Designing Secure & Usable Picosiblings

An exploration of potential pairing mechanisms

Fabian Matthias Andre Krause

Wolfson College

14th June 2014

Dissertation

MPhil Advanced Computer Science (Option B)

Easter Term 2014

Supervisor Dr. Frank Stajano (University of Cambridge)

by: Fabian Matthias Andre Krause
Email: fabianmakrause@me.com
Course of Study: MPhil Advanced Computer Science (Option B)

University: University of Cambridge
Department: Computer Laboratory
Address: 15 JJ Thomson Avenue, Cambridge CB3 0FD

College: Wolfson College
Address: Barton Road, Cambridge CB3 9BB

UNIVERSITY OF CAMBRIDGE

WOLFSON COLLEGE CAMBRIDGE

Statement

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

Place, Date	Signature

Word count:
14970

Word count retrieved using *TeXcount*[1].
from p. 1 excluding appendix[2], bibliography, and figures, but including tables and footnotes.

[1] TeXcount 3.0.0.24: http://app.uio.no/ifi/texcount/online.php. Used commands *%TC:group tabular 1 1* and *%TC:group table 0 1* to include tables. Figures were manually removed from the *TeX*-file.
[2] As stated in III.c of http://www.cl.cam.ac.uk/teaching/acs_projects/proj-guidelines.html, appendices can be excluded from the word count if they merely consist of supporting or verbose material that is not directly relevant for understanding the thesis. This is the case in this thesis.

Abstract

The security of token-based authentication systems relies heavily on their resilience to theft and loss. The Pico's approach to preventing user impersonation are Picosiblings: small devices the user carries that are in permanent contact with the Pico. Outside the Picosiblings' range, the Pico locks itself, thereby preventing abuse.

Protocols for communication between the Pico and its Picosiblings have previously been proposed, unlike usability and management factors. Bootstrapping the security relationship is crucial to security and usability.

A comparative usability analysis was performed between the most promising Picosibling pairing mechanisms. The findings inform the design of a system that avoids common usability flaws and promotes an appealing user experience. Users were found to prefer scanning QR codes over typing passwords, using cables, and pairing inside a Faraday cage. Furthermore, protocols were developed for the addition and removal of Picosiblings, periodic resharing, and the integration of backups. Combined, the protocols provide the foundation for an integrated, secure, and usable Picosibling design.

The subsequent development of a management framework can build on these results and benefit from the findings of this thesis. Further user studies will benefit from the developed Android prototype that includes the major protocols described in this thesis, allowing for a more realistic user experience.

Keywords

Pico, Authentication, Token-based authentication,
Usability, Android prototyping,
Pairing, Bootstrapping, Secret Sharing

Acknowledgements

Firstly, I wish to thank my supervisor, Dr. Frank Stajano, who guided me through this challenging project. He hinted me at material to close my gaps in knowledge and motivated me to complete this project. I further thank Jeunese Payne and Dr. Alan Blackwell. They taught me the basics of HCI and helped me to design, execute, and evaluate an experiment in a field that I was unfamiliar with prior to this project. I also thank the whole Pico team for the valuable insights and discussions during our supervisions.

I extend gratitude to Mason Gordon, Kamleshan Pillay, Victoria Quast, and the Cambridge language centre's director of English for Academic Purposes (EAP), Dr. Karen Ottewell, for proofreading parts of my thesis.

Furthermore, I thank the developers of ZXing, whose QR code scanner I integrated in my implementation, and the *stackoverflow*[3] community. Even though I have never asked them for help, their feedback on other's problems allowed me to understand the underlying basics of the Android environment.

[3]http://stackoverflow.com

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BEDA	Button-Enabled Device Association
d2d	device-to-device
DoS	Denial of Service
GPS	Global Positioning System
h2d	human-to-device
HMAC	Hash Message Authentication Code
LED	Light Emitting Diode
MAC	Message Authentication Code
MANA	Manual Authentication
MiB	Message in a Bottle
NFC	Near Field Communication
OOB	Out of Band
PAKE	Password Authenticated Key Exchange
PGP	Pretty Good Privacy
PIN	Personal Identification Number
POS	Point of Sale
QR Code	Quick Response Code
RFC	Request for Comments
RFID	Radio-Frequency Identification
SiB	Seeing is Believing
SAS	Short Authenticated Strings
SMS	Short Message Service
SUS	System Usability Scale
UDS	Usability-Deployability-Security Framework

Definitions

Host	The object with which the Picosibling is incorporated.
Pairing	Associating two objects with each other, e.g. by exchanging cryptographic keys.
QUASI NO-TYPING	A Pico setup that requires entering a passkey only for setup operations but not authentication.
Secure core of the Pico	The part of the Pico that contains the credentials of the user that are encrypted under the master-key
Outer core of the Pico	The part of the Pico that contains the key material to communicate with the Picosiblings and to recreate the master-key

1. Introduction

Besides its predominance, the quality of passwords as an authentication scheme is dubious. Morris and Thompson [55] found in 1979 that password security depends on the correct implementation and the user's behaviour. Users circumvent cumbersome password guidelines to mitigate mental overhead (Florencio and Herley [29]) and to focus on their primary tasks (Adams and Sasse [1]). Furthermore, service providers' implementations are often flawed (Bonneau and Preibusch [13]) leading to data leaks - as seen in the case of *LinkedIn* [5].

Stajano [77] proposed the Pico as a token-based authentication scheme that strives to overcome these shortcomings. However secure an authentication scheme, it has to provide other usability or deployability benefits to become an alternative to passwords (Bonneau et al. [14]). The Pico's approach to usability is predicted on reducing users' mental overhead.

The security of token-based authentication schemes[4] relies heavily on its resilience to theft and loss [14]. When a token is stolen, the perpetrator should be prevented from accessing a user's credentials. The Pico's approach to preventing loss comprises an innovative self-locking mechanism that becomes activated when the user is not in the immediate proximity. A set of Picosiblings, small devices incorporated in a user's belongings, enables "telepathic locking". Whenever a sufficient number of Picosiblings are not in proximity to the Pico[5], the Pico locks its secure core and wipes the key to it from memory.

Picosiblings are based on threshold security, a concept with which the majority of users are not familiar. It is essential to the success of Pico that the design space of handling, managing, and using Picosiblings is explored thoroughly prior to deployment to prevent the Pico's benefits from being compromised [8, 77, 81]. Usability studies are to explore users' attitude towards Picosiblings, their understanding of the underlying concepts, and develop an overall usable system. The focus of this thesis lies in the management of Picosiblings.

I will explore the protocol design space of managing the set of Picosiblings from a security *and* usability point of view.

[4]The Usability-Deployability-Security (UDS) framework of Bonneau et al. [14] discusses the benefits of token-based authentication systems.

[5]Picosiblings implement a form of threshold security such that a threshold of k out of a set of n Picosiblings have to be in close proximity of the Pico to unlock it.

1.1. Contributions

Following a discussion of background material to understand this thesis (section 2), I will analyse which existing pairing mechanisms are suitable for associating the Picosiblings and the Pico through an

- *extensive literature review* (section 3).

Such an analysis accounting for the constraints and goals of the Pico project has not been done before and will allow for a narrowing down of the set of potential pairing mechanisms.

The most appropriate pairing mechanisms will be analysed within a

- *comparative usability study* (section 4)

which accounts for the overarching goal of the Pico to outperform passwords in terms of usability. The study captures the preferences of potential users and their thoughts on the pairing process to shape future implementations. Further, it leads to the

- *development of secure and usable protocols for managing Picosiblings* (section 5).

These protocols are implemented as an Android prototype (section 6) to allow the Pico project to build on my work in later stages, e.g. by using the prototype in future user studies. While an Android prototype had been developed before, it focused primarily on the communication between Pico and Picosiblings and not the management process.

Besides from the above, protocols for removing, resharing, and adding Picosiblings as well as for integrating the management of Picosiblings into the ideas for backups are presented.

2. Theoretical Background

The following sections introduce the main concepts on which this thesis is building. For an explanation of the used notation see appendix A.2.

2.1. Introduction to the Pico

Stajano [77] developed the Pico to reduce the mental burden on the user of remembering an increasing number of passwords. Users can use the Pico to store all of their credentials and access their accounts online as well as offline. As a clean slate design to replace passwords its main goals are to be *memoryless*, *scalable*, and *secure*. It is *memoryless* in that users do not have to remember any password. It is *scalable* in that any number of the user's accounts can be effortlessly accessed via the Pico. And it is *secure* by using multi-level security (use of a visual channel as well as radio) and providing continuous authentication.[6]

Continuous authentication between the Pico and a service is accomplished by using a short-range radio protocol after the user has initially logged in. The initial login requires users to scan a QR code of the service with their Pico. The scanning of the QR code enables the Pico also to authenticate the application or service before providing the user's credentials. For an extensive discussion of the communication protocol between Pico and application see the work of Bentzon [8].

The user's credentials are stored securely inside the Pico and can only be accessed if the Pico is unlocked with a master key. This master key is shared among a number of Picosiblings that form a presence aura around the user. Thereby the Pico implements "family feelings" as introduced by Stajano [76]: The Pico only unlocks if a sufficient number of Picosiblings are in close proximity.

2.2. Picosiblings

A major problem of token-based authentication is the uncertainty manifesting regarding whether the person in possession of the token is also the legitimate owner. Picosiblings are the Pico's way of approaching this uncertainty and to realise theft-resistance as well as loss-resistance by what is called *telepathic locking*. Whenever there is an insufficient number of Picosiblings near the Pico, the Pico locks itself and wipes the master key securely from memory.

The master key is secured through a k-out-of-n secret sharing scheme in which the Picosiblings are the shareholders that have to collaborate in order to recreate the key. To ascertain the presence of a sufficient number of Picosiblings, the Pico *pings* Picosiblings in its surrounding, prompting them to send their share. Internally, the Pico is implemented to have a decay timer for each Picosibling which triggers the ping protocol.

[6] A more thorough discussion of its security benefits can be found on page 6 of Stajano [77].

It is envisioned that Picosiblings are digitally enhanced objects users wear or carry with them. Picosiblings interact with the Pico over short-range radio. They are designed without a user interface to allow for embedding them completely in their hosts. Two special shares can enhance security even further: a share that is stored on a remote server and a share yielded from biometric scans. These special shares have more weight and could be explicitly required for certain actions or to use the Pico at all. They could allow remote deactivation of the Pico, tracing online login attempts, and special use cases.

Stannard and Stajano [82][81] developed the *ping*-protocol between Pico and Picosiblings to ascertain the presence of Picosiblings and to acquire their shares. Bentzon [8] added a check that prevents Picosiblings from sending their share if only their presence is requested.

The Pico sends a nonce N_i, the current value of a counter c_i, and a boolean value "$KReq$" that indicates whether the Pico requests the share of the ith Picosibling. This message is encrypted under the shared symmetric key $K_{P,PS_i,1}$. A MAC of the resulting ciphertext completes the Encrypt-then-MAC mechanism used in this step.

$$P \rightarrow PS_i\text{: } \{N_i, c_i, KReq\}_{K_{P,PS_i,1}}$$

Consequently, if the MAC authenticates the message, the ith Picosibling sends back the nonce and its share (if requested). The share is encrypted under an ephemeral key $K_{P,PS_i,2}$ that is refreshed by hashing it according to the counter.

$$PS_i \rightarrow P\text{: } \{s_i\}_{K_{P,PS_i,2}}, N_i$$

This is the protocol described by Stannard [81].

2.2.1. Pairing of Picosiblings

A crucial step in using Picosiblings has not received much attention thus far: setting up the set of Picosiblings and managing them over the lifecycle of the Pico. Originally, a variant of the resurrecting duckling protocol of Stajano and Anderson [78] was proposed to bootstrap the relationship between Pico and Picosiblings. Undoubtedly, pairing[7] is an essential part of ensuring that the Pico communicates only with legitimate devices and provides no others with a share of the master key. Stannard [81] proposed a protocol for the initial pairing process based on a physical connection between Pico and Picosibling.

$$P \rightarrow PS_i\text{: } PAIR, s_i, K_{P,PS_i,1}, K_{P,PS_i,2}, c_i$$

However, this does not account for removing Picosiblings when they are lost or stolen. A physical connection might not even be possible when the vision of embedding Picosiblings in their hosts[8] is realised.

[7]Pairing describes the process of establishing a secure association between two devices, here Pico and Picosibling.

[8]The host of a Picosibling is defined as the object the Picosibling is attached to or interwoven with.

2.3. Secret Sharing

Two properties of a secret sharing scheme are essential for Picosiblings: security and flexibility. Assuming a (k, n)-threshold scheme, an attacker with less than k shares must not be able to reconstruct the secret. Further, flexibility is needed because Picosiblings need to be unpaired from the Pico if the host of the Picosibling is sold to another person or lost.

Several secret sharing schemes have been proposed from which Shamir's [71] seems to be the most appropriate. Blakley [11] proposed a scheme based on the intersection of $(k-1)$-dimensional hyperplanes. However, this scheme is less flexible and requires k-times more space to save the shares. General access structure secret sharing schemes such as presented by Ito et al. [39] also require an increasing share size with an increasing participant number as noted by Beimel [6]. Weighted share threshold schemes[9] could model the special shares of the Pico and provide a meaningful representation of domain aspects. Nevertheless, Shamir's scheme is used here since it is ideal[10], well studied, and has beneficial security and resource properties. Further, weighting can also be modelled in Shamir's scheme by handing out several shares to one party or changing the implementation accordingly as presented by Dawson and Donovan [24].

Shamir's *(k, n)* secret sharing scheme is based upon the interpolation of polynomials in finite fields. A polynomial of degree $k-1$ is uniquely identified by k points in the 2-dimensional plane. To share a secret S, one has to choose an arbitrary polynomial of degree $k-1$: $q(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1}$ in which S is represented by a_0. The n shares s_i that are to be distributed among the involved parties can be calculated by solving $s_1 = q(1), ...s_n = q(n)$.

This scheme guarantees perfect secrecy when using finite fields. Knowledge of $k-1$ or fewer shares leaves the attacker with no information about the secret S while k shares or more allow the recomputation of S by interpolation. In addition to perfect secrecy, it also allows for flexibility to add shares and perform resharing while keeping k fixed.

2.4. Threat Model for Pico

An attacker is allowed to seize control of $k-1$ Picosiblings at a single point of time as they do not reveal sufficient information to recreate the shared secret. Furthermore, the adversary can obtain possession of the Pico when it is locked. The adversary is even allowed to obtain possession of the Pico when it is unlocked. In this case, he or she must not be in possession of a sufficient number of Picosiblings nor able to use the Pico in time to access the user's accounts before it locks. The attacker can have control over the computer used for logging in the user, the remote service, and the wireless channel.

[9]For a discussion of weighted threshold schemes and their information rate properties see Morilla et al. [54].
[10]A secret sharing scheme is ideal if its information rate is equal to 1. See Stinson [83] for more details.

Re-pairing attacks as described in Stannard and Stajano [82] are countered by requesting the received shares again each time the Pico switches to pairing mode.[11]

The attacker model during pairing is the Dolev-Yao [27] model assuming an omnipotent attacker who can overhear, intercept, create, and modify any messages sent over the wireless channel before, during, and after pairing.

[11] In this case the pairing mode is the mode the Pico is in when clicking "Pair", not the state for the initial pairing.

3. Bootstrapping the Relationship between Pico and Picosiblings

In the following, the most common pairing mechanisms will be presented and analysed. Requirements and constraints of pairing Picosiblings to the Pico will be used to determine which pairing mechanisms are or could become applicable.

3.1. The Resurrecting Duckling

The resurrecting duckling security policy had been used as a way to model the Picosiblings. Its main concepts are outlined below to enhance the understanding of constraints and requirements with regard to pairing.

Stajano and Anderson [78, 79] developed the resurrecting duckling as a security policy for transient association of constrained devices in wireless ad hoc networks focusing on master-slave relationships. Stajano [76] then expanded the idea to peer-relationships.

The core idea is that a duckling (slave device) identifies its mother (master device) as the first thing it recognises visually and audibly (which sends the ignition key). This association lasts until the duckling "dies", setting the duckling back to its imprintable pre-birth state. In this pre-birth state the duckling could be sold or associated to another master device. Imprinting is to be accomplished via physical contact because of the limited capabilities of ducklings. Delegation from a position of strength and multiple souls (imprinted states) occupying one body (device) resemble the Biba integrity model [9].

3.2. Requirements Analysis

Stannard [81] has considered three requirements for pairing Picosiblings to the Pico:

- Transferring the data to run the ping protocol.
- Preventing eavesdroppers from impersonating a paired Picosibling.
- The user must know which devices are being paired, even in the presence of an attacker.

Aside from these,

- pairing must only be possible when in possession of k Picosiblings.

Furthermore, hardware constraints must be accounted for. The Pico only consists of a display, two buttons ("Main", "Pair"), a camera, and a short-range radio. Pairing mechanisms can exploit one bidirectional human-to-device (h2d) channel provided by the Pico and one device-to-device (d2d) channel between the Pico and Picosibling. Picosiblings are designed to be small and cheap. Thus, they are limited in terms of power supply and computation capabilities. The cost and size aspects further prohibit visual interfaces, accelerometers, GPS, audio input or output, and other sensors or technologies. All of these are either impractical, too expensive, or would drain the power of the Picosibling too quickly.

