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 provided 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 users, but with a constant increase 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 started to be enforced when creating a password. Some password mechanisms may require a minimum length, one or more numeric characters, or 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 provided by the Picosiblings, 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. Some additional security features are suggested, 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 biometrics as part of an unified solution. The output from each biometric mechanism is combined to generate an overall confidence level that the owner is still in possession of the Pico.

We will explore and evaluate 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 the Picosiblings. The data is then be 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 designed 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 different 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 the 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 memoryless 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 benefits 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 offers 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 effort, such as searching or typing credentials, is required from the user.

Another important feature offered by Pico is continuous authentication. Traditional password mechanisms authenticate the user for an entire session. The user is responsible of managing and closing the session when it is no longer needed. Instead, Pico offers the possibility of periodic re-authentication of its owner using short range radio. If either the Pico or its 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, similarly to a key. It contains the following hardware features:

- 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 notifications.
- 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. 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 certificate 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 find 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.

An important aspect of Pico which was not yet fully discussed is the locking process. The Pico 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 the Pico. After the initialisation process, each Picosibling responds to a ping request from the Pico. During each successful ping, the Picosibling sends its k-out-of-n share back to the 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 predefined 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 briefly 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-off 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 solution to Picosiblings. The project aims to achieve better results in some categories, without necessarily completely outperforming it.

3.1 Related work

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 mechanisms. For example, properties such as "Browser-compatible" do not apply, while others need to be redefined to fit 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 difficulty 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 assessed by evaluating whether it offers or does not offer each property. In the case where a scheme almost offers a property, the authors mark it as Quasi-offered. To simplify the framework, properties which are not applicable are marked as offered.

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", "efficient-to-use", "and easy-recovery-from-loss". However, from a security perspective passwords don't perform as well. They only offer 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 offer 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 used by an attacker using a pre-recording of the user. Due to the ease with which valid sample data may be gathered by an attacker, they are also not resilient to theft. There is a one to one correlation between the owner and their biometric recording, therefore the "unlinkable" property is not offered by these mechanisms.

By analysing the framework results, we see that some authentication schemes, such as security tokens, offer "memory-effortless" in exchange for the "nothing-to-carry" property. The only schemes that offer both are biometric mechanisms. This is a consequence of repacing "something you know" with "something you are" instead of have. For different reasons no mechanism offers both "memory-effortless" and be "resilient-to-theft".

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

[1] Not applicable to passwords

secure authentication mechanisms, 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 offers a property if it is offered by both authentication mechanisms. In terms of security, only one of the two mechanisms needs to offer the property in order for the two factor combination to offer it as well. Wimberly & Liebrock [12] observe that combining passwords with a second authentication mechanism scheme leads to weaker credentials.

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

3.2 Token unlocking framework

Web based authentication mechanisms are initiated locally and performed remotely. In contrast, token based authentication is initiated and performed locally, leaving no room for man in the middle attacks or any other 3rd party participation. For this reason, a subset of the properties described in paper [5] by Bonneau et al should also be present in the framework we have developed. Properties from the original framework by Bonneau et al which do not apply, or would be satisfied by any token unlocking mechanism were removed in this work.

The following list shows what properties of the framework developed by Bonneau et al are relevant to token based authentication mechanisms:

Memory-effortless

Different types of tokens would have different results. For instance the RSA SecurID [] token doesn't require any authentication, while the FIDO (Fast IDentity Online) Alliance [] may use a PIN requiring a known secret.

Nothing-to-carry

Different token unlocking mechanisms may require additional hardware. Such may be the case for some biometric schemes.

Easy-to-learn

Token authentication mechanisms may have different learning curves. As an example a CAP reader is fairly easy to use, while a Pico device may prove more difficult for the inexperienced.

Efficient-to-use

Time required by the token user authentication mechanism may differ from one type of authentication to the other. The time required for registering a new user or unlocking the token for its owner should be reasonably short.

