue. The majority of teams working on LA are focusing on current releases, and in some cases there is simply not enough time and resources to merge Google security updates to the OEM repository. Furthermore, the consequences of a failed system update for the user may cause problems as serious as \brick"-ing the device which is a huge risk for the manufacturer. All these issues contribute to a severe lack in security updates. Therefore, important updates such as RELro are never pushed to LA 4.0, making the device vulnerable.
2. OEMs weakens the security architecture and con guration of LA by introducing custom modi cations before they roll out a device.
3. The Android permission model is defective. As pointed out by Ste en and Mathias, according to Kelley et al [19], 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 be able to use a new exciting app. According to the same study, most applications are over-privileged due to the developers not understanding what each privilege does. Furthermore, as previously

pointed out, applications from the same owner may share resources, therefore creating a valid data ow attack path.

4. Google Play has a low barrier for malware. A developer distribution agreement (DDA) and a developer program policy (DPP) need to be greed 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 suspect of breaking the agreements. According to [20] there are ways of circumventing the Bouncer program. An example is treated in an article [] written in Tech Republic.

5.3.2 Prototype threat model

Let us now continue by studying the threat model of the Pico authenticator application. We will consider the security mechanisms presented above as the prede ned assumptions made in this model. In order to reduce the threat space we will consider the application is running on a hand held device running Android 4.4.2 with all recent updates.

Availability

Breaking the scheme's availability if 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 for the di erent mechanisms. Furthermore, in this case the owner is already no longer in possession of their Pico, so basically the Pico is already made unavailable..

Let us continue however and study what can be achieved from a DoS perspective by the attacker from the perspective of the individual user app accounts, which would need to be reset by the owner. In order to gain any sort of access and make credentials reset not possible, or at least have a chance in doing so, the attacker would have to unlock the Pico.

From a malware attack perspective data used by the authentication should not be modi able. This would guarantee that all mechanisms have their cached biometric data available at all times and may function properly. Due to the Linux permissions mechanism and the fact that each application has its own UID and GID, data stored in internal memory should not be readable or writeable by any other user level application in the system. If however the device is rooted and the owner is mislead into granting root privileges to another application, then the security model would be broken and the data would be exposed. This could lead to deletion which would make the mechanisms not function properly, resulting in a DoS attack.

Integrity

Just as mentioned in the Availability section, the authenticator should be safe against any data accesses from other applications as long as the application does not have root privileges. This would allow malware to break the integrity property of the data.

From a data flow point of view Intents used for communication within the authenticator as well as with the Pico application are not modifiable. Furthermore Intents are not broadcasted using the implicit Android broadcast mechanism, which makes them impossible to replay or even intercept.

Confidentiality

Considering a circumstance where a root malware process would have access to the authenticator's data stored on disk, this would not lead to a direct compromise of the owner biometric data. All cached files are stored in internal memory in encrypted format. The mechanism used for encryption is RSA, with the private key stored using the Android KeyChain API.

On a rooted device however, the encryption layer provides only another bi-passable layer of security. With root access, an application could retrieve the master key of the KeyChain and use it to retrieve or private key and decode the owner's biometrics.

From a data flow perspective, internally the Pico authenticator uses an self-developed broadcast system. Client processes need to register with the broadcaster, such as the UAService, in order to receive updates. This ensures data confidentiality throughout the system. Furthermore, the authenticator and Pico should be released under the same author. This would allow locking the application from outside Intents as well as interaction with different components. Sandboxing communication is always a desirable property from a confidentiality perspective.

The paper by Adrienne Porter Felt et al [21] shows that according to their surveys only 17% of users pay attention to the Android permissions dialogue, and only 3% understand what each permission represents. A malware application which has granted full permissions gets pass the Bouncer and is installed as an application. Even so, due to the Linux permission model adopted by Android, the con dentiality of the authenticator's data would not be compromised. Instead however, the malware application may collect all relevant on its own from the user, allowing for a powerful replay attack in the future.