- Picosiblings are resource-constrained and have limited capabilities. However, realistic technological advancement can be incorporated into the design.

The design of the pairing process must be guided by the long-term vision of the Pico. Picosiblings could be incorporated into objects the user carries around [77]. If a Picosibling is contained in a watch or within a piece of jewellery, it is impractical to have a visual interface or a physical link between Pico and Picosibling.

- Picosibling might be physically inaccessible.

3.3. Existing Pairing Mechanisms

The pairing problem in wireless networks is concerned with choosing the right device out of potentially many with which to communicate. Evil twin attacks, accidentally pairing a device controlled by an adversary, and man-in-the-middle attacks are to be prevented. It is well established that some kind of user involvement is unavoidable, e.g. using an auxiliary OOB (out-of-band) channel, as stated by Kumar et al. [45]. Physical contact as proposed for the resurrecting duckling is said to defeat the benefits of wireless networks [45]. However, users are used to connecting devices to computers to set them up. For this reason, physical connection is not discarded prematurely in this thesis.

Public key infrastructures, PGP, or Needham-Schroeder (Kerberos) as authentication bases would simplify the problem drastically. Unfortunately, they are inapplicable due to the lack of internet connectivity, service availability, and non-scaling costs. The well-studied key exchange protocols like Diffie-Hellman [26] lack authentication of the communication partners in an offline scenario.

The goal is thus to find a pairing mechanism that minimises usability costs under the constraints and requirements of the Pico (see section 3.2).

Visualisation Perrig and Song [61] presented *Random Art* to encode hashes of the devices' public keys as images. Ellison and Dohrmann [28] introduced *Flags* as hash visualisations of the process output. These hash visualisations are compared by the user and require both devices to have a display. Roth et al. [66] presented a solution based on blinking lights representing colours of a short authentication string (SAS). Picosiblings do not provide such interfaces. When Picosiblings become inaccessible, sending and capturing of lights is infeasible as is providing a display.

SiB (Seeing is believing) of McCune et al. [53] fits the context better. One device has to be equipped with a camera to scan a QR-code that contains the hash of the other device's public key. The other device presents the QR code on its display. Even though Picosiblings do not provide a display, the QR code could be delivered on a tag by the manufacturer together with the host. Such a system was proposed by Hanna [34]. While *SiB* was found to have poor usability characteristics by Kumar et al. [45], Pico users will use QR codes for authentication purposes. Using a similar technique for pairing might generate synergies and enhance the system's overall usability.

The system of Saxena et al. [69] also requires both devices to display hashcodes dynamically during protocol execution. Saxena and Uddin [67] relax the display requirement by having one device supporting blinking LEDs. Picosiblings neither provide a display nor an interface to capture LED (potential inaccessibility).

Special Technologies Balfanz et al. [4] proposed *Talking to strangers*. Audio or infrared is utilised in a pre-authentication phase to transmit bits for authenticating the key. Like *Proximal Interactions* of Rekimoto et al. [64] and the idea of Swindells et al. [84], which both rely on pointing infrared devices at each other, *Talking to strangers* is not applicable because of its use of infrared. The technological requirements would force the Picosiblings to become more expensive and be accessible for visual channel support.

Picosiblings do not contain specialised hardware (costs), making approaches based on special OOB channels infeasible. Kindberg and Zhang [44] proposed to use ultrasound and lasers [43] as a means to guarantee the physical proximity of the devices. The approach of Buhan et al. [17] models trust relationships of people on their devices using biometrics. Users take pictures of each other that the devices use in their protocol for authentication. Apart from the fact that Pico and Picosibling are owned by the same person, the Picosiblings cannot capture biometrics since they lack specialised hardware. Setting up Picosiblings to be able to send and receive *SMS* to exchange secret information as in *LoKey* of Nicholson et al. [56] is also impractical and expensive. Further, vibration (or measuring vibration by the Picosibling) is not supported which is required by Saxena et al. [70] in their *PIN-Vibra* to transfer secrets over the vibration OOB channel.

MiB of Kuo et al. [46] requires a Faraday cage and a keying beacon for pairing. Pico and Picosibling are placed in the cage to perform a pairing protocol while the keying beacon outside the cage jams signals and serves as an indicator as to whether the cage is closed or not. This initially suggests a cost and usability overhead as users would have to own a Faraday cage and remember to place the keying beacon outside the cage for pairing. However, the Faraday cage allows the pairing of several Picosiblings simultaneously and requires only an initial investment. As the Pico becomes widely used, the cost of Faraday cages would drop and even low-cost alternatives shielding electric fields could be explored.

Audio Methods using audio to transfer key information as by Soriente et al. [74], audio-based OOB channels for secure transmission of key information as by Goodrich et al. [32], or audio for SAS validation to complete the protocol as by Goodrich et al. [33] encounter the same constraint the other additional technologies encounter. Audio-visual pattern recognition as by Prasad and Saxena [62] requires additional hardware and physical access to the Picosibling. Assuming the cost of the additional hardware for Pico and Picosiblings was acceptable, the Picosibling's host could potentially block audio signals and compromise the reliability of these approaches. Audio-based approaches also require the user to push buttons on the Picosibling to validate or initiate the protocol.

Shaking and Gestures Acceleration-based pairing methods using shaking are inapplicable because Picosiblings might be embedded in valuable hosts that could break if the user slips while shaking. Hosts could further be bulky, which makes shaking cumbersome and could lead to bad pairing input. Moreover, the Picosibling needs to be equipped with a 2-axis-accelerometer. Mayrhofer and Gellersen [52] developed a pairing protocol based on users shaking the devices together. *Smart-Its* of Holmquist et al. [38] recognise their "friends" by comparing accelerometer data. Hence, a user shakes both Smart-Its to establish a connection. Another approach by Castelluccia and Mutaf [21] relies on shaking devices to randomise the reception power and thus to incrementally establish a secret by transferring bits. It does not require additional hardware but is questionable in terms of usability since the devices have to be shaken randomly until a secret is established, which can take minutes. Gesture-based pairing mechanisms underly similar constraints as they require either accelerometers or audio capabilities. Hinckley [37] developed a pairing mechanism that allows bumping devices together to create a shared secret while Peng et al. [60] created a protocol that identifies the device to pair with by pointing at it and calculating the reception times of audio signals.

Buttons In an effort to create a pairing method that does not require special hardware, *BEDA* by Soriente et al. [73] and other button-based techniques by Rekimoto et al. [63] and Iwasaki et al. [40] have been developed. They require the devices to only provide buttons for pairing. Users click the buttons synchronously or when prompted. Even this minimal interface is not guaranteed for potentially inaccessible Picosiblings. Other variants that only require one device to provide a button require the other device to provide sensors, accelerometers, or LEDs. Similarly, *RhythmLink* by Lin et al. [49] prompts the user to tap a sequence or melody. Besides the Picosibling's lack of a button, microphone, and sensors required for *RythmLink*, users have to remember the tapped sequence, defeating the Pico's benefit of being *memoryless*.

Proximity Methods based on proximity estimate the distance between devices or use an additional near frequency channel, e.g. NFC. The issue of the former lies in its increased computational effort that the Picosiblings have to cope with and the increased duration the user has to have Pico and Picosibling in close proximity to reduce error rates and attacks. Varshavsky et al. [88] approach the pairing problem by comparing the devices' radio environment in their *Amigo* system, Rekimoto et al. [65] utilise the signal strength of the pairing devices, and Mathur et al. [51] establish a key based on the shared fluctuations in the wireless environment. Switching to near frequency channels only shifts the pairing problem to another medium. While it can be argued that pairing Picosiblings using NFC is a neat idea, eavesdropping – for example by using an antenna – cannot be completely prevented. Furthermore, this would require Picosiblings to support an additional communication channel which will likely increase their costs. Distance bounding could also be achieved by using ultrawide-band or ultrasound as shown by Cagalj et al. [19].

Amariucai et al. [2] proposed the idea of an RFID-tag as an *Adopted Pet* that is imprinted by spending a long time in range of its owner (the reader). For the Pico, this period might be too long because users always need to be able to access their credentials. *Noisy Tags* by Castelluccia and Avoine [20] utilise noise tags that are placed near the devices the user pairs to generate sufficient noise such that an adversary cannot identify the messages used for pairing.

Short Authenticated Strings (SAS) approaches introduced by Vaudenay [89] and expanded by Pasini and Vaudenay [58] include “Compare-and-Confirm”, “Copy-and-Confirm”, and “Select-and-Confirm” of checksums. They all require a display and “Copy-and-Confirm” even an input device to copy the other device’s checksum. Since the user’s involvement is high in these cases, a rushing user [25] who is task-focused could defeat the security benefits of “Compare-and-Confirm” while “Select-and-Confirm” and “Copy-and-Confirm” were found to be too cumbersome to use in a usability analysis by Uzun et al. [87]. Saxena and Uddin [68] analysed the usability of number comparisons and colour comparisons with regard to rushing users. Dynamic schemes that require the user to check results on both devices are not suitable for the Pico’s case.

MANA (manual authentication) schemes MANA I-IV by Gehrmann and Mitchell [30] and Gehrmann and Nyberg [31] have different requirements on the devices and rely on user involvement in the form of entering check values and confirming the success of the protocol (MANA I). MANA I is not appropriate for Picosiblings as it requires one device to have a display and the other to have a keypad that the user can use to enter a check-value displayed by the other device. MANA II can be executed by devices that both have output interfaces and a simple input interface for the user to confirm success. MANA IV of Laur and Nyberg [47] depends on the user comparing strings on displays. Lastly, MANA III requires both devices to have a keypad to enter a passkey generated by the user as well as a way for the user to confirm success. Chong et al. [23] provide an overview over MANA schemes.

Even though Picosiblings have neither a keypad nor a way of indicating the success of the protocol, a variant of MANA III as proposed by Jakobsson [41] allows it to run without the user confirming the success of the protocol. The shared secret is split and shares are only revealed in succeeding protocol rounds. This idea is similar to Bluetooth’s Simple Pairing [12].

A "Copy" variant of this scheme presented by Uzun et al. [87] might be appropriate. The user only copies the passkey that was delivered together with the Picosibling.

Physical Connection Similar to physical connection as suggested by Stajano and Anderson [78], *tranSticks* of Ayatsuka and Rekimoto [3] allow cable-like connections between two devices. Pre-paired tranSticks are connected to each device thereby establishing a wireless connection between the devices. A variant of this could be suitable for the Pico in that the Picosibling is pre-paired to a tranStick that the user connects to the Pico to complete the pairing.

Using the body as a physical connection to transfer data as in *TAP* of Park et al. [57] or the approach of Soriente et al. [75] would, similar to the connection via cable, be an approach that works until the Picosiblings are inaccessible. Personal area networks (Zimmermann [90]) have limited bandwidth and the user would have to connect the Pico and Picosibling for a longer time period. Other than the time-factor, it might also raise user concerns or interfere with other devices worn on or inside the user's body.

Discussion Typing a passkey into one device violates the *no-typing* benefit of the Pico. For an initial setup this violation might be acceptable which is why setups that require typing for the initial pairing are referred to as *quasi-no-typing* and are not discarded.

This section illustrates why most pairing categories fail to model pairing between the Pico and Picosiblings adequately. Capabilities and the interface of Picosiblings are limited. Dynamic checksum comparisons or sophisticated sensor technologies are further infeasible leaving the OOB-channel to be either a physical channel, if we relax the constraint of Picosiblings being inaccessible, or to be a static channel that allows information exchange prior to running the protocol.

3.4. Suitable Mechanisms

As a result of the literature review, the QR code option of *SiB*, the physical connection á la resurrecting duckling, entering a passkey, and the usage of a Faraday cage as in *MiB* are chosen to be explored further in user studies.

Physical connection, despite its obvious disadvantages, can be beneficial to use in the early stages of Pico's deployment, although it might become obsolete when Picosiblings become inaccessibly interwoven with their hosts. It has low computational demands on the Picosiblings and is similar to other use cases with which the user is familiar. Furthermore, it can be substituted by a *tranSticks* variant for inaccessible Picosiblings as discussed in the previous chapter.

Protocols for each of the chosen mechanisms are outlined below.

3.4.1. Physical Contact

After establishing a physical connection the data is exchanged in a manner similar to that proposed by Stannard [81]:

$$\begin{aligned} P \rightarrow PS_i: &\quad PAIR \\ PS_i \rightarrow P: &\quad desc_i \end{aligned}$$

Fresh key material and a share s_i for the Picosibling are created by the Pico. The Pico stores the description[12], a distinctive ID for the Picosibling, the x-coordinate x_i of the share s_i, the key material $K_{P,PS_i,1}, K_{P,PS_i,2}, c_i$, and a slot used for nonces of the ping protocol. It then sends the Picosibling the required key material together with the counter and the y-coordinate y_i of the share[13]:

$$P \rightarrow PS_i: \quad PAIR,\ y_i,\ K_{P,PS_i,1},\ K_{P,PS_i,2},\ c_i$$

3.4.2. QR Code Scanning

The user scans a QR code of the Picosibling that contains a code identifier, the hash of the Picosibling's public key $h(K_i)$ and a secret to authenticate the Pico to the Picosibling S_i.

The Pico prompts all unpaired Picosiblings in its environment to respond to a pairing broadcast.

$$P \rightarrow PS_i\ \forall i \in PS: \quad PAIR,\ N_i,\ K_P$$

Picosiblings that have not been paired respond by sending their public key together with the received nonce and a new nonce encrypted under the Pico's public key K_P.

$$PS_i \rightarrow P: \quad K_{PS_i},\ \{N_i, N_j\}_{K_P}$$

To authenticate the received public key of the ith Picosibling, the Pico hashes it and compares it to the one contained in the QR code. If they match, the Pico knows that it is communicating with the legitimate Picosibling[14].

The Pico hashes the secret from the QR code of the Picosibling together with a nonce to authenticate the Pico to the Picosibling. A newly created session key $K_{P,PS_i,1}$, which is sent encrypted under the Picosibling's public key, encrypts this hash. The Pico uses the received nonce N_j to link the message containing the secret to the public key of the Pico that has been used in the previous message.

[12]See section 3.5 for more information regarding the description.

[13]Section 5.3 explains the idea behind only sending the y-coordinate of the share.

[14]Here, legitimate means: The one that is associated with the QR code scanned by the user. If the user is deceived into scanning a bogus code, the Pico would authenticate a Picosibling that is not legitimate in the traditional sense.

$$\begin{aligned} P \rightarrow PS_i\text{:} &\qquad \{K_{P,PS_i,1}\}_{K_{PS_i}} \\ P \rightarrow PS_i\text{:} &\qquad \{h(S_i, N_j)\}_{K_{P,PS_i,1}} \end{aligned}$$

The Picosibling can now authenticate the legitimate Pico by hashing the secret S_i together with the nonce N_j and comparing the result to the received string from the Pico.

If this comparison was successful, the description of the Picosibling as well as the remainder of the key material can be exchanged. Encrypt-than-MAC can increase the efficiency and integrity of the final messages.

$$\begin{aligned} PS_i \rightarrow P\text{:} &\qquad \{desc_i\}_{K_{P,PS_i,1}} \\ P \rightarrow PS_i\text{:} &\qquad \{y_i,\ K_{P,PS_i,2},\ c_i\}_{K_{P,PS_i,1}} \end{aligned}$$

3.4.3. Pre-defined Passkey

This protocol builds on password-authenticated key exchange (PAKE) by Bellovin and Merritt [7], Boyko et al. [15], and Katz et al. [42]. Yet a variant of MANA IV would also be applicable (see section 3.3).

A hard-coded passkey is printed on a label and delivered together with the Picosibling. Users enter the passkey into an auxiliary device that passes the passkey to the Pico. The Pico can display the passkey to the user to implement another security check preventing an adversary from corrupting the communication channel between auxiliary device and Pico[15].

Pico and Picosibling then execute a password-authenticated key agreement protocol after which both generate a symmetric key $K_{P,PS_i,2}$ to communicate with each other. Thereafter the final stage of the previous scenario is executed.

$$\begin{aligned} PS_i \rightarrow P\text{:} &\qquad \{desc_i\}_{K_{P,PS_i,2}} \\ P \rightarrow PS_i\text{:} &\qquad \{y_i,\ K_{P,PS_i,1},\ c_i\}_{K_{P,PS_i,2}} \end{aligned}$$

To prevent offline attacks, an encrypted key exchange protocol has to be used, as discussed by Bellovin and Merritt [7].

3.4.4. Faraday Cage

Kuo et al. [46] introduced the *MiB* protocol based on a Faraday cage. A keying device (Pico) supplies the device to be paired (Picosibling) with the key material inside the Faraday cage. A keying beacon exchanges heartbeat messages with the keying device and ensures that the cage is closed. Once closed, the beacon jams all outside signals.