Infrequent-errors

The token unlocking mechanism may reject true positives. If the number of false negatives is reasonably low, then the mechanism has this property.

Easy-recovery-from-loss

The user's ability to get another token which uses the same authentication mechanism. Tokens which unlock using biometrics for instance, if not properly secured may lead to the user's inability to use that mechanism again.

Accessible

The ability for all users to use the authentication mechanism. As an example, PINs may be entered by any user regardless of disabilities, on the other hand other biometric mechanisms may not be available.

Negligible-cost-per-user

The total cost enquired by the user in order to use the authentication mechanism.

Mature

It refers to the number of users that have successfully used the mechanism, open source projects based on the mechanism, and any other usage by a third party which did not participate in the development of the scheme.

Non-proprietary

Anyone can implement the token unlocking scheme without having to pay royalties to anyone else.

Resilient-to-physical-observation

An attacker should not be able to impersonate the user after observing them authenticate.

Resilient-to-targeted-impersonation

An attacker should not be able to impersonate the user using knowledge about the user, or previous recordings of his biometrics.

Resilient-to-throttled-guessing

The resilience to an attacker automating a guessing process in order to brute force the unlock of the token.

Resilient-to-unthrottled-guessing

An attacker which only physical access to the token cannot guess the required unlocking resource.

Resilient-to-internal-observation

An attacker cannot tamper with the token in order to intercept user input. Furthermore it is impossible for the attacker to gather the input from within the token's storage.

Unlinkable

The unlocking mechanism does not generate data which if leaked would compromise the identity of the user. ...

The properties described above are derived from the original framework presented by [5]. Additional details relevant to each property, including when a property is only quasi (partially) satisfied may be found in the original work by Bonneau et al. Some properties, such as Nothing-to-carry or Server-compatible, do not apply for token unlocking schemes and therefore are not included.

Although the framework by Bonneau et al provides a good base set of properties, a few others are needed in order to fully characterise token unlocking mechanisms. The following list is part of the project's contributions to the overall evaluation framework.

Continuous-authentication

The concept, although mentioned, is omitted from the framework developed by

Bonneau et al [5]. It stressed a bit more as part of the benefits of Pico by Stajano [7]. This is a property belonging to the security category of the original framework. The property is satisfied if, once authenticated, the user remains authenticated to the token for as long as he is in its presence. This is similar to an authentication session with the added property that the session remains active for as long as the user requires it. The property should be satisfied by mechanisms which may re-perform authentication in a non explicit way, leaving the user unaware of the underlying process.

Multi-level-unlocking

This is a security category property. If satisfied, the unlocking mechanism may allow for multiple types of unlocking based on the user confidence or identity. This is something that may be characteristic for mechanisms which involve biometrics or accounts with multiple security levels.

Availability

This is an usability property. If satisfied, the user has the ability to use the unlocking mechanism in any circumstance. This is not related to any disabilities preventing the user from using certain mechanisms. The properly is strictly related to whether the scheme can be used in any circumstances, and may provide feedback regardless of light, noise, or other external factors. As an example gait recognition would only function while moving on foot, or a recognizer that restricts access based on pulse would not satisfy this requirement. A mechanism requiring only a PIN on the other hand would work in any circumstance.

3.3 Example evaluation

In order to demonstrate how the framework works we have assessed three token based authentication mechanisms: Picosiblings, PIN, and Face-unlock. Each of the mechanisms represent a different type of authentication method. Picosiblings essentially are a secret the owner has, PINs are a secret the owner knows, and Face-unlock is something the owner is. The following sections will provide as an example of how the framework

should be applied. Results in the Picosiblings section will be used in the following chapter as a token authentication benchmark which will be used to compare the proposed unlocking scheme with the existing one.

3.3.1 Picosiblings

The current token unlocking mechanism used by Pico is called Picosiblings. In order to unlock the token, k-out-of-n secrets are required. Each of these secrets are held within a Picosibling carried by the user. If the Picosibling is within range of the Pico, these secrets are communicated to the token which can then be used to unlock its master key and implicitly the token.