5.4 Future work

The application was implemented as a proof of concept. It is developed in order to show that di erent data may be obtained without the owner's knowledge. Additional improvements can be made in order to increase the con dence level of the authenticator. Furthermore, due to time constraints and unavailability of free to use biometric libraries, a number of mechanisms were not implemented. The list can easily be extended by simply creating a class which extends the \AuthMechService" abstract class.

One way to improve the voice recognition mechanism would be to start sampling data whenever a call is active. This would increase the chances of capturing an accurate sample of the owner's voice. In this context, a better voice recognition library can be used, which supports multiple speakers and/or ignores background noise. If such a library is not available, we can rely on the fact that most of the times people take turns when speaking. For the duration of the call, with a high enough sampling frequency, the individual sampling voice of both participants should be captured. However, it is important to take into account a situation in which the thief is calling the owner on a di erent phone in order to unlock his Pico.

Immediate improvements can be made to the face recognition mechanism. Just as recommended in the description of the mechanism's implementation, another library which provides more meaningful face coordinates may be used for face detection. Alternatively, and preferably, a di erent library which performs both face detection an recognition can be integrated with the mechanism.

Another improvement for the face recognition mechanism would be from the data sampling perspective. Instead of capturing images at a xed interval, pictures should be taken only when the phone unlock event is triggered. While the phone is unlocked it is highly likely that the user will face its front camera.This would provide better chances of processing meaningful data.

5.5 Results

Need to gure out how to present meaningful results for di erent scenarios, and how the mechanism would work. Use dummy authenticator to generate data.

5.6 Related work

Liang Cai et al makes analyse in their paper [22] 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 it is noted that the user is not to be trusted with granting permissions to di erent applications. An important point made in this paper is the fact that malware may deny service to legitimate applications such as our authenticator by creating a race condition for acquiring a lock on the sensor. The solution proposed by the authors would be an user noti cation, allowing for the owner to decide which application acquires the lock. A suggestion to this approach would be to allow for di erent priority levels, such that malware applications would not acquire the lock in a race condition, or even more, would lose it when a high priority application such as the Pico authenticator would require sensor data.

The paper by Derawi et al [23] 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". According to the paper, the Android implementation has an equal error rate (EER) of 20%. Dedicated devices have an EER of only 12.9%, and the cause for this is the sampling rate used by the authors. They have used a Google G1 phone with about 40-50 samples per second, which is much inferior to dedicated accelerometers which sample data at 100 samples per second. However, by conducting personal experiments with the Accelerometer of a Google

Nexus 5 phone, using the highest sampling setting (SENSOR_DELAY_FASTEST) sampling rates go above 100 samples per second. Therefore the performance of devices has increased, making the authentication mechanism more reliable.

Ming et al presents in his paper [24] how to improve speaker recognition accuracy on mobile 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 biometric implementation related and is therefore outside the scope of the project.

Another technique in performing speaker recognition involves using voiceprints. These are a set of features extracted from the speaker sample data. Kersta explains in their paper [25] the mechanism in more detail. The benefits of having feature extraction based on a voice sample as opposed to a different voice recognition mechanism is the fact that voiceprints do not require knowing any secrets. The speaker doesn't have to reproduce a voice sample. This increases the usability of the mechanism in the scenarios required by the Pico authenticator.

A popular paper on face authentication [26] was written by Turk and Pentland. The biometric authentication process is based on the concept of eigenfaces. Eigenfaces are a name given for the eigenvectors which are used to characterise the features of a face. These features are projected on to the feature space. Using Euclidean distances in this feature space, a classification can be performed in order to correctly identify individuals. An implementation of this concept was implemented with the prototype of the Pico unlocking scheme.

A more unconventional method for authenticating users was presented by Clarke and Furnell in their paper [27] on keystroke analysis. This mechanism is unobtrusive and authenticates users during normal interactions such as typing a text message or a phone number. It is based on a neural network classifier, reporting an EER of 12.8%. Input data used for classification is composed out of time between successive keystrokes, and hold time of a pressed key.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

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

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

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

[4] 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.

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

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

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