For the user study, a simplified version of this protocol was used that omitted the keying beacon and used a box as the Faraday cage. The user places all Picosiblings to pair as well

[15] An attacker could still compromise the auxiliary input device and perform the pairing before the Pico had the chance to pair the Picosibling.

as the Pico in the Faraday cage. Within the cage, the *MiB* protocol is run exchanging keying information. The Pico indicates the success of the pairing process by displaying a success message or emitting an alarm sound. This indicator prompts the user to extract the devices from the cage. Once extracted, the final message to complete the *MiB* protocol is sent and a common key $K_{P,PS_i,1}$ established between the Pico and each of the Picosiblings that had been in the cage.

The key material is exchanged similarly to the former protocols.

$$\begin{aligned} PS_i \rightarrow P\colon &\quad \{desc_i\}_{K_{P,PS_i,2}} \\ P \rightarrow PS_i\colon &\quad \{y_i,\ K_{P,PS_i,2},\ c_i\}_{K_{P,PS_i,1}} \end{aligned}$$

3.5. Management Options for Picosiblings

A management layer between Pico and the user is needed to identify Picosiblings uniquely in case they are lost or stolen. Otherwise, the removal of Picosiblings outside the user's possession is impossible.

An auxiliary application can be used to manage the Picosiblings. Picosiblings paired via QR codes or passkeys could further be managed by keeping the QR code/passkey and scanning/entering it again for management purposes. The latter approach requires users to keep all the QR codes/passkeys of their Picosiblings. Users can also remove Picosiblings by re-establishing the physical connection between Pico and Picosibling in the case of the physical connection or the Faraday cage, unless the Picosibling is lost or stolen.

Application The auxiliary application can be run on a user's device. If the docking station provides I/O, it can also be used. Initial association of the Pico to such an auxiliary technology could be established through a physical connection. A device's internet connection can be exploited by the Pico to validate the certificate of the application and to set up the remote and backup server.[16]

During pairing, the user has to enter distinctive names for Picosiblings to allow management of them in later stages. Aiding this, the Picosibling sends the Pico a manufacturer-provided description of its host. This description is sent to the application by the Pico and displayed to users giving them the opportunity to accept it or to alter it. After prompting the user to confirm the displayed description on the Pico, the Pico continues with the pairing protocol.

Picosiblings never communicate with the application directly but use the Pico as a proxy. This adds a layer of indirection that allows for integrity checks and enhances overall security. The application equally never receives any secret information besides from descriptions of Picosiblings.

[16]See 5.4, 5.5.

4. User Study

A user study was performed to evaluate which of the chosen pairing mechanisms are the most suitable for the Pico project in terms of usability. The goal of this study was to capture the perception of a potential target group for the Pico and analyse their feedback for each of the mechanisms. This feedback helps to not only choose a pairing mechanism that will be widely accepted but also to adjust the design of this mechanism in a way that appeals to the user.

4.1. Experimental Design

Apparatus The four above presented techniques were simulated on a *Samsung GT-S6500* in combination with a 13-inch MacBook Air (mid 2011). Participants were provided a mouse to minimise usability effects the Mac OS might provide. Implementations were not functional but provided an interface that guided the participants through the pairing process. The researcher was not actively involved during the tasks the participants performed. A watch, a belt, and a necklace were used as Picosiblings. Ports for establishing the physical connection were modeled with plastic modeling mass. QR-codes, passkeys, and descriptions were printed on paper and placed next to the Picosiblings. The Faraday cage was simulated using a hand-crafted box. Fig. 1 shows the setup for the user study and appendix A.6.2 contains screenshots of the interfaces and further images.

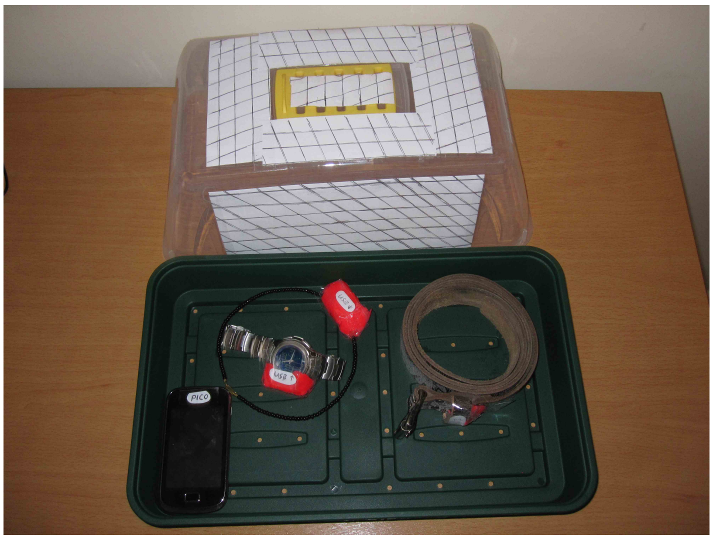


Fig. 1: Setup for the User Study

This study focused on the pairing process. Interfaces for all scenarios were modelled consistently, included safe defaults, and were implemented on the same devices. The scenarios' task completion times were automatically logged to minimise human error. Moreover, the number of steps and the security of each scenario was comparable to achieve meaningful and fair results. Passkeys for each pairing were 8 characters long, consisting of capital letters and numeric values.

Although the application is not the only option to manage the set of Picosiblings, it is the only that supports all use cases discussed in this thesis. To guarantee comparability of the scenarios, all of them used the application option for managing the Picosiblings.

Appendix A.5.1 presents the architecture and functions of the prototypes' implementation.

Participants 24 participants (12 male, 12 female), consisting of students from the University of Cambridge studying subjects unrelated to computer science, were recruited. Participants ranged from 22 to 32 years old (mean = 25.5). Younger participants, as those recruited for this study, are most likely to be early adopters of the system and thus might become the target group for initial deployments (see Lee et al. [48]).. The surveyed user group is potentially more experienced and comfortable with technology including new technology such as the Pico. They are more likely to adapt the auxiliary technologies developed by Stajano et al. [80] that allow an incremental deployment.

Users from different age groups might prefer different pairing mechanisms, and so future studies should explore whether the results of this thesis are transferable to other groups.

Pilot Study All of the previously discussed scenarios required the participant to confirm that the entered description was the one displayed on the Pico. This step was simplified for the study as it did not differ across the scenarios. Pilot interviews that were conducted to validate the experimental procedure revealed that the simplified version still confused participants, and so the step was omitted.

The pilot interviews further revealed that the original design of the experiment included too many steps and confused participant by switching between the phone and laptop too often. As a result, the design was altered to focus more on the pairing mechanisms by simplifying the steps involved. Further, removal of Picosiblings was no longer subject of this study as two of the scenarios used the same removal procedure while one (physical connection) used another. The final study added a pairing scenario that involved a Faraday cage.

Procedures A within-subjects design was implemented to yield results that allow comparison of the scenarios. Participants were subject to the same procedure. A short introduction made them familiar with the Pico and Picosiblings. Next, a description of the pairing scenarios was given to them on paper. Participants were asked to complete three pairing tasks for each scenario.

The scenarios modelled the pairing mechanisms from section 3.4. As a result of the pilot study they were simplified to focus on the differences and yield comparable results. Figure 2 shows the flow charts of the tasks the users performed on the Pico and in the desktop application for the QR code scenario. Appendix A.6.4 contains flow charts for the other scenarios and appendix A.6.5 shows the original flow charts of the scenarios during the pilot interviews to emphasise the differences in complexity and number of steps involved. To counter-balance order effects (training effects or exhaustion), a Balanced Latin Square design for the scenarios was used as shown in table 1.

	Scenario 1	*Scenario 2*	*Scenario 3*	*Scenario 4*
Group 1	Physical Contact	QR Code	Faraday Cage	Passkey
Group 2	QR Code	Passkey	Physical Contact	Faraday Cage
Group 3	Passkey	Faraday Cage	QR Code	Physical Contact
Group 4	Faraday Cage	Physical Contact	Passkey	QR Code

Table 1: Balanced Latin Square Design

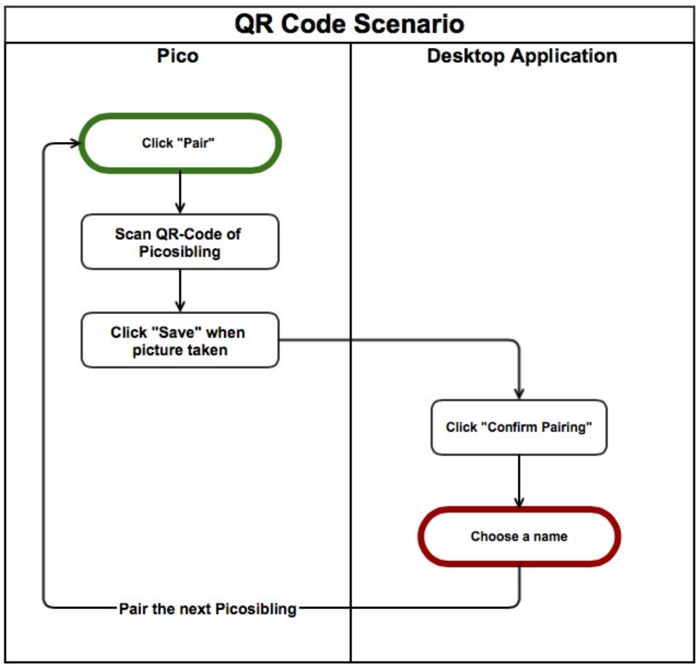


Fig. 2: Flow Chart of the User Study's QR Code Scenario

While completing the tasks, the researcher noted problems encountered, statements made, and feelings expressed. After each scenario, participants were asked to complete a questionnaire capturing their subjective perception of the process. They were also asked to identify parts of the scenario that were most intuitive to them, that they did not understand, and that were clearest to them. Participants were given the opportunity to state what they would like to have changed about the scenario. Furthermore, the System Usability Scale (SUS) [16] was used as a measure of usability for each scenario. This is a 10-item Likert scale that is widely used and reliable. At the end of the experiment the participants were asked to explain the steps involved in the scenarios in their own words. This was used to assess participants' mental model of the processes. The previous sheets of paper were removed before participants answered these questions. Finally, participants were polled on their age, gender, previous pairing experience, which scenario they perceived the most secure, and their personal preference.

The 24 interviews lasted between 31:18 and 59:18 minutes each; participants were compensated with snacks at the end of the interview. The mean duration of the interviews was 43 minutes and 4 seconds.

Usability Measures Within-subjects and between-subjects dependent variables were both collected:

- **Within-subjects usability measures**: Task Completion Time, SUS Score, Error Rate, Preference, Security Perception
- **Between-subjects factors**: Gender, Prior Pairing Experience

Time measurement started as soon as the participant first clicked the "Pair“ button on the Pico and ended once the participant confirmed the identifier of the third Picosibling in the application. While the Samsung phone and the MacBook were not synchronised during the experiment, clock deviation did not influence the validity of the time measurement as the task completion time was analysed in a comparative fashion and all scenarios were subject to the same clock deviations.

4.2. Statistical Results

The independent variables are the scenarios for pairing; the dependent variables are task completion time, SUS score, and recall error rate. Statistical significance will be reported at the 5% level represented by * or at the 1% level represented by **.

Error rates were calculated to measure mental model accuracy of the scenario. Each step of the pairing process yielded one point that had to be recalled. Forgotten steps indicate steps that were not sufficiently intuitive to remember them. The recall of the scenarios was done at the end of the experiment to separate the tasks from recalling them. However, the error rates did not yield reliable results. Recalling all scenarios after another helped participants remember steps they had forgotten before because several steps were similar across the scenarios.

For conciseness, only selected graphs are presented. Appendix A.6 contains further graphs used in the analyses. Calculations were performed using the language *R*.[17] The scenarios are abbreviated as *QR* for the QR code, *Cable* for the physical connection, *Passkey* for the passkey, and *Cage* for the Faraday cage scenario.

4.2.1. Techniques

Task completion time was normalised for evaluation to adjust for differences between the scenarios. The normalisation was performed similar to the approach of Cheng et al. [22] for information retrieval times. I executed each scenario 10 times to retrieve measurements t_1 to t_{10}. The average of these “expert” measurements $N_e = \frac{\sum_{i=1}^{10} t_i}{10}$ was used as the numerator of the normalised time measurement $N_j = \frac{N_e}{t_j}$ where t_j is the task completion time of the participant.

Using gender and experience as between-subjects factors, the scenario as the within-subjects factor, a mixed analysis of variance was performed on the usability measures (independent

[17]All scripts and generated graphs can be found in the uploaded *.zip*-file or at https://github.com/fabianmakrause/MPhil.

Scenario	*Mean Rank*	*Std. Dev.*	*Rank 1*	*Rank 2*	*Rank 3*	*Rank 4*	*Secure*
Physical Connection	2.667	1.00722	3x	8x	7x	6x	9x
QR Code	1.792	1.06237	13x	6x	2x	3x	5x
Passkey	3.000	0.83406	1x	5x	11x	7x	5x
Faraday Cage	2.542	1.25036	7x	5x	4x	8x	5x

Table 2: Preference Rating Results

Each entry represents the number of participants who ranked the scenario as their first choice (Rank 1), second choice (Rank 2), third choice (Rank 3), or last choice (Rank 4).

variables). Between all levels of between-subjects factors that indicated significant differences through the ANOVA or Friedman tests, pairwise t-tests were performed on the within-subjects measurements (task completion time, normalised task completion time, and SUS score by scenario). Friedman tests were performed in case the data could not have been normalised. Validation of the applicability of the t-tests was ensured by testing the datasets for normality. Normality tests included the Shapiro-Wilk test and a graphical analysis through QQ-plots. Non-normal datasets were adjusted via transformation when appropriate (if possible). The whole procedure is described for scenario effects on SUS scores and normalised task completion time. Afterwards only results are presented. Task completion time is reported in *milliseconds* if not otherwise stated.

4.2.2. General Results

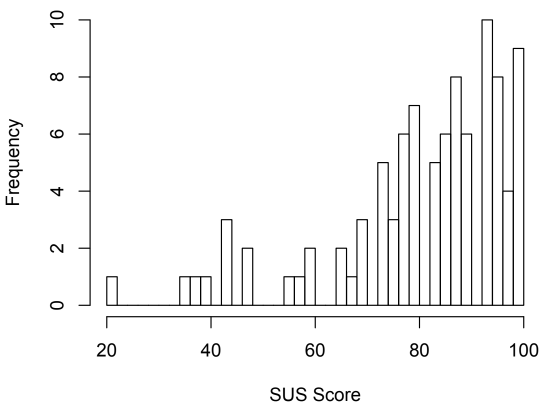
Nine participants rated the physical connection scenario as the most secure, whereas QR code, passkey, and Faraday cage each received five votes to be the most secure. This difference is not significant according to a chi-squared test ($p = 0.5724$). Neither are the differences when splitting the votes by gender or experience.

Looking at the personal preferences of the participants, the QR code scenario appears to be most appealing with an average ranking of 1.792. Table 2 summarises the results of the preference ratings showing how many participants chose which scenario as their most favourite, second favourite, etc. A chi-squared test found the difference to be significant ($p < 0.01$). Chi-squared and fisher tests[18] between all levels revealed the following significant differences:
$QR >^{*} Cable$ ($p < 0.05$), $QR >^{**} Pass$ ($p < 0.01$)
$Pass >^{(*)} Cage$ (fisher test: $p = 0.05242$, chi-squared: $p < 0.05$)

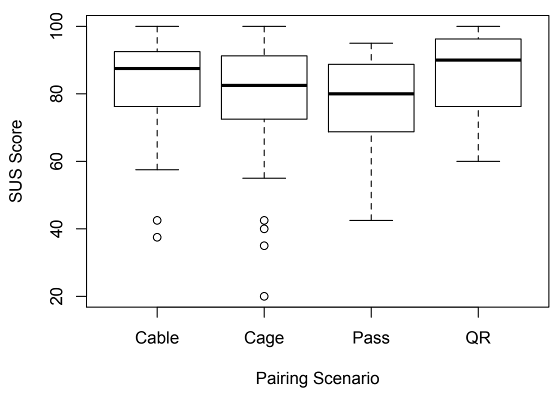
[18]The small sample size and low values in some cells (more than 20% of cells contain a value of 5 or less) require additional tests to the chi-squared test to confirm the results.

Scenario	*Mean*	*Std. Dev.*	*Min.*	*Max.*
Physical Connection	82.71	16.745	37.5	100
QR Code	86.88	11.867	60.0	100
Passkey	76.88	15.363	42.5	95
Faraday Cage	76.25	21.882	20.0	100

Table 3: SUS Scores by Scenario



(a) Histogram of SUS Scores



(b) Boxplot SUS Scores by Scenario

Fig. 3: Summary of SUS Scores

4.2.3. Scenario Effects

Effect on SUS Score General data of SUS scores by scenario is presented in table 3. With regard to the mean value of the SUS scores, the QR code was the most usable scenario followed by physical connection. However, several participants stated their strong discontent with using a cable because it is "cumbersome" and "time-consuming".[19]

The boxplot of the SUS scores by scenario shown in fig. 3b indicates possible differences between the QR code and all others as well as between the Cable scenario and the Faraday Cage/Passkey.

Fig. 3a indicates that the SUS scores are skewed to the right (ceiling effect). Shapiro-Wilk tests on SUS scores separated by scenario confirm that the data are not normally distributed ($p < 0.01$ for Cage and Cable; $p = 0.02078$ for Passkey; and $p = 0.01444$ for QR code).