Since the user doesn't need to remember any secrets, the scheme satisfies the memory-effortless property. The nothing-to-carry property is not satisfied since the Picosiblings are external emitters. The easy-to-learn property doesn't quite apply to the authentication mechanism, given that the user only has to carry the Picosiblings. In the original paper this was marked as not satisfied due to the unfamiliarity of the user with the Pico device, not the authentication mechanism. It only partially satisfies the efficient-to-use, and infrequent-errors properties. Easy recovery from loss is not offered neither for the Pico or the Picosiblings, since additional hardware would need to be ordered. The scheme relies fundamentally on the presence of the Picosiblings, but may be used in any normal scenarios as long as the devices are present. Therefore it satisfies the availability property.

The original paper marks Pico with the Picosiblings scheme as not accessible. We will assume the property transfers to the unlocking mechanism as well. The negligible-cost-per-user property is not satisfied due to presumably expensive embedding of Picosiblings in everyday items. The scheme was not used in any open source projects and is at the stage of a prototype, having very little user testing. For this reason the scheme does not satisfy the mature property. Since this is an academic research project, no royalties need to be paid in order to implement the scheme. Therefore the scheme is non-proprietary.

Since it does not rely on any user input it is resilient-to-physical-observations. In disagreement with the original Pico assessment by the authors, since anyone in the possession of the Picosiblings may unlock the Pico it is not resilient-to-targeted-impersonation.

It is however resilient-to-throttled-guessing and resilient-to-unthrottled-guessing due to the communication protocol described by Stajano in his paper [7] as well as the resurrecting ducklings protocol [11]. Due to the use of cryptography is is also resilient-to-internal-observations, and if lost the data is unlinkable to its owner. The scheme was designed to provide continuous-authentication. Since the mechanism offers secrets which keep the master key as either locked or unlocked, it does not satisfy the multi-level-unlocking property.

3.3.2 PIN

PIN should be viewed as passwords with a severely limited subset of 3(reearacteffrreesfftio6g)-Adrna

passwords, PINs do have the property of resilient-to-throttled-guessing property due to the security model around them. A brute force attack cannot be performed on a locked device unless some sort of implementation error is present, such as an unprotected file. Furthermore PIN mechanisms are generally designed with different penalties such as locking the SIM or the OS for a time period. Due to poor practices in using PINs, they do not have the resilient-to-unthrottled-guessing property. PINs are not resilient-to-internal-observations. The fact that they are chosen randomly by the user means they are unlinkable. However, they do not offer the continuous authentication property due to the fact that they cannot authenticate the user in a non-evasive way. They only offer a locked/unlocked state, and therefore they do not have the multi-level-unlocking property.

3.3.3 Android Face unlock

Android offers a face recognition unlock mechanism for the mobile device. A mobile phone may be viewed as a multi-purpose token, making the face-unlocking mechanism a valid token unlocking scheme. This feature was made available starting with Android 4.0 (Ice Cream Sandwich). This essentially is a biometric unlocking mechanism, which due to the sensor platform offered by Android may be used for token unlocking purposes.

As no secrets are required for biometric mechanisms, the scheme is memory-effortless. It satisfies the nothing-to-carry property as well, since in the case of mobile devices with a camera the scheme does not require additional external hardware. The mechanism is efficient-to-learn, since it only needs the user to look at the camera. Time is required to load face models when first loading the mechanism, but not upon further authentication requests. Therefore the scheme quasi-satisfies the efficient-to-use property. Errors are quite frequent for this mechanism, both from personal testing as well as existing literature [11]. The easy-recovery-from-loss does not apply for biometric mechanisms, especially in the case where no external hardware is required. The availability property is not satisfied due to the fact that based on external factors such as light or obstacles between the camera and the face authentication is not possible.