[19]The absolute time the participants needed to complete the cable-scenario was not significantly higher.

Using the Box-Cox power transformation (fig. 5a), an exponent of 3.10101 is yielded to transform the SUS scores. The resulting distribution is a lot less skewed as seen is fig. 5b. Shapiro-Wilk tests for transformed SUS scores result in: $p = 0.2139$ (Cage), $p = 0.1123$ (Cable), $p = 0.2028$ (Passkey), $p = 05301$ (QR) meaning we cannot reject the null hypothesis that each of these datasets is normally distributed. While the boxplots of the transformed datasets (fig. 4) still hint at possible differences between the SUS scores by scenario, the one-way repeated measures ANOVA was non-significant ($p = 0.542$). However, the Friedman test results in $p < 0.01$. As characteristic for the Friedman test, this result is connected to a loss of power. Accepting this loss of power, pairwise t-tests for the transformed datasets show a significant difference between QR code and passkey ($p < 0.05$, $power = 67.94\%$) and borderline significance between QR code and cable ($p = 0.0617$, $power = 44.65\%$).

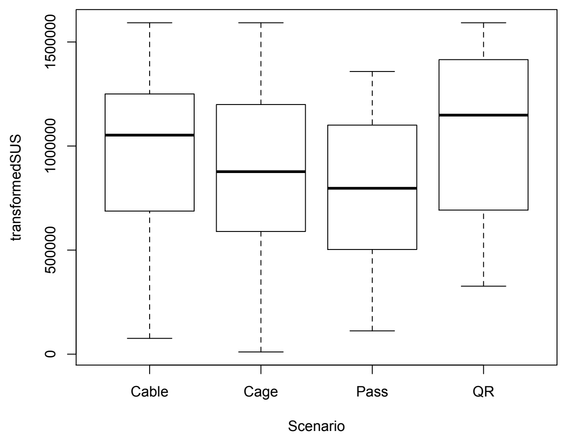
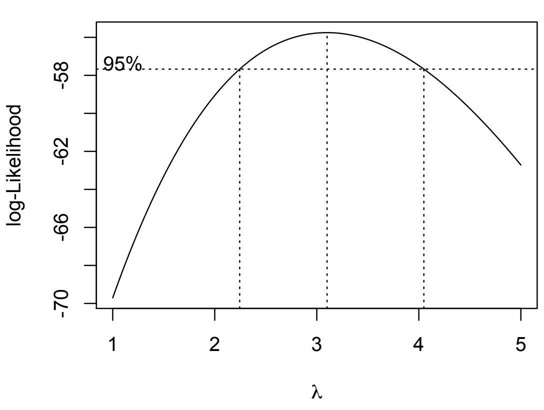


Fig. 4: Boxplot Transformed SUS Scores by Scenario

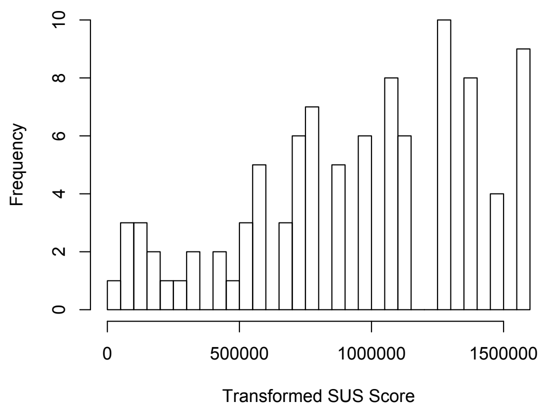
The effect of the within-subjects factor scenario on SUS scores can be summarised as follows:
$QR >^{(*)} Passkey$
$QR >^{((*))} Cable$
$QR > Cable > Cage > Passkey$

Effect on Task Completion Time As indicated by the boxplot of task completion times by scenario (fig. 6a), there was no significant effect of scenario on task completion time. Transforming the dataset by an exponent of -0.14141 did not reveal a signiicant effect either.

Effect on Normalised Task Completion Time Fig. 6b shows the boxplot of the normalised task completion times by scenario. It indicates a significant advantage of the Faraday cage scenario. However, considering the experimental design, this variation can be explained due to a design error in the measurements as the time for the Faraday cage scenario started as soon as the user pressed the “Pair” button. A minute after this, the alarm sound rang to notify that the pairing was completed. More realistic would be a measurement starting as soon as the button is pressed and waiting a minute as soon as the cage is closed. This was not possible with the experimental setup.



(a) Box-Cox Transformation Plot of SUS Scores



(b) Histogram of Transformed SUS Scores

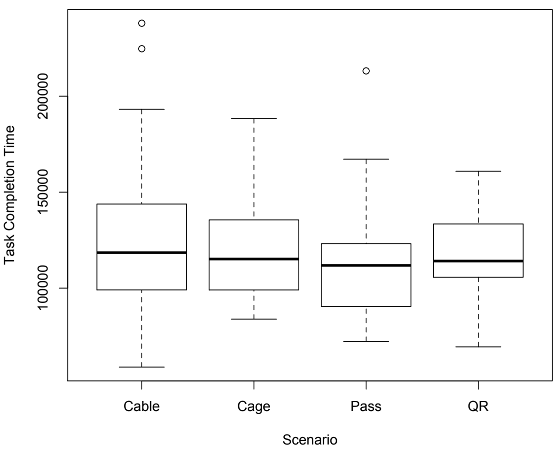
Fig. 5: Summary of SUS Transformation

	Cage	Passkey	QR Code
Cable	< 0.01** (98.33%)	< 0.05* (62.97%)	< 0.01** (62.66%[20])
QR Code	< 0.01** (93.09%)	0.49	
Passkey	< 0.01** (92.44%)		

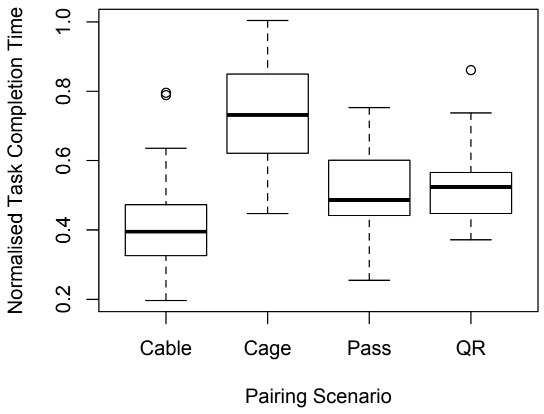
Table 4: Pairwise T-Tests between Normalised Task Completion Times

Shapiro-Wilk tests show that the data is not normal for the QR code ($p < 0.05$) and cable ($p < 0.05$) even though H_0 of the Shapiro-Wilk test cannot be rejected for the passkey ($p = 0.8716$) nor for the cage ($p = 0.6713$). The Friedman test indicates a significant effect of scenario on normalised task completion time ($p < 0.01$), which leads to a transformation by an exponent of 0.4242424 (using the Box-Cox method). Shapiro-Wilk tests and QQ-plots of the transformed datasets confirm that they can be assumed normal.

One-way ANOVA confirms the existence of significant effects within the normalised dataset ($p < 0.01$). Results of pairwise t-tests are shown in table 4 (the power of the t-test is presented in brackets behind the α-level). In summary, the effect of the within-subjects factor scenario on normalised task completion time is as follows:
$Cage >^{**} Cable$, $Cage >^{**} QR$, $Cage >^{**} Passkey$
$QR >^{(*)} Cable$
$Passkey >^{(*)} Cable$
$Cage >^{**} QR > Passkey >^{(*)} Cage$



(a) Time (in ms) by Scenario



(b) Normalised Time by Scenario

Fig. 6: Boxplot of Task Completion Times by Scenario

4.2.4. Gender Effects

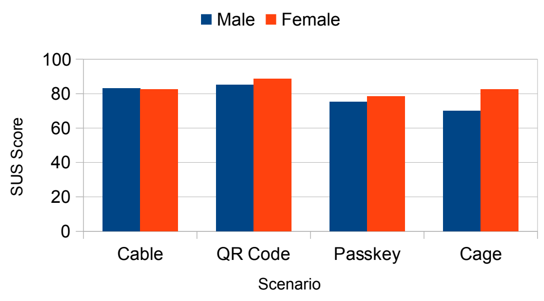
The mean values of the males' preferences ranking are: QR (1.9166) > Cable (2.25) > Cage (2.6667) > Passkey (3.1667). The difference between QR code and passkey is significant ($p < 0.05$).

Femalesʼ preferences can be summarised as follows: QR (1.6667) > Cage (2.1667) > Passkey (2.8333) > Cable (3.0833). QR codes are rated significantly better than the passkey and the cable (p < 0.05 in both cases).

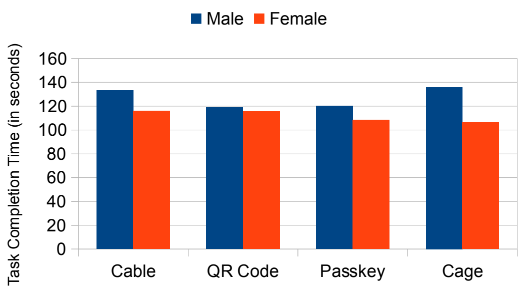
Figures 7a, 7b, and 7c show the averages of the usability factors for each scenario by gender.

Effect on SUS Score Females generally assigned higher SUS scores (mean = 83.02, sd = 14.9996) than males (mean = 78.33, sd = 18.9437), but the effect is not significant. There is a stark difference in SUS scores for the cage scenario (female mean: 82.5, male mean: 70, not significant). The QR code scores of females are higher than the passkey scores of males ($p < 0.05$, $power = 52.15\%$) and the female QR code scores are higher than the male cage scores ($p < 0.05$, $power = 46.91\%$). Borderline significance is found between the passkey and QR code SUS scores of females ($p = 0.05462$, $power = 44.61\%$).

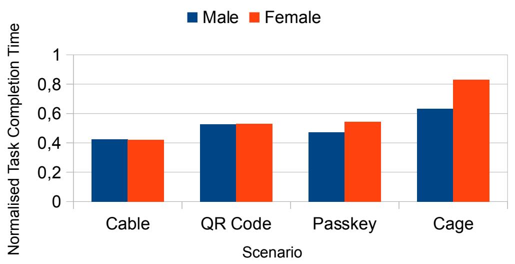
Effect on Task Completion Time The tasks were completed faster by females (female mean: 111.6s (sd = 27.476), male mean: 127.1s (sd = 36.061), $p < 0.05$, power = 57.51%). Pairwise t-tests between male and female scenario task completion times show a significant difference of the cage scenario ($p < 0.05$) confirmed by the Wilcoxon rank sum test ($p < 0.01$). The power of the t-test was at 65.56%. Males were faster completing the cage scenario than females



(a) Mean SUS Scores by Gender



(b) Mean Task Time by Gender



(c) Mean Normalised Task Time by Gender

Fig. 7: Gender Effects

completing the cable scenario ($p < 0.05$, $power = 47.93\%$), the QR code scenario ($p < 0.05$, $power = 55.82\%$), and the passkey scenario ($p < 0.05$, $power = 62.76\%$).

Effect on Normalised Task Completion Time Normalised task completion times confirm that females were faster than males. Their normalised task completion time mean is 0.5812 (sd = 0.1952), the one of the males 0.5142 (sd = 0.1547) (not significant). There was a significant effect of gender on the normalised task completion times of the cage scenario. Females were significantly faster (mean: 0.83 vs. 0.63, $p < 0.05$, $power = 80.7\%$). Comparisons of the other normalised times are shown in table 5 (power in brackets).

Males were faster at completing the cage scenario than the passkey ($p < 0.05$, $power = 84.28\%$), the QR code ($p < 0.05$, $power = 52.2\%$), and the cable scenarios ($p < 0.05$, $power = 75.01\%$). Females were also faster completing the cage scenario than the passkey ($p < 0.05$, $power = 89.97\%$), the QR code($p < 0.01$, $power = 80.15\%$), and the cable scenario ($p < 0.01$, $power = 88.7\%$). They were also faster completing the QR code scenario then the cable ($p < 0.05$, $power = 67.25\%$) and faster on the passkey than the cable ($p < 0.05$, $power = 57.23\%$).

		Male			
		Cable	QR Code	Passkey	Cage
Female	Cable	-	< 0.05 (51.83%) ↓	-	< 0.05 (93.32%) ↓
	QR Code	-	-	-	< 0.05 (61.4%) ↓
	Passkey	-	-	-	-
	Cage	< 0.05 (93.91%) ↑	< 0.05 (92.49%) ↑	< 0.01 (85.26%) ↑	< 0.05 (80.7%) ↑

Table 5: Comparison Normalised Task Completion Times by Gender
Arrow up = females performed better, arrow down = males performed better.

4.2.5. Experience Effects

Experienced users ranked the QR code on average higher than the other scenarios in the preference ranking: QR (1.7059)[21] > Cage (2.4118) > Cable (2.7647) > Passkey (3.1176). Differences between passkey and cage ($p < 0.01$), cable and cage ($p < 0.05$), cable and QR code ($p < 0.01$), and passkey and QR code ($p < 0.01$) are significant.

The preference ranking of inexperienced users does not yield any significant effects:
QR (2) > Cable (2.4286) > Passkey (2.7143) > Cage (2.8571).

Since the sample sizes for experienced and inexperienced users differ, power analyses were performed using *Cohen's d* as an indicator of the effect size.

Figures 8a, 8b, and 8c show the averages of the usability factors for each scenario by experience.

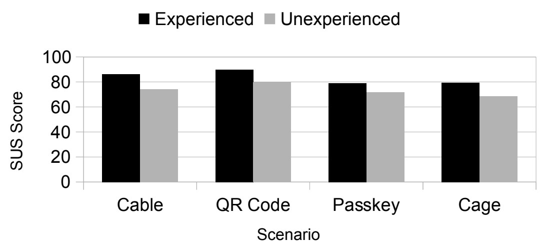
Effect on SUS Score On average experienced users assigned a 10 point higher SUS score (mean = 83.6, sd = 15.2735) than inexperienced users (mean = 73.57, sd = 19.57214). A Wilcoxon rank sum test indicates that this difference is significant at the 1% α-level.

Experienced users assigned higher SUS scores for the QR code scenario than the passkey ($p < 0.05$, $power = 59.93\%$) and the cage ($p = 0.05348$, $power = 46.82\%$).

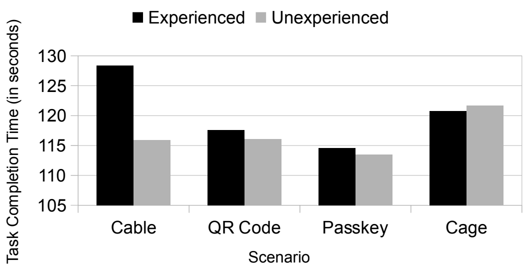
They also assigned higher SUS scores to the cable scenario than inexperienced users to the passkey ($p < 0.05$, $power = 77.48\%$) and rated the QR code higher than the unexperienced users rated the cable ($p = 0.07584$, $power = 61.35\%$), the passkey ($p < 0.05$, $power = 93.86\%$), and the cage ($p = 0.6046$, $power = 69.12\%$).

Effect on Task Completion Time There was no significant effect of experience on task completion time. Experienced users were on average slightly faster by less than 2 seconds with 120.4s (sd = 33.6707) opposed to 116.8s (sd = 31.1152) of inexperienced users (not significant).

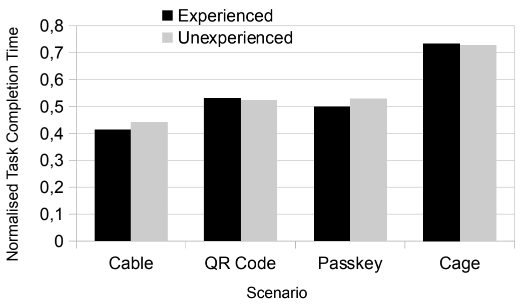
[21]The number in brackets represent the mean values of the preference ranking.



(a) On SUS Score



(b) On Task Completion Time



(c) On Normalised Task Completion Time

Fig. 8: Experience Effects

Effect on Normalised Task Completion Time Within the set of experienced users, significant advantages using pairwise t-tests were found for the cage scenario compared to the cable ($p < 0.01$, $power = 91.36\%$), compared to the QR code scenario ($p < 0.05$, $power = 90.43\%$), and compared to the passkey scenario ($p < 0.01$, $power = 82.44\%$). The QR code was also completed relatively faster than the cable scenario ($p < 0.05$, $power = 60.66\%$) and there was a borderline advantage of the passkey compared to the cable scenario ($p = 0.07688$, $power = 40.37\%$.
$QR >^{(*)} Cable$, $Pass >^{(*)} Cable$
$Cage >^{**} Pass$, $Cage >^{**} QR$, $Cage >^{**} Cable$

Unexperienced users completed the cage scenario relatively faster than the cable ($p < 0.05$, $power = 60.77\%$), the passkey ($p < 0.05$, $power = 45.37\%$), and the QR code ($p < 0.05$, $power = 55.9\%$).

Table 6 shows the results of pairwise t-tests between experienced and inexperienced users. Besides the bias towards the cage scenario, inexperienced users completed the QR code scenario more efficiently than experienced users completed the cable scenario.

		Experienced			
		Cable	QR Code	Passkey	Cage
In-Experienced	Cable	-	-	-	< 0.01 (92.03%) ↓
	QR Code	< 0.05 (44.12%) ↑	-	-	< 0.01 (74.84%) ↓
	Passkey	-	-	-	< 0.05 (82.19%) ↓
	Cage	< 0.01 (96.8%) ↑	< 0.05 (88.25%) ↑	< 0.05 (95.92%) ↑	-

Table 6: Comparison Normalised Task Completion Times by Experience
Arrow up = unexperienced performed better, arrow down = experienced performed better.

4.3. Qualitative Results

Observing the participants, it became apparent that users rarely read instructions presented on the Pico or in the application. This calls for implementing a pairing mechanism that is very intuitive. After a successful first pairing, the consecutive pairings were rushed by participants. Warnings about possible compromises of the process are likely to be ignored. In four cases, this behaviour led to mistakes in the pairing process.

For some Participants (4)[22] the connection between Pico and Picosiblings was only clear in the physical connection scenario. They did not draw the connection between a separate QR code (2) or passkey (2) and the Picosibling even though the description/name of the Picosibling was clearly written on top of it. This suggests that if issuing QR codes or passkeys for Picosiblings, they need to link to the host of the Picosibling. One participant even scanned the same QR-code twice for different pairings.

Switching between the Pico and the application further presented a challenge for some participants (5). While many participants expressed their content with using a management application "like iTunes", for some, using more than one device at a time was challenging and cumbersome. Two participants expressed their satisfaction in having a pre-defined description for each Picosiblings that they can confirm. However, many did not check the description in successive pairings. This behaviour could become problematic for identifying Picosiblings uniquely. Taking a picture of the Picosibling as discussed in section 5 could overcome this problem.

Many participants clicked the "Pair" button on the Pico more often than necessary because they did not know whether the Pico is already in pairing mode or not. This issue of clear feedback could be approached by signalling the user that the Pico is in pairing mode via the display.

Despite emphasising the connection between Pico and Picosibling while pairing, the physical connection scenario was often seen as "unnecessary" (6), a "constraint" and one participant even stated that "nobody would like to plug anything into their things". Connecting two devices

[22]The number in brackets indicates the number of participants that were found to show this behaviour.

by cable was also thought to be “easier for older people” and “increases confidence in security [sic]” (4).

A quarter of the participants (6) criticised that they had to take a picture in order to scan a QR code. They are used to applications that scan the QR code automatically when holding the smartphone above the QR code. Another issue that was verbalised was the lack of clarity surrounding whether or not the picture taken by the participant covered sufficient space of the QR code (3). An automatic scanning option would help to overcome this issue. Participants also stated that QR codes are ubiquitous and easy to use because of their experience with similar technologies (5). This complies with the principle of consistency in HCI. One person stated that QR codes “increase confidence in security” while another found that a “QR code appears to be easier to be hacked” than a passkey. The latter was due to the fact that QR codes were often used “for fun” while passkeys have an “image [...] for security”.

During the Faraday scenario participants were often confused about why they would have to use the Faraday cage and did not understand its purpose (9). They would prefer to “just have devices in close proximity” (1). Six participants forgot to put the Pico into the cage despite the instructions and others started pairing with only one Picosibling inside the Faraday cage (5). Nonetheless, for many participants the cage was an effortless way of pairing more than one Picosibling at a time. One explicitly stated that the time the pairing takes in the Faraday cage “does not matter” since they “can do other things” meanwhile. Contrary to this, four participants were discontent with the duration of the pairing and would have liked to have an indicator of the time the pairing takes. It was also stated that an output to signal the beginning of pairing would help them understand the process and confirm that the pairing process started (6), and two were confused about what to do after pressing “Pair”. There was also confusion about how to acquire a Faraday cage for pairing and how much such a cage would cost (3).

Concerning the passkey scenario, five participants were confused about why they had to use a passkey for a technology that aims at replacing passwords. Nevertheless, some participants stated that using passkeys was intuitive and easy as they are used to typing passwords (6). Two participants would have liked to enter the passkey on the Pico rather then the computer. After correctly entering the passkey there was no confirmation that the passkey was correct. However, the newly added Picosibling appeared in the list of Picosiblings. One participant stated that an explicit confirmation after entering the passkey would clarify the process and two participants did not understand the role of the Pico in the process since it was only used for clicking “Pair” initially. A few participants also thought they were choosing passwords for each Picosibling and three feared that “remembering many passwords will be a problem”. Two participants started typing the name of the Picosibling instead of the passkey. However, two also stated that passwords increase their confidence in security.

Several participants in the user study reported that voice-output as guidance through the pairing steps would be beneficial. Audio-output would possibly increase the usability of the Pico and offer new pairing options when pairing Picosiblings that have an audio-receiver.

4.4. Discussion of the Study

While limited time and resources prevented an in depth analysis with a larger age range, the results are a good indicator of potential pairing scenarios for a target group of young adults. Future studies should adjust the system for users of all ages to aid wide deployment of the Pico.

One result of this study is that users have different preferences of pairing mechanisms. While some despise using cables for pairing, for many users the physical connection scenario was intuitive and familiar. Statistically, the QR code scenario can be seen as the mechanism that will most likely attract the highest number of users. It achieved the best results for SUS scores among all tested groups and had the second best normalised task completion times. Users tend to be experienced using QR codes. The Pico's authentication system could further strengthen their confidence in them. Pairing via the Faraday cage showed the best normalised times. However, this result is biased by the measurement issues previously discussed.

I propose using a pairing framework similar to Bluetooth's simple pairing [12] or the USB wireless standard to offer different pairing mechanisms for different purposes and preferences. Users would choose their preferred system and in future stages of the Pico's deployment, when Picosiblings become inaccessible, the pairing mechanisms can be adjusted or expanded. It seems sensible to first implement the QR code option and the physical connection. On the one hand, this allows for flexibility in later stages and a usable system that many users are familiar with (QR code). On the other hand, users can choose to use a system that they are confident in to provide security and that makes the process of pairing explicit (physical connection).

5. Picosibling Management

The management process of Picosiblings must support at least the following use cases:

1. Pairing a set[23] of Picosiblings to a blank Pico.
2. Adding Picosiblings to an existing set without changing the threshold number.
3. Removing a Picosibling from a set. This should be possible even if a user is no longer in possession of the Picosibling.
4. Recovering from the loss of $n - k + 1$ Picosiblings.
5. Support the management of special shares in later stages.

Alternatives to having an auxiliary technology to support these use cases would require the presence of all Picosiblings, including those stored in safety deposit boxes - a usability nightmare.

Alternatively, the Pico could be used as a proxy to identify the missing Picosibling and remove it. While the token's I/O capabilities are limited, an Android application as envisioned by Stajano et al. [80] offers sufficient capabilities to manage the Picosiblings.

Picosibling Identities Instead of using descriptions, Picosiblings could also be identified by images of their hosts. These images can either be manufacturer-provided and sent to the Pico or be taken by the users with the Pico. This could eliminate the need for an auxiliary application altogether as the Pico could serve as the (limited) I/O device.

5.1. Attacks on Picosiblings

Tompa and Woll Tompa and Woll [86] discussed an attack in which cheating participants deceive others by sending false shares. The aim of this is to gain access to the deceived participants' shares and to reconstruct the master key. A variant of this attack is a danger for the Pico in that attackers gaining control of Picosiblings could send false shares. Such false shares lead to a weak denial of service (DoS) attack on the Pico as the Pico would have to determine which shares are incorrect by creating all potential polynomials. The cheating participant could even remain undetected if only k Picosiblings sent their shares. If users had no knowledge of the corruption of the Picosibling, they would not be able to access their credentials.

A possible counter to this attack would be to store a salted hash $h(y_i, salt)$ of the Picosibling's share part y_i in the Pico for integrity checks and to notify the user of corrupted shares. More computationally expensive counters presented by Tompa and Woll [86] or Martin [50] are unnecessary since the Pico serves as a trusted combiner in our case.

[23]Hereafter, "set" refers to the set of Picosiblings.

Jamming, sleep deprivation torture attacks, and others The jamming of signals in public places next to ATMs or points of sale (POS) would deny user-transactions. Fallback solutions using the special shares could overcome this DoS.

Similarly, sleep deprivation torture attacks as described by Stajano and Anderson [78] could drain the resources of Picosiblings sufficiently quickly to disable them. Users need to be notified in cases where Picosiblings run out of energy so that they can initiate recharging. Picosiblings could broadcast a final message encrypted under the key used for communicating with the Pico. One solution to this is the use of a relay system through other Picosiblings. However, specifics are yet to be developed as there are severe privacy risks involved.

Users need a way to inspect their Picosiblings. Attackers could steal Picosiblings, disable them, and return them to users who are not aware of their altered state. An option to display the state of all Picosiblings currently in the proximity of the Pico could hint at compromises, power failures, and other problems that threaten the operability of the Picosiblings.

Malicious pairing Attackers might pair Picosiblings after resetting them or before users had the chance to pair them to their own Pico. Until they are inaccessible, users can reset them through physical contact and re-pair them to their Pico. However, there has to be a way of resetting inaccessible Picosiblings, because such an attack would be an effective DoS.

5.2. Pairing

Potential pairing protocols were discussed in section 3.4. This chapter elaborates on management considerations regarding pairing.

Initial pairing The threshold number must be fixed before initial pairing. Otherwise, Picosiblings would have to stay in proximity until all Picosiblings have been paired to the Pico to receive their shares. If the threshold is fixed, however, the Pico must handle the eventuality that a user leaves prematurely before pairing a sufficient number of Picosiblings. Keeping the Pico in pairing mode for an extended period is a security risk, as adversaries could pair their own Picosiblings or extract the master key.

Section 5.5 discusses the option to recover from losing all Picosiblings via the remote server. It should be set up prior to initial pairing. After a long idle period in the initial pairing mode, the Pico should lock or wipe the credentials that have been generated. The remote share can than be used to unlock the Pico and continue the pairing. Potential auxiliary devices, the backup server, and the biometric share should also be set up prior to the initial pairing. An auxiliary application on a device with internet connectivity could be paired via USB and provide the means to set up the special shares after verifying its authenticity.

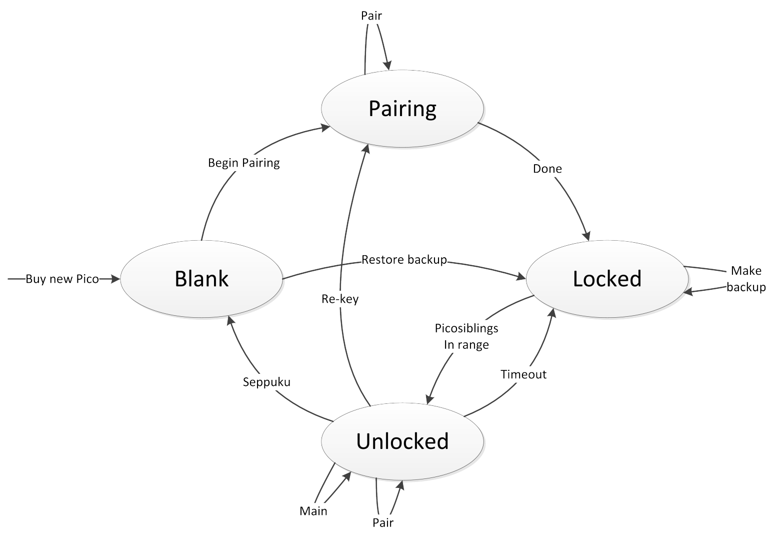


Fig. 9: Pico State Diagram

Further pairings Prior to successive pairings after the initial setup, the Pico should request all current shares to confirm the presence of a sufficient number of Picosiblings. This would complicate the re-pairing attacks identified as a security risk by Stannard and Stajano [82].

Restoring backups When losing $n - k + 1$ Picosiblings, users need a way to recover their credentials. One option would be to use the backup to restore the Pico (see 5.4) and allow the user to be in the initial pairing mode upon proving access to the remote share, biometric share, and key for backup recovery. The Pico's state diagram[24] (fig. 9) is altered slightly in that restoring the backup and proving possession of all the previously mentioned credentials leads to the "pairing" or "unlocked" state instead of the "locked" state. Restoring backups when still in possession of Picosiblings does not require the remote share. In that case the restored Pico will switch from the "blank" state to the "locked" state as shown in the state diagram.

5.2.1. Resharing and Changing the Threshold Boundaries

Resharing schemes usually presuppose that there is no trusted party to serve as a dealer and combiner [36], thus all participants must collaborate to generate the new shares. The Pico can take on this role.

Proactive resharing improves security, forcing the attacker to compromise all k Picosiblings during a single resharing interval. Picosiblings that have been compromised in a previous interval cannot be exploited to recreate the master key in combination with Picosiblings of the current interval. Their shares belong to different polynomials.

[24]According to Stannard [81] p. 4.

In Shamir's secret sharing scheme resharing is accomplished by choosing a new polynomial $n(x)$ of degree $k-1$ that is fixed at position 0 by $n(0) = 0$ and is added to the existing polynomial. This allows the master secret to remain constant since $f^{(t)}(0) = f^{(t-1)}(0)+n(0) = S+0 = S$ in which $f^{(t)}$ represents the polynomial to recreate the master key at interval t. All participants update their shares by adding the value of the new polynomial at their share's position yielding $f^{(t)}(x) = f^{(t-1)}(x) + n(x)(mod\ q)$[25].

The Pico chooses this new arbitrary polynomial $n(x)$, calculates the values $n(x_i)$ at the positions x_i stored for each Picosibling, and propagates these values to the Picosiblings. Since Picosiblings might not be in range of the Pico during resharing, the Pico stores these offsets and issues an update operation upon being in proximity. Offsets for Picosiblings that miss several resharing rounds are added. This simplifies the resharing process for the user, who needs not to gather all Picosiblings for each resharing transaction. It also does not reveal any information to attackers who acquire possession of a Pico. While the offsets are stored outside the secure storage of the Pico, they reveal no information about the secret.[26] The offset can be included in the ping-protocol by adding another field:

$$P \rightarrow PS_i\text{: } \{offset,\ N_j,\ c_j\}_{K_{P,PS_i,1}}$$

Aside from the nonce's function for the ping-protocol, it also prevents replay attacks that could invalidate the Picosibling's share if the Picosibling updates its share by an old offset. If the offset is not available, the Picosibling continues with the ping-protocol. Otherwise, it updates its share first. When the Picosibling sends the pong-message, containing the share, the Pico can check the integrity of the newly generated share by exploiting the in section 5.1 proposed countermeasure against cheating participants. The presented hash of the share that is concatenated with the salt can be XORed with the offset for the share and the resulting value stored in the Pico. When the Pico receives the updated share, it can calculate the old share and create the integrity hash. This hash is compared against the stored hash and if they match, the hash of the share and salt is updated.

A crucial question is how often resharing should occur. A counter of share retrieval requests, as suggested by Peeters [59], does not seem practical for the Pico, as several Picosiblings are requested far more frequently because of their hosts' value to the user. Another option is to have the resharing take place periodically and to further initiate resharing each time the set of Picosiblings is altered. Ideally, the Pico adjusts the timings according to its environment. Sensors and machine learning techniques could ensure that resharing occurs more frequently in hazardous environments, for example when users visit areas of high crime rates.

[25] q denotes a prime chosen initially to set up the secret sharing scheme to define the finite field $\mathbb{Z}_q$ that serves as the computing base for all mathematical operations regarding secret sharing.

[26] An adversary who gets access to a Pico that stores k offset values for Picosiblings yields 0 for $n(0)$ instead of the secret $S = f^{(t)}(0)$.

Resharing can also involve changing the master key, e.g. because the user indicated a possible compromise. When changing the master key, each Picosibling has to update its share-data. A possible solution for Picosiblings that are not present is to save the new share for the Picosibling in the Pico encrypted under the ephemeral key between Pico and Picosibling. After encrypting the share, the Pico would hash the ephemeral key and store a counter for the number of hash-rounds. This ensures that an adversary seizing possession of the Pico cannot extract the share, since the key to decrypt the share is no longer in the Pico. The Picosibling can receive the encrypted share, decrypt it, and hash its ephemeral key according to the Pico's instructions. The security of this variant would than depend on the user correctly identifying Picosibling thefts. Otherwise, an attacker stealing a Picosibling could return to within proximity of the Pico to receive the share update.

This resharing process relies on tamper-evidentness[27] of the Picosiblings. An attacker who can extract key information, an old share y_i, place the Picosibling back in the victim's possession, and eavesdrop upon the communication, can update the retrieved share at each resharing round.

An issue of resharing usually occurs when k Picosiblings miss several resharing rounds (or all of them) and the attacker gains access to these k Picosiblings. When the master key has not been changed, an attacker having access to the x_i values of these Picosiblings' shares can reconstruct the master key. However, the x_i values are only stored in the Pico and the encrypted backup system.[28] An attacker who has access to either (and the k Picosiblings), essentially breaks the system regardless of resharing.