To do: please provide some input for cost-per-user, mature, and non-proprietary. Face recognition is accessible for anyone to use. It has a negligible cost-per-user due to the

fact that it is non-proprietary and no charges are usually made with each authentication. The mechanism quasi-satisfies the mature property due to the limited user exposure.

Purely by observing the owner authenticate using face recognition does not provide any advantage to an attacker. The scheme therefore has the resilient-to-physical observations property. Targeted impersonation is an issue with any authentication mechanism. Nothing would stop an attacker from taking a picture of the owner and use it in a replay attack. Therefore the scheme does not offer the resilient-to-targeted-impersonation property. The resilient-to-throttled-guessing and resilient-to-unthrottled-guessing properties do not apply and therefore are considered as satisfied. The resilient-to-internal-observations property is not satisfied in the case of mobile devices. This is due to the fact that malware may capture the exact same data when the screen is turned on for example, or immediately after a picture was taken. If data used in authentication is exposed, it is directly linkable to the owner, and therefore the unlinkable property is not satisfied. Continuous-authentication would only possibly be offered based on implementation and conditions such as the user facing the camera of the phone. The property is therefore only quasi-satisfied, although arguably not satisfied. Multi-level-unlocking would be possible based on an Euclidean distance metric, but no efforts have been made in this regard.

3.4 Conclusions

At the end of this peace of work a new framework for the evaluation of token unlocking mechanisms was developed. Existing properties have been identified from the literature and added to the framework together with original work. An initial evaluation was made for existing token unlocking mechanisms which will serve as a benchmark for the proposed solution. From the example evaluations, neither of the three token unlocking mechanisms is dominant over the others.

Chapter 4

Design

4.1 Proposed design

The design proposed by Stajano [7] requires Pico's locking and unlocking mechanism to be performed without the use of any password mechanisms. Furthermore support needs to be added for continuous authentication. What this means is that Pico needs to detect the presence of the user in a non obtrusive way.

Given these properties, my idea is to combine multiple continuous authentication mechanisms in order to compute a confidence level. If the confidence level is satisfactory then the Pico is notified that the owner is present and becomes unlocked. This would provide a non-obtrusive continuous authentication mechanism that would suit the Pico requirements for satisfying its claims.

Since the process needs to be continuous and non-obtrusive, most of the mechanisms rely on biometric data such as iris recognition, face recognition, voice recognition, gait recognition, and others. Another source of data would be the user's location compared with previous history, and other patterns that give with a certain confidence whether the owner is in the presence of the Pico.

The approach offered by this project is different from simply stating that Pico is using biometric data as an unlock mechanism. The novelty in this design is based on how the data is combined in order to compute the owner confidence level. Each is assigned a different weight based on the level of trust it offers in identifying the user. This doesn't

necessarily need to be the precision of the mechanism but it would be a good indicator for choosing a value.

Just as the Picosiblings solution, the Pico may have a wide range of supported biometric inputs. However, not all data is always available or relevant. As an example gait recognition would only work while the user is travelling on foot. Other biometrics such as iris recognition may not always be available based on how the sensors are integrated and carried by the user. For this reason all mechanisms will have a decaying confidence level which decreases in time from the last successful recording.

Let us take for example voice recognition which would be sampled every minute. The current weight of the mechanism is 0 so its output is completely ignored. The next sample is recorded showing a confidence of 70% that the owner is present. Upon this recording, the mechanism weight is updated to it's original value. For the next 10 minutes the owner will be silently reading. Since the mechanism does not manage to identify any voice present, the weight of the mechanism decays. The overall result will be impacted less by the voice recognition mechanism up to the point that the recording will be so old, it will no longer be taken into account, or a new successful identification will be performed.

Each mechanism outputs a probability that the recorded data represents 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:

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

In the equation above:

- $P(H|E)$ represents the probability of hypothesis H after observing evidence E; this 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.
- $P(E|H)$ represents the probability that the current evidence belongs to hypothesis H.