An attacker who steals Picosiblings over time, returns them, and eavesdrops upon the following communication between Pico and Picosiblings would be able to update the share-information. Further research has to be conducted on how to prevent such an attack. When users notice this compromise, they can perform a re-pairing of the Picosibling. However, identifying this attack correctly is difficult.

5.3. Adding and Removing Picosiblings

Adding a Picosibling requires the Pico to be unlocked to counter primitive re-pairing attacks. The biometric special share could be required each time a Picosibling is to be added, thereby making re-pairing attacks infeasible altogether. Otherwise, the Pico should remove all received shares (and the corresponding master key) and try to reacquire shares of a sufficient number of Picosiblings upon the user pressing "Pair". Resharing should always take place after the user adds or removes Picosiblings to add further time diffusion.

[27]When Picosiblings are inaccessibly incorporated with their hosts, their hosts implement a weak form of tamper-evidentness in that users notice when the hosts are damaged.

[28]See section 5.4.

A Picosibling can be removed by wiping all credentials for it from the Pico's memory. Any attempts by the Pico to use the Picosibling's share afterwards will fail because of the inability of the Pico to decrypt the Picosibling's messages and – more importantly – its inability to recreate the share s_i.[29] Attackers who seize possession of k Picosiblings and the Pico cannot use the shares of previously-removed Picosiblings, as they cannot determine the x_i-values for their shares to recreate the master-key. Removing lost Picosiblings is only possible when the user can uniquely identify them through an auxiliary technology or the Pico.

5.4. Backups

Previous Development The Pico team works on a backup-system that backs up the Pico's state each time a pairing to a web service occurs. When setting up the Pico, users must write down a recovery code representing the master key to access the secure core of the backup. If all Picosiblings were lost, users can use this recovery code to recover their credentials from the secure core of the Pico encrypted under the master key.

My Additions If the password was only used to decrypt the secure core and not other credentials of the Pico, these credentials would be vulnerable to an attacker who acquires the backup. Furthermore, resharing with a changed master key, as presented in 5.2.1, would be very cumbersome for users who would have to write down a new recovery code for each resharing. Another issue is that an attacker does not need possession of the Pico to gain access to the user's credentials. With only the backup and k Picosiblings, one can create a functional Pico to impersonate the user.

Each time a Picosibling is added or removed, the backup of the outer core should be updated. Therefore, the outer core of the Pico is encrypted under a key K_{backup}. The secure core remains encrypted under the master key *XOR'd* with the biometric share. A useful side-effect of this approach is the ability to restore the set of Picosiblings on another Pico without having to restore the credentials. Thus, different Pico-instances for different purposes accessed under the same Picosiblings can be realised. After "cloning" the Picosiblings' credentials, the new and original Pico could stay synchronised through the remote server. A solution for this synchronisation is left for further research.

This scheme allows for the recovery of credentials even if all Picosiblings are lost. The remote server provides a sufficient number of shares that can recreate the master key when combined with the x_i-values stored in the outer core encrypted under K_{backup}, and users are unlikely to lose their biometrics. Attackers who only access the remote servers and forge the biometric,

[29] Even if the Pico acquire the y_i value of the share, the missing x_i value prevents utilising the share-part for recreating the polynomial.

Version	**Secure Core**	**Outer Core (R)**
...	...	...
v12	$\{Sec.\ core\ incl.\ K_{backup}\ at\ t = 12\}_{K_{Master\ XOR\ biometric}}$	$\{Outer\ core(R)\}_{K_{backup}}$
...	...	...
v14	$\{Sec.\ core\ incl.\ K_{backup}\ at\ t = 14\}_{K_{Master\ XOR\ biometric}}$	$\{Outer\ core(R)\}_{K_{backup}}$
...	...	...

Outer Core
$\{Outer\ core\ (Complete)\}_{K_{backup}}$

Table 7: Structure of the Backup Repository

however, cannot access the secure core of the backup since they also need knowledge of the key K_{backup} to access the x_i-values of the shares.

K_{backup} is further stored inside the secure core of the Pico to allow the unlocked Pico to encrypt the outer core each time a removal or addition of a Picosibling occurs. Otherwise, the user would have to enter K_{backup} each time a backup is performed.

While the secure core of the Pico's backup is stored versioned, the backup's outer core is replaced each time a change in the set of Picosiblings takes place. This approach fails if the master key is changed and the user desires access to an earlier backup. Therefore, the part of the outer core consisting of the x_i values of the remote server is also versioned. The remote server provides a sufficient number of shares to recreate the master key at any point of time and can version its own share-parts y_i (see table 7 for a visualisation of the backup repository's structure). Nevertheless, the remote server cannot recreate master keys since this also requires the x_i values stored encrypted under K_{backup} or in the Pico.

The user is required to reacquire the biometric share each time a backup is created, unless the biometric share is stored inside the secure core. Since an internet connection must be available for backups, the remote shares can be updated. While reacquiring the biometric share might decrease the usability of the system, such a solution would also counter re-pairing attacks as discussed in section 5.3. After each change in the set of Picosiblings, resharing takes place to invalidate all former shares and the backup of the outer core is updated. This adds further time-diffusion, as an attacker having access to the backup at time t and the password K_{backup} to decrypt the outer core must also acquire k Picosiblings of the current resharing interval t.

5.5. Special Shares

The need for special shares is axiomatic. Many use cases cannot be secured without a remote or biometric share. This chapter presents use cases and possible solutions using special shares.

Value of the shares Stannard [81] proposed always requiring special shares to unlock the Pico. The master key would be created by *XORing* both special shares and the reconstructed secret from the Picosiblings. While using the biometric share this way is sensible as the user is unlikely to lose the biometrics, demanding the remote share for unlocking has usability issues. Assuming the Pico is used for offline authentication as well as for online authentication, it cannot be guaranteed that the Pico can access the remote server sufficiently frequently to refresh the share for all offline transactions.

The remote share is a means for revocation when the Pico is lost or stolen. When the remote server stops sending its share periodically, the Pico will lock. It can also simplify recovery from losing all Picosiblings. Recovery would require the remote server to store shares similar to those of the Picosiblings. It could store k shares that are only sent when users execute a multi-level security protocol.

Furthermore, for online authentication, the Pico could *XOR* the credentials for the web service (retrieved from the Pico's secure core) with a remote value. These remotely stored values can be stored in a distributed manner to counter DoS attacks and can be dynamically combined with the stored credentials when accessing web services.

Using remote shares in this manner enables users to prevent any access to their online accounts upon executing a multi-level security protocol to disable the remote shares. This would be a way to recover even from losing the Pico, all Picosiblings, and the attacker having a forgery of the biometric scan.

Locking Occasionally, users will need a way to lock their Pico to be inaccessible even if presented with shares of k Picosiblings. At swimming pools, gyms, or other venues wherein users leave their belongings in a locker, an attacker could easily seize possession of the Pico and k Picosiblings. A simple locking mechanism would involve the user simultaneously pressing the "Pair" and "Main" button. Once locked, the Pico can only be unlocked using the biometric share.

Pairing Users could set up the remote share and biometrics upon performing the initial pairing. Biometrics and the remotely stored values presented earlier in this section can be validated using the salted hash variant discussed in section 5.1.

Remote shares are used like other Picosiblings, having the Pico storing corresponding x_i values. While the remotely stored values are set up when creating new user accounts, the remote shares are set up at the initial pairing.

Requesting the shares The special shares should have a longer decay timer. For security-critical operations like pairing, removing, and even for accessing valuable accounts, the biometric share could always be required. Remote shares are requested whenever the Pico is connected to the internet, and the remotely stored values are requested at each authentication request for the corresponding web service. The versioned remote shares can only be accessed by performing a multi-level security protocol that allows recovery from losing all Picosiblings.

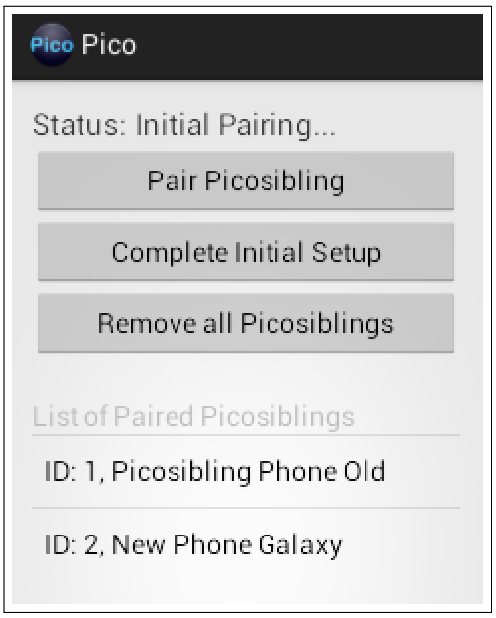
6. Prototype Implementation

The implementation goal was to build a running prototype of the Pico-Picosibling subsystem, including the creation of shares for the master key, adding and removing Picosiblings, pairing, and the communication between Pico and Picosiblings. Such a prototype allows for further usability studies that can explore the pairing and handling of Picosiblings, and can be re-used for the final Pico Android application.

The implementation followed an iterative incremental approach that sought to produce a running prototype at the end of each iteration. This chapter briefly outlines the implementation. *Javadoc* files are attached to the implementation, and appendix A.5 describes software engineering aspects of the implementation in more detail. Fig. 10 shows the UI of the Pico prototype.

6.1. Components

Storage An SQLite database served as the outer core storage of the Pico. The two tables for the Pico's general properties and the Picosibling credentials were accessed via two database management classes that ensured the integrity of the database. Conversely, the Picosibling's storage was modeled using Android's *SharedPreferences*, because the amount of data stored by the Picosibling did not justify using a database and *SharedPreferences* offer resource-conserving data access.



Status: Initial Pairing...

ID: 1, Picosibling Phone Old

ID: 2, New Phone Galaxy

Fig. 10: Pico Prototype UI

Secret Sharing Instead of using an open source implementation of Shamir's secret sharing, his algorithm was developed from scratch for two reasons: First, the quality of publicly-accessible secret sharing implementations cannot be easily verified and, second, flexibility of the implementation was needed to allow for extensibility (especially for resharing). The final implementation includes all necessary functions for the Pico. Master keys of varying lengths can be generated and split into a desired number of randomly generated shares over the finite field that is defined by a prime number greater than the master key. New shares are generated when presenting a threshold number of existing shares, and the master key is recreated when a sufficient number of shares are available. Several implementations of this scheme use the natural number array $1, 2, \ldots, n$ as the x-coordinates of the shares. This is insufficient when splitting shares into their coordinates, and only propagating the y-coordinate to the Picosibling. The range of possible x-coordinates would be limited to 1 *to n* for n Picosiblings, allowing attackers to run brute-force attacks on the x-coordinate once they acquired a sufficient number of Picosiblings. The developed implementations uses the complete range of the finite field for its x-coordinates.

Cryptographic operations were centralised through a *CryptoManager* class used for encryption, decryption, and creating hash-values.

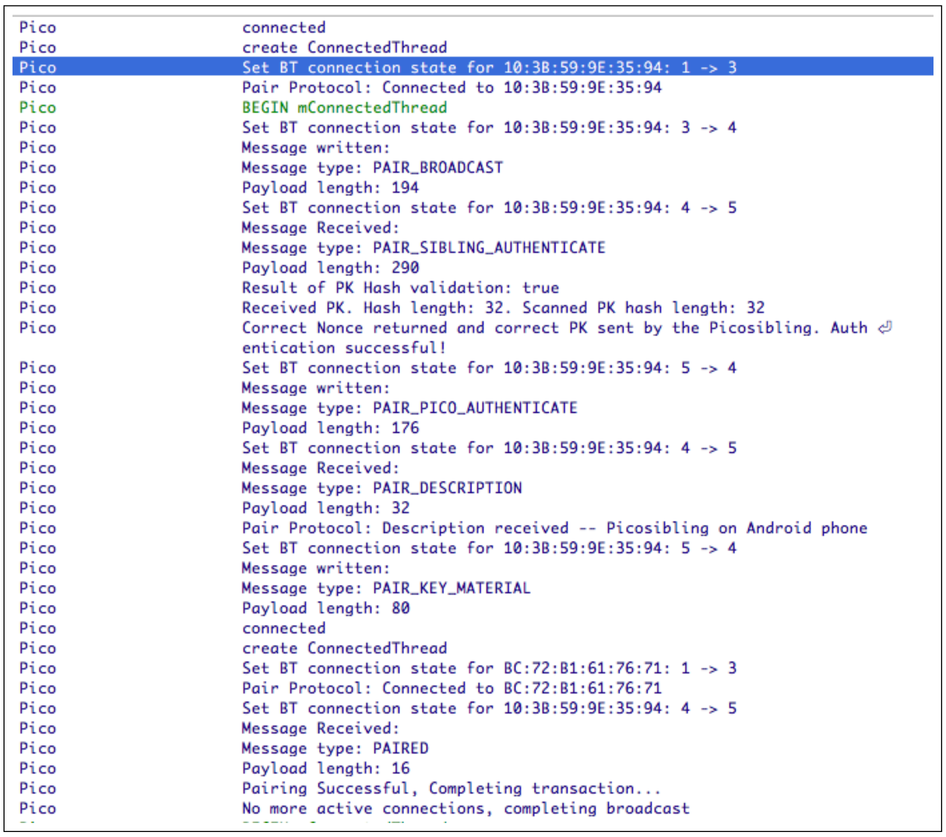
Bluetooth Message exchange is accomplished through sending *PicoMessage* objects via Bluetooth. These encrypted objects contain the information for the protocol's steps and further parsing information. Each connection to a Bluetooth device is modelled by a *BluetoothConnection* object that is responsible for sending data, receiving data, and managing the sockets. *BluetoothConnectionManager* classes for each protocol manage all *BluetoothConnection* objects for a protocol execution. *PicoMessage* objects for data transfer are created using *Builder* and *Helper* classes for parsing. This ensure correct creation of these objects and extensibility.

Third Party Software To create, scan, and parse the QR codes required for pairing, the *ZXing*[30] application was integrated via *Intents*. Two classes of the *ZXing* project – *IntentIntegrator* and *IntentResult* – were used in combination with an intent-call. *ZXing* sources were further used by the Picosibling to display the QR code. The QR code modelled a *Base64*-encoded string consisting of a one-byte identifier, a 32-byte hashcode of the Picosibling's public key, and the Picosibling's secret (32-byte).

6.2. Protocols

The ping-protocol is implemented as described by Stannard [81]. Instead of running it permanently, it is only executed when the user clicks “Pair”. If the ping-protocol acquires a sufficient number of shares from Picosiblings in proximity, a management interface is accessible to pair new Picosiblings, remove Picosiblings by description, and to reset the Pico. The ping-protocol requirement only applies to the scenario that an initial pairing had been performed previously. Otherwise, the user can complete the initial pairing without requiring the proximity of additional Picosiblings.

[30]https://github.com/zxing/zxing



```
Pico              connected
Pico              create ConnectedThread
Pico              Set BT connection state for 10:3B:59:9E:35:94: 1 -> 3
Pico              Pair Protocol: Connected to 10:3B:59:9E:35:94
Pico              BEGIN mConnectedThread
Pico              Set BT connection state for 10:3B:59:9E:35:94: 3 -> 4
Pico              Message written:
Pico              Message type: PAIR_BROADCAST
Pico              Payload length: 194
Pico              Set BT connection state for 10:3B:59:9E:35:94: 4 -> 5
Pico              Message Received:
Pico              Message type: PAIR_SIBLING_AUTHENTICATE
Pico              Payload length: 290
Pico              Result of PK Hash validation: true
Pico              Received PK. Hash length: 32. Scanned PK hash length: 32
Pico              Correct Nonce returned and correct PK sent by the Picosibling. Auth ⏎
                  entication successful!
Pico              Set BT connection state for 10:3B:59:9E:35:94: 5 -> 4
Pico              Message written:
Pico              Message type: PAIR_PICO_AUTHENTICATE
Pico              Payload length: 176
Pico              Set BT connection state for 10:3B:59:9E:35:94: 4 -> 5
Pico              Message Received:
Pico              Message type: PAIR_DESCRIPTION
Pico              Payload length: 32
Pico              Pair Protocol: Description received -- Picosibling on Android phone
Pico              Set BT connection state for 10:3B:59:9E:35:94: 5 -> 4
Pico              Message written:
Pico              Message type: PAIR_KEY_MATERIAL
Pico              Payload length: 80
Pico              connected
Pico              create ConnectedThread
Pico              Set BT connection state for BC:72:B1:61:76:71: 1 -> 3
Pico              Pair Protocol: Connected to BC:72:B1:61:76:71
Pico              Set BT connection state for 10:3B:59:9E:35:94: 4 -> 5
Pico              Message Received:
Pico              Message type: PAIRED
Pico              Payload length: 16
Pico              Pairing Successful, Completing transaction...
Pico              No more active connections, completing broadcast
```

Fig. 11: LogCat Output of the Pico while Executing the Pairing Protocol

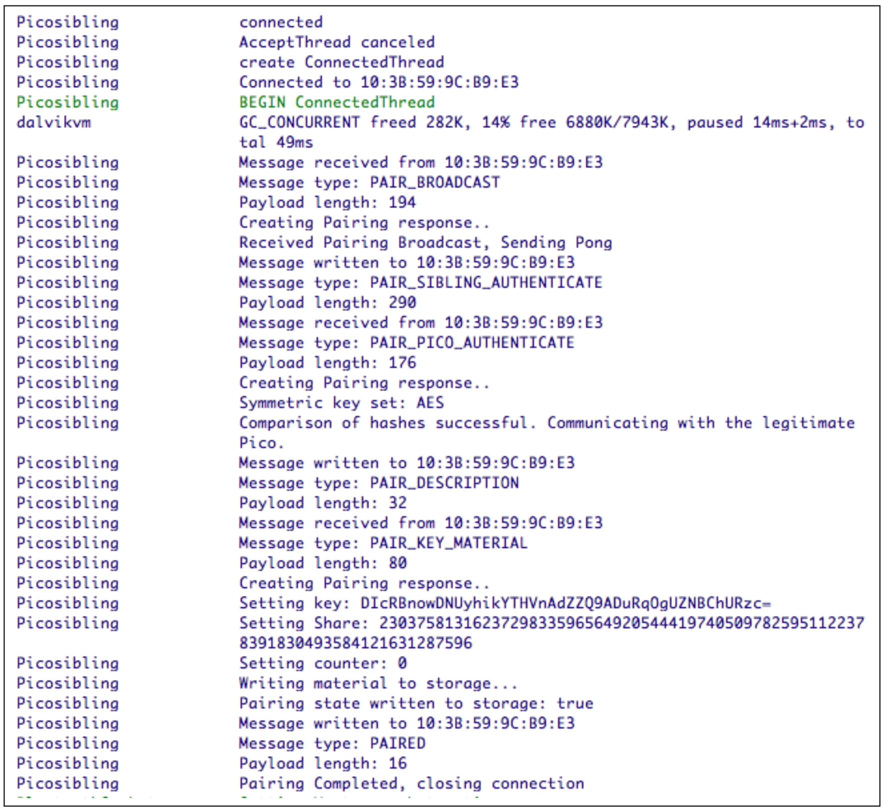
Following the pairing protocol's specification of section 3.4.2, the Pico's pairing broadcast is issued after the user scans the QR code. The messages for key-establishment are encrypted under RSA, while the final messages use AES. After the successful execution, the user is prompted to enter a distinctive description that is stored with the Picosibling's credentials in the outer core of the Pico. A list of paired Picosiblings indicates to the user whether the protocol succeeded.

Both protocols are executed as asynchronous tasks in Android to separate them from the user interface.

At the time of this writing, the resharing protocol is not completely integrated. All necessary methods are implemented and have been tested individually but final tests have not been performed yet.

6.3. Testing

Besides unit tests using *JUnit*, integration tests of the main components, and extensive *LogCat*-logging, I tested my prototypes using three Samsung GT-S5310 phones. Logs of the test runs can be found in the *zip*-file containing my implementation next to the LogCat output that documents protocol runs on both devices. I created a test project for the Pico application and another for the Picosibling containing integration and unit tests.



```
Picosibling              connected
Picosibling              AcceptThread canceled
Picosibling              create ConnectedThread
Picosibling              Connected to 10:3B:59:9C:B9:E3
Picosibling              BEGIN ConnectedThread
dalvikvm                 GC_CONCURRENT freed 282K, 14% free 6880K/7943K, paused 14ms+2ms, to
                         tal 49ms
Picosibling              Message received from 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIR_BROADCAST
Picosibling              Payload length: 194
Picosibling              Creating Pairing response..
Picosibling              Received Pairing Broadcast, Sending Pong
Picosibling              Message written to 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIR_SIBLING_AUTHENTICATE
Picosibling              Payload length: 290
Picosibling              Message received from 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIR_PICO_AUTHENTICATE
Picosibling              Payload length: 176
Picosibling              Creating Pairing response..
Picosibling              Symmetric key set: AES
Picosibling              Comparison of hashes successful. Communicating with the legitimate
                         Pico.
Picosibling              Message written to 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIR_DESCRIPTION
Picosibling              Payload length: 32
Picosibling              Message received from 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIR_KEY_MATERIAL
Picosibling              Payload length: 80
Picosibling              Creating Pairing response..
Picosibling              Setting key: DIcRBnowDNUyhikYTHVnAdZZQ9ADuRqOgUZNBChURzc=
Picosibling              Setting Share: 2303758131623729833596564920544419740509782595112237
                         8391830493584121631287596
Picosibling              Setting counter: 0
Picosibling              Writing material to storage...
Picosibling              Pairing state written to storage: true
Picosibling              Message written to 10:3B:59:9C:B9:E3
Picosibling              Message type: PAIRED
Picosibling              Payload length: 16
Picosibling              Pairing Completed, closing connection
```

Fig. 12: LogCat Output of the Picosibling while Executing the Pairing Protocol

An exemplary log of the pairing-protocol is shown in fig. 11 for the Pico and in fig. 12 for the Picosibling.

6.4. Discussion

The methods *listenUsingInsecureRfcommWithServiceRecord* and *createinsecurerfcommsocket-toservicerecord* for Bluetooth communication are supposed to allow message transfer of previously (Bluetooth-) unpaired devices. However, testing revealed that these methods behave non-deterministically in that the user is occasionally prompted to perform Bluetooth-pairing between the devices to continue. This is a known problem (I used Samsung GT-S5310 and Samsung GT-S6500 smartphones for testing, and they both showed this behaviour) and cannot be easily overcome. Bluetooth-pairing is only possible with user interaction. Furthermore, the *insecureRfcomm* methods require discoverability of the device setting up the server socket. Discoverability also requires user interaction in Android.

To provide a realistic user experience for user studies, I switched to support communication only between Bluetooth-pre-paired devices. When using the applications in user studies, the devices have to be Bluetooth-paired before performing the Pico-pairing.

6.5. Future Work

The final prototype can serve as a user experience prototype. Building on my work, a prototype embedded into “real” Picosiblings would be the next step to allow for a more realistic user experience. Since a remote server was not available, I excluded scenarios relying on such, for example users leaving initial pairing prematurely. Removing Picosiblings is forbidden when this would leave less than a threshold number of Picosiblings paired. Implementing these functions will be the next step for further support user studies.

7. Conclusions

Preliminary Work To undertake this project, I needed to acquire knowledge in several areas. I knew little of the Android environment. I had not previously designed a user study, and knew nothing about HCI processes or the work associated with them. My statistical knowledge was at a very basic level and I had never designed a cryptographic protocol before.

Objectives met I fulfilled or surpassed all the tasks set for the project. Defining requirements for Picosiblings and completing the extensive literature survey revealed that several pairing mechanisms are currently suitable for pairing Picosiblings to the Pico and more might become feasible in the future. To prove the applicability of the found pairing mechanisms for the Pico, I developed variants of the protocols for the Pico. Evaluating *SiB* (QR code pairing), *MiB* (Faraday cage pairing), a *PAKE* variant (passwords), and physical contact as means to bootstrap the security relationship between Pico and Picosiblings in a user study yielded promising results. Physical connections are seen as secure and facilitate the understanding of the association process while QR codes are a familiar concept to users. Users favour scanning QR codes over the other alternatives

I further introduced the idea of resharing through the novel approach of splitting the secret share into its x and y-component and only propagating the y-component onto the Picosiblings. This allows for resharing without changing the master key in that only the y-coordinate is updated and the Pico can even store the offset without endangering the secrecy of the master key in the eventuality that the Pico is compromised.

Protocols for adding and removing Picosiblings were developed as well as an idea of resharing by changing the master key. The original design of the Pico and Picosiblings was adjusted at several points, use cases and options for the secret shares introduced, and an idea for the backup system proposed to integrate my ideas and create a system that maximises the potential synergies.

The final prototype for the Picosibling subsystem of the Pico project was developed to allow for further exploration of usability aspects of my suggested pairing mechanisms and protocols. It offers functionality for pairing, adding and removing Picosiblings, and the general communication interface between Pico and Picosiblings. Such a prototype is valuable for identifying possible shortcomings, misunderstandings, and gaps in the mental model of users. Test cases proofed that the protocols are feasible. A comprehensive analysis including mathematical proofs, penetration testing, and usability studies remains to be performed. However, these contributions combined with my considerations regarding the backup system and special shares form the foundation of an integrated, secure, and usable Picosibling design.

Table 8 summarises the completed deliverables of this project.

Deliverables
Requirements Analysis of the Pico's Pairing Process
Literature Survey of Pairing Mechanisms
Development of Suitable Pairing Protocols
Quantitative Results of User Study
Qualitative Results of User Study
Prototypes for the User Study (on phone and desktop)
Resharing Protocols
Protocols for Removing and Adding Picosiblings
Integration of the Backup System into Picosibling Management
Special Share Ideas and Use Cases
Android Implementation of the Pico and Picosibling

Table 8: Completed Deliverables of this Project

Future Work Several issues were dealt with in only a preliminary/surface manner. There are alternative secret sharing schemes as described by Simmons [72] or Tassa [85] that allow compartmentalisation and possibly better support for the special shares. These offer the flexibility to define different levels of security without the need for workarounds or weighing the special shares higher by associating them with a higher number of shares. Blakley et al. [10] even introduced a secret sharing scheme that allows disenrollment.

The special shares need to be further explored within user studies and the protocols for them shaped. They can solve many of the discussed security issues and help to prevent the attacks described in section 5.1.

More user studies need to be performed with different age groups. My results show a tendency as to which pairing mechanisms can be discarded for young adults. They are valuable for early deployment stages that involve additional setup as my user group was generally more technology-affine. Yet the study does not prove the usability of the mechanisms for all age groups. Furthermore, studies regarding management alternatives have to be performed. They were excluded for this project because different pairing mechanisms can be executed by different management systems.

Validating the robustness and security of the protocols could formally be accomplished by using *BAN* logic of Burrows et al. [18]. As technology progresses, research can be conducted as to whether other technologies for pairing or communicating become viable. NFC is a particularly attractive option as it allows more natural use cases. However, until technological progress removes certain constraints of embedded systems it must be evaluated whether the proposed protocols are viable in a distributed system of low power devices that underly computational limitations. This thesis was mainly concerned with creating usable and secure protocols without considering the aforementioned constraints in depth.

Lastly, it is yet to be seen whether Picosiblings are the “right” way of locking the Pico. Carrying more devices, mental overhead, and users’ confusion might prevent such an idea from successful deployment.[31] Nevertheless, it can be assumed that users will carry more digitally enhanced items in the future. If Picosiblings could be integrated into such devices, users would not be burdened with additional mental overhead and Picosiblings could become *the* primary way of locking token-based authentication systems.

[31]See appendix A.4 for a discussion of limiting factors for deployment.

A. Appendix

A.1. Content of the Uploaded Material

The uploaded *.zip*-File contains the following material created for this project:

- The research proposal for this thesis.
- Questionnaire used in the user study.
- R scripts for statistical evaluation.
- Graphs created during statistical evaluation, e.g. to confirm normal distribution.
- Source code of the prototypes used in the user study (Android and desktop system).
- Source code of the final Pico prototype.
- Source code of the final Picosibling prototype.
- Source code of the test projects for the Pico and Picosibling prototypes.
- Javadoc documentation of the final Pico and Picosibling prototypes.
- Output log for test runs.
- Output log for *LogCat* showing debugging messages during process execution.

These files can also be found in my public GitHub repository: https://github.com/fabianmakrause

A.2. Notation

Standard notation for cryptographic operations was used in this thesis. Messages are represented in the form

$$A \rightarrow B\text{: } \{MSG\}_K$$

which expresses that principal A sends a message *MSG* to principal B encrypted under key K. Curly brackets represent a cryptographic operation under a key denoted by placing the key in subscript of the message.

Symbol	*Explanation*
P	The Pico of the user
PS	The set of all Picosiblings
PS_i	The ith Picosibling
N_i	A nonce generated by principal i
c_i	Counter i
$K_{P,PS_i,j}$	jth symmetric key between P and PS_i
K_i	Public Key of principal i
K_i^{-1}	Private Key of principal i
$h()$	Hash function
$MAC(MSG)$	Message authentication code of message MSG
S_i	A secret issued by principal i
K_{Master}	The master key to access the secure core of the Pico
s_i	The share of the ith Picosibling
$desc_i$	Description of the ith Picosibling

PS ID	Description	Key Material	Counter	Nonces	Share	Decay	Offset
...	...	...	...	...	...	...	...
267	Watch 23/01/12	$K_{P,PS_{267},1}, K_{P,PS_{267},2}$	34	N_P, N_{267}	x_{267}	13	o_{267}
...	...	...	...	...	...	...	...

Table 9: Storage Structure of the Pico's Outer Core

A.3. Storage Structure

The internal storage of Pico and Picosibling after pairing is as follows.

Within the Pico, descriptions/images of all Picosiblings are stored with a distinctive ID of the Picosibling. Further, the key material $K_{P,PS_i,1},\ K_{P,PS_i,2},\ c_i$ and the x-coordinate of the share (x_i) are stored as well as recently used nonces.

Table 9 shows the exemplified storage structure of the Pico's outer core.[32] The example shows the case that Picosibling 267 has not responded to potential pings of the Pico. Therefore, the share part y_{267} is not stored. Otherwise, the value would be stored within the "Share" part of the storage. Picosiblings are identified by a unique *PS ID*. The Pico further stores the description of the Picosibling, key material consisting of the two symmetric keys, and the counter that determines the hashing of the ephemeral key. Sent and received nonces are stored as well as the x-value $x_{PS_{ID}}$ of the Picosibling's share. The y-value $y_{PS_{ID}}$ is only stored after successfully pinging the Picosibling. A decay timer defines the time the y-value has to be re-fetched from the Picosibling. If this re-pinging of the Picosibling fails, the y-value will be wiped from the memory. The offset is used for the resharing protocol as described in section 5.2.1.

Picosiblings, conversely, store the key material $K_{P,PS_i,1},\ K_{P,PS_i,2},\ c_i$, recently used nonces, and the y-coordinate of the share (y_i). Depending on the pairing method they could also store their public keys, their descriptions/images, and their passkeys or secrets. See table 10 for the exemplified storage structure of a paired Picosibling.

If the countermeasures against sending corrupted shares, as described in section 5.1, are in place, the storage will be expanded by two columns: Salt and hash of the share's y-coordinate concatenated with the salt.

[32]The outer core of the Pico contains the cryptographic values to communicate with paired Picosiblings. This part is not contained in the secure core and is stored in plaintext within the Pico.

Description	Key Material	Counter	Nonces	Share	State
...	...	...	...	...	...
Casio 9371237	$K_{P,PS_{267},1}, K_{P,PS_{267},2}$	34	N_P, N_{267}	y_{267}	PAIRED
...	...	...	...	...	...

Table 10: Storage Structure of the Picosibling

A.4. Discussion

The following sections describe issues raised in this thesis in more depth. They are not essential for following the argumentation but provide more elaborate explanations.

A.4.1. Deployment

Bonneau et al. [14] found that passwords are superior in terms of deployability and no other authentication scheme is strictly better. Developing an alternative authentication scheme requires benefits over passwords but also to create incentives for users and service providers to adapt. If there is no incentive for them to switch to the new technology, they will not change their habits nor will they be willing to spend money on marginal benefits. Marginal improvements over passwords are not worth the cost of disruption as stated by Herley and van Oorschot [35].

Several participants noted their reluctance to wear further devices as Picosiblings. Prior to deploying Picosiblings as the locking mechanism for the Pico, users must be familiarised with them. In early deployment stages, the number of Picosiblings could be decreased and Picosiblings integrated in smart-devices with which the user is already familiar. Smart-watches, tablets, Google glass, and smartphones, once the Pico is developed as a stand-alone token, could serve as Picosiblings and help users become acquainted with using and managing Picosiblings.

A.4.2. Pico

The Pico could offer varying degrees of the master key to support different levels of authentication. Less valuable accounts could be stored outside the secure core, such that the user only needs the Pico (without the Picosiblings) to access them. One of the main problems of passwords is that they are used for different degrees of security, as discussed by Herley and van Oorschot [35]. The Pico could overcome this issue by providing authentication by classification thus clarifying different degrees of security for the user.

The *no-typing* benefit of the Pico is violated by several protocols discussed in this thesis. This problem could only be overcome for managing the Picosiblings if Picosiblings were identified through photographs instead of descriptions. However, the backup system still requires writing down a recovery code and entering it when the backup is to be accessed, which is why the *no-typing* benefit of the Pico is weakened to a *quasi-no-typing* benefit. Typing is only required for setup but not for authentication.

A.5. Implementation

A.5.1. Architecture of the Prototypes used in the User Study

Management Application For the Java application that simulates the management of the Picosiblings, a simple design decoupling domain objects, utility classes, and the GUI was used.

At start, a *JFrame* object is created and filled with an abstract *ScenarioPanel*. The ScenarioPanel encapsulates common behaviour for all used scenarios, e.g. layout, Picosibling-list, and the remove button. The specific *JPanel* for the current scenario extends the ScenarioPanel class and injects scenario-specific behaviour like actions when clicking the "Confirm Pairing" button or the description for the scenario.