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

We will use the "Law of total probability" in order to compute the value of $P(E)$ in order to accurately compute the posterior probability. The formula is the following:

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

Using this the Bayes' Law equation becomes:

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

Our model however, contains only two hypothesis: the recording of the data either belongs to the user, 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 $P(\ H)$ is $1 - P(H)$ and $P(E|\ H) = 1 - P(E|H)$ Using this information, the rule for updating mechanism probability that the recording belongs to the user becomes:

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

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 confidence 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 satisfied, 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 confidence level is still satisfied. Based on the decay rate of the weights and the input data available of the authenticator's mechanisms this will constantly be recalculated.

At some point the confidence 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 confidence level of 95%, while the authenticator may only output a 20% confidence 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 confidence 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 difference 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 defined 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-satisfied due to the fact that it relies on the implementation of the design. Ideally all authentication data should be gathered from an unified 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 fixed time intervals, or is fired 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 confidence 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 different score weights based on importance and accuracy. This is much more likely to be accurate, which is why I will mark this as Quasi-satisfied. 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 satisfied. The user would have to re-acquire a new token and reconfigure 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 office. 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 confidence 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 Satisfied or as a very least Quasi-satisfied.

The scheme functions based on available biometrics, without having any predefined 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 satisfied. If the implementation reuses sensors that the user already possesses, the the property is fully satisfied 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 satisfied 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-satisfies 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 satisfied, this property is also satisfied.

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 satisfied 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 confidence 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 satisfied by the authentication mechanism. It allows the token to grant access to different authentication accounts based on the precomputed level of confidence 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-effortless	Satisfied	Satisfied
Nothing-to-carry	Not-satisfied	Quasi-satisfied
Easy-to-learn	Satisfied	Satisfied
Efficient-to-use	Quasi-satisfied	Satisfied
Infrequent-errors	Quasi-satisfied	Quasi-satisfied
Easy-recovery-from-loss	Not-satisfied	Not-satisfied
Availability	Satisfied	Satisfied
Accessible	Not-satisfied	Satisfied
Negligible-cost-per-user	Not-satisfied	Quasi-satisfied
Mature	Not-satisfied	Not-satisfied
Non-proprietary	Satisfied	Satisfied
Resilient-to-physical-observation	Satisfied	Satisfied
Resilient-to-targeted-impersonation	Satisfied	Satisfied
Resilient-to-throttled-guessing	Satisfied	Satisfied
Resilient-to-unthrottled-guessing	Satisfied	Satisfied
Resilient-to-internal-observation	Satisfied	Satisfied
Unlinkable	Satisfied	Not-satisfied
Continuous-authentication	Satisfied	Satisfied
Multi-level-unlocking	Not-satisfied	Satisfied

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 Confidentiality.

It is important to note that confidentiality 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 modification would affect data availability. Simply disconnecting the sensor would not affect 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 identified which is due to the fact that using this model the decryption key needs to be stored on the device. An attacker which has hardware access could therefore extract the key and decode the data. The original Picosiblings solution circumvented this approach by keeping

4.3.2 Dedicated device with shared sensors

We will relax the original set of assumptions by saying that the communication path with the sensors is no longer secure. Furthermore the sensors may be shared with other owners, via a wireless communication link for example. Another feasible scenario is that although sensors are located on the same device as the Pico, the Pico application is fully compartmentalised from the outside world.

What we are trying to stress with this scenario is that the sensors are no longer part of a trusted secure box, but are outside and communication with them, as well as their input may no longer be secure.

Availability

Since the sensors are no longer dedicated, other users may access sensor data. Depending on the hardware and software platform supporting the sensors, this may lead to a denial of service attack on the scheme. For example, if the sensors may only have one owner at a time, an attacker may request data from all sensors keeping them locked from the biometric authentication mechanisms. If the system is built in such a way, then there

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 confidentiality 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 modified 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 fit 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 confidential. 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 confidentiality 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 confidentiality breach.