Scenarios are modeled as an *enum* class. They represent the domain model together with the *ExperimentData* class that encapsulates data collected during the experiment. A *DataWriter* class to manage I/O of ExperimentData objects served as a utility class and completes the system.

Pico Android Application Similar to the management application, the Android application was separated into GUI elements, domain objects, and utility classes.

Activities for each *Scenario* extend a superclass *ScenarioActivity* that encapsulates the layout of the app, management of the domain objects *ExperimentData* and *Scenario* that are identical to the management application's classes, and management of the *DataWriter* utility class that writes collected *ExperimentData* to an external file on the phone. The specific *Activities* for each scenario further specify the behaviour of the "Pair" button as well as the instructions and dialogs.

At the end of a run, indicated by the number of paired Picosiblings, the application created a log file containing information about the number of clicks and the time needed for task completion.

Screenshots of the general layout of both applications can be found in appendix A.6.2.

A.5.2. Architecture of the Final Prototype

Complete documentation of the classes can be found in the javadoc attached to the source codes.[33]

Domain Modelling The prototypes consistently use domain objects to perform their tasks on.

Messages between Pico and Picosibling are encapsulated in *PicoMessage* objects that are of a *PicoMessageType*. Each part of a *PicoMessage* is encapsulated as a *PicoMessageSection* to simplify parsing through subclasses of the *PicoMessageParser*. Each protocol is associated with a parser: *PicoPairMessageParser* or *PicoPingMessageParser*. Messages for the ping-protocol are built using the *PicoPairMessageBuilder*, whereas the *PicoPingMessageBuilder* creates messages for the ping-protocol.

When the state of a *BluetoothConnection* changes, e.g. the connection to a Picosibling is cancelled or established, the corresponding *BluetoothState* is changed as well to propagate the new state to the subclasses of the *BluetoothConnectionManager*.

For the Pico prototype, the properties of the Pico are encapsulated in a *PicoProperties* object and the state modelled as a *PicoState* enum. Within the Picosibling application, a *PicosiblingInternalState* object models the state of the Picosibling. The Pico application, on the other hand, models the states of its associated Picosiblings through *Picosibling* objects.

Database Management The database schema is defined by the *PicoOuterCoreContract* class and the tables created through the *PicoOuterCoreOpenHelper* class. Access to the database is proxied through two database management classes: *PropertiesDBManager* and *Picosiblings-DBManager*. The former provides methods to access and modify the PROPERTIES table containing the Pico's state, its own key material, the threshold number, the modulus for calculations, flags indicating that certain protocols have been completed, and other information that simplify the process flows. Picosiblings' credentials are stored in the PICOSIBLINGS table that basically models the table shown in appendix A.3.

Architecture To allow for a flexible implementation and extensibility, the applications were developed using best practices of software engineering. The asynchronous tasks for running the protocols are only loosely coupled to the activity controlling the GUI by implementing a form of the *Observer* pattern. Similarly, the *BluetoothConnection* objects are observed by the *BluetoothConnectionManager* subclasses to prevent tight coupling and potential exceptions.

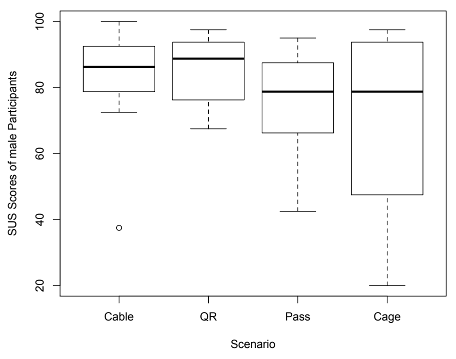
As previously described, domain states were modelled consistently using *enums* and domain classes. Whenever possible, builder classes and proxies encapsulated access to critical data,

[33]In the zip-file or at https://github.com/fabianmakrause.

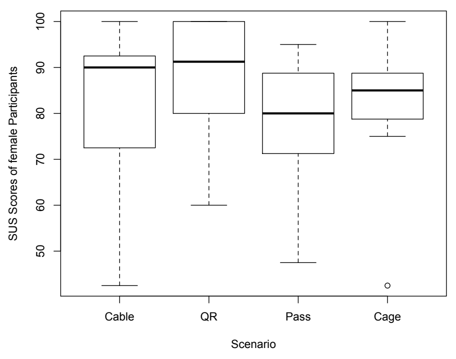
for example using database management helper classes for accessing the database and builder classes to create the protocol's messages.

A.6. Experiment Material

A.6.1. Graphs for Statistical Analysis

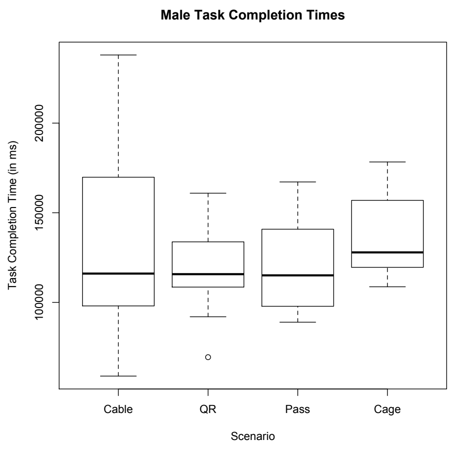


(a) Boxplot of Male SUS Scores by Scenario

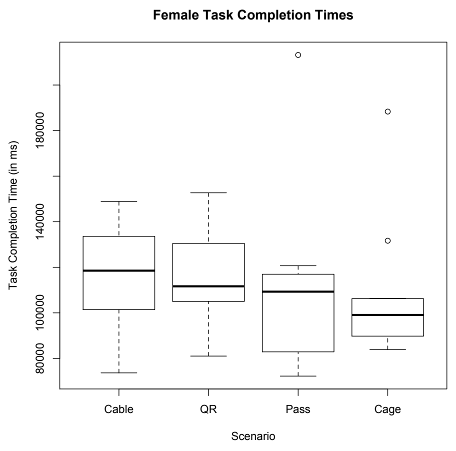


(b) Boxplot of Female SUS Scores by Scenario

Fig. 13: Boxplots of SUS Data by Gender

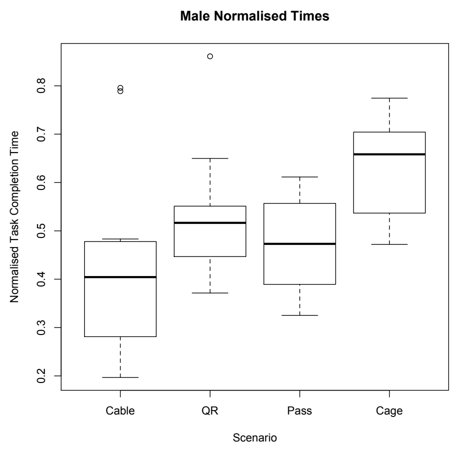


(a) Boxplot of Male Time by Scenario

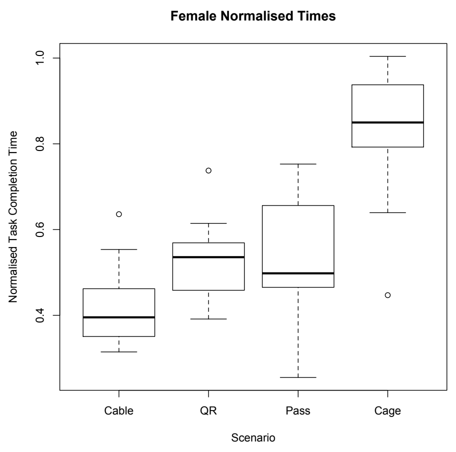


(b) Boxplot of Female Time by Scenario

Fig. 14: Boxplots of Time Data by Gender

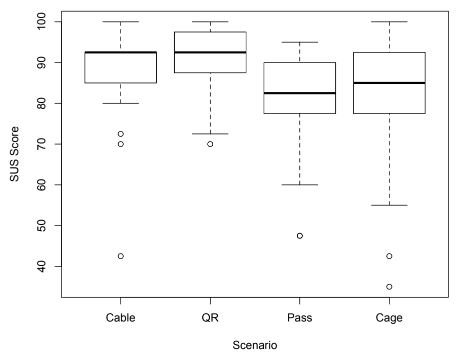


(a) Boxplot of Male Normalised Time by Scenario

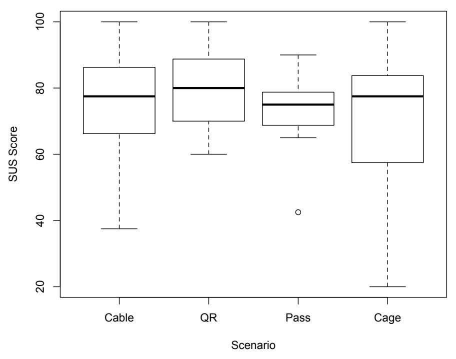


(b) Boxplot of Female Normalised Time by Scenario

Fig. 15: Boxplots of Normalised Time Data by Gender

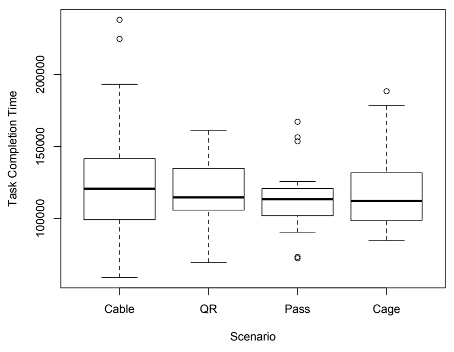


(a) Boxplot of SUS scores of Experienced Participants by Scenario

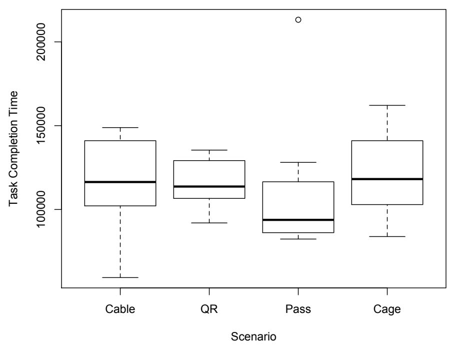


(b) Boxplot of SUS scores of Unexperienced Participants by Scenario

Fig. 16: Boxplots of SUS Scores by Experience

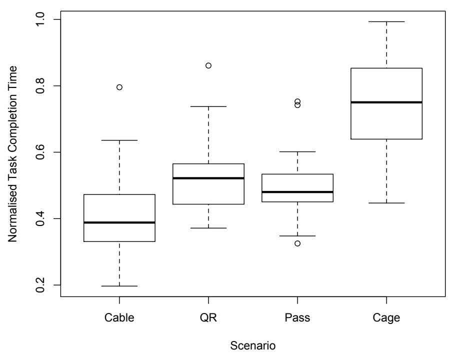


(a) Boxplot of Time of Experienced Participants by Scenario

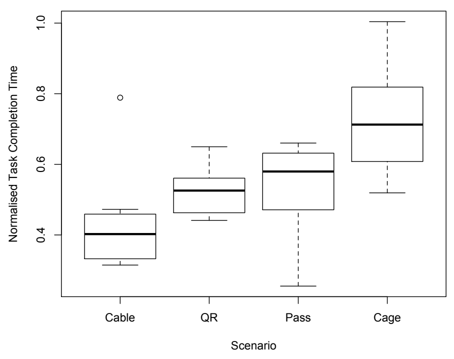


(b) Boxplot of Time of Unexperienced Participants by Scenario

Fig. 17: Boxplots of Task Completion Times by Experience



(a) Boxplot of Normalised Time of Experienced Participants by Scenario



(b) Boxplot of Normalised Time of Unexperienced Participants by Scenario

Fig. 18: Boxplots of Normalised Task Completion Times by Experience

A.6.2. Pictures of the Experimental Setup

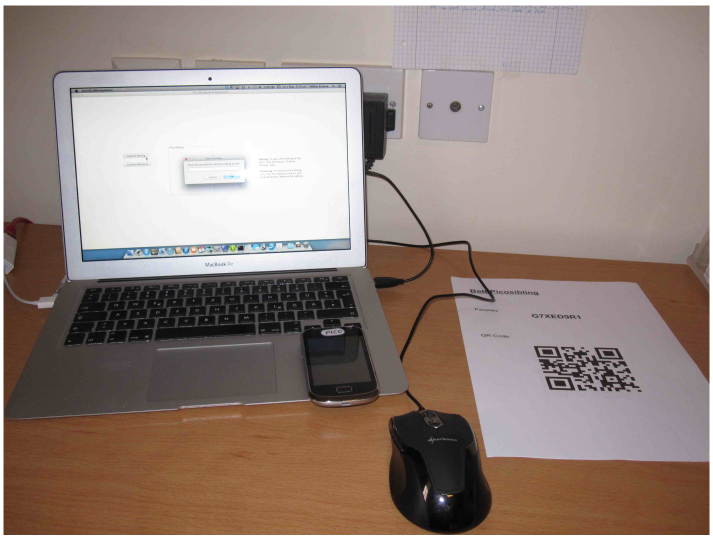


Fig. 19: Setup of Pico, Laptop, and Picosibling Sheet

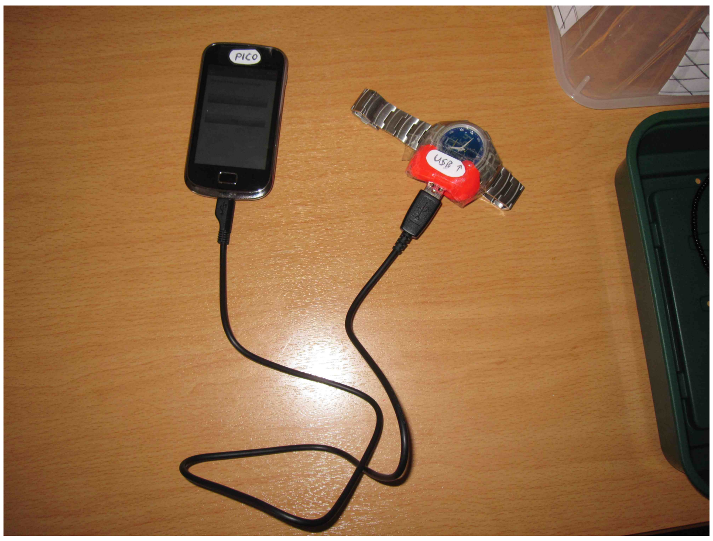
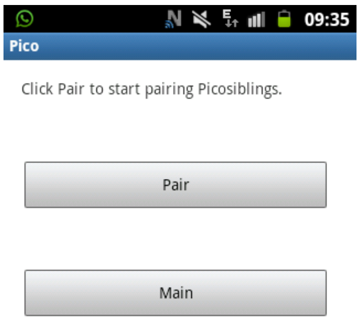


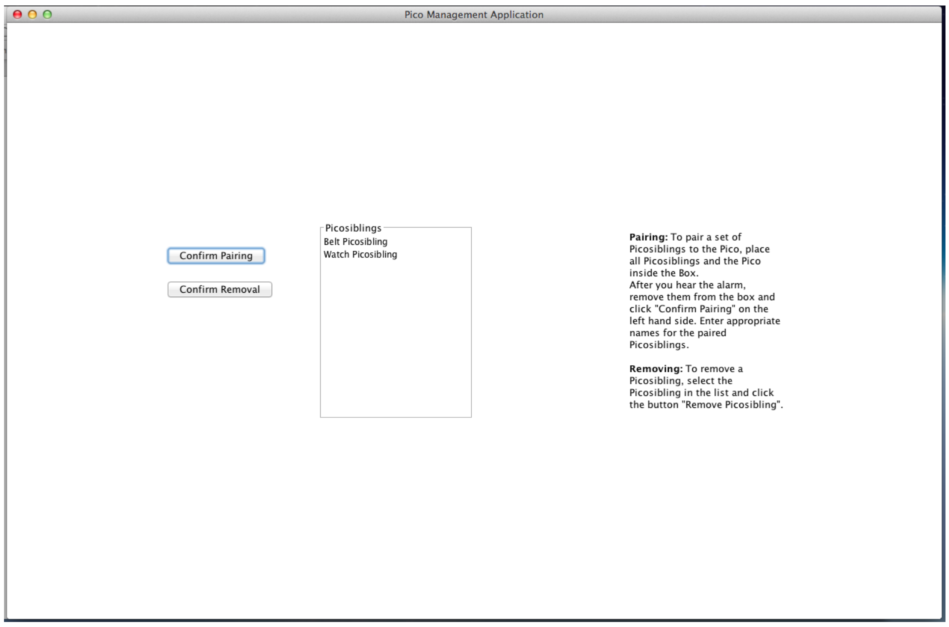
Fig. 20: Physical Connection between Pico and Picosibling

A.6.3. Screenshots of the User Study Prototype



Click Pair to start pairing Picosiblings.

Fig. 21: User Interface of the Pico



Belt Picosibling
Watch Picosibling

Pairing: To pair a set of Picosiblings to the Pico, place all Picosiblings and the Pico inside the Box.
After you hear the alarm, remove them from the box and click "Confirm Pairing" on the left hand side. Enter appropriate names for the paired Picosiblings.

Removing: To remove a Picosibling, select the Picosibling in the list and click the button "Remove Picosibling".

Fig. 22: User Interface of the Application

A.6.4. Flow Charts of the Scenarios