Let us briefly discuss any issues using the availability-integrity-confidentiality framework.

Availability

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

Integrity

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

Confidentiality

No known attacks on data confidentiality 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: fingerprint 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 briefly, 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 different 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 defined 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 on 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 confidence 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 confidence 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 confidence 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 defines 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 confidence 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 offer a wide range of sensor data such as GPS, accelerometer, camera, and microphone.

Based on the sensor data offered 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 predefined list of mechanisms offering continuous authentication as well as explicit.

A number of continuous authentication mechanisms may be developed using solely the standard sensors offered 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 offered 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 confidence 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 offered 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 confidence 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 offered by most phones. If the sampling quality is not sufficiently good, meaningful features from the iris may not be extracted, making the confidence 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. Different 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 confidence 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 configured to listen for input.

Gait recognition

This mechanism is based on the concept of analysing individual walking patterns. Different people walk in different ways, which even though may not be entirely unique for every individual, would still provide some confidence level regarding the user of the device. In the lack of an existing reliable library, efforts 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 offers 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 notified 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 fingerprint 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 offered 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 files 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 predefined compatible configuration listed in the following listing:

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

The minimum buffer 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 modified. When a request is made by UAService, the value is returned using the IBinder message passing mechanism offered 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 confidence level is outputted for the current sample. Just as the scheme proposes, this confidence 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 briefly present the changes made when porting the Javafaces library. The "BufferedImage" class had to be replaced by its Android counterpart, Bitmap. All BufferedImage references and initialization had to be changed. The API was modified to support direct Bitmap input in order to add more flexibility 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 reflect the Bitmap convention where all 3 colour channels are present but have the same value. Additional modification 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 offers 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 fixed time interval, a thread running within the "FaceService" object samples data from the camera at a fixed time interval. This is performed using an instance of the Camera class. Additional configuration 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 notification 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 difficult. The solution used with this prototype was to exploit Android's option to render the preview image to a "SurfaceTexture" object. This satisfies 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 fix this issue, Bitmaps collected from the camera are scaled to 50

5.2.3 Owner con guration

In order to configure the biometric authentication mechanisms in a flexible, 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 confidence 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 files 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 different threat models. This will reveal any flaws 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 fits, 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" offered by different mechanisms such as Intents and filters based on these intents.

An interesting paper by William Enck et al [17] offers 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 file 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 fixed entry point such as a main() routine. Based on its purpose, the SDK defines 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 defined 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 affect 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 defines how apps are allowed to access different components.

Components may be defined as either public or private. This refinement is configured by the "exported" field defined in a manifest file. It defines whether or not another application may launch or interact with a component from another application. At the time the paper was written, the "exported" field was defaulted to "true". However, as shown in a more recent paper [18] written by Steffen 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 affect the flow 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 predefined 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 confidential data. A developer may therefore construct pairs of legitimate applications in order to mask a data flow attack.

The LA system offers a number of memory corruption mitigations in order to avoid buffer overflow 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 first 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

offset 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 Steffen 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 configuration of LA by introducing custom modifications before they roll out a device.

3. The Android permission model is defective. As pointed out by Steffen 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 flow 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 predefined 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 different 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 modifiable. 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 confidentiality 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 different data may be obtained without the owner's knowledge. Additional improvements can be made in order to increase the confidence 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 different 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 different 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 fixed 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 figure out how to present meaningful results for different 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 sniffing 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 different 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 notification, allowing for the owner to decide which application acquires the lock. A suggestion to this approach would be to allow for different 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 definition offered by the authors "gait recognition describes a biometric method which allows an automatic verification 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-off. In *Advances in Cryptology-CRYPTO 2003*, pages 617–630. Springer, 2003.

[4] Jeff 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 fingerprint 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 effort in verification. 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] Steffen 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-sniffing 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 identification. *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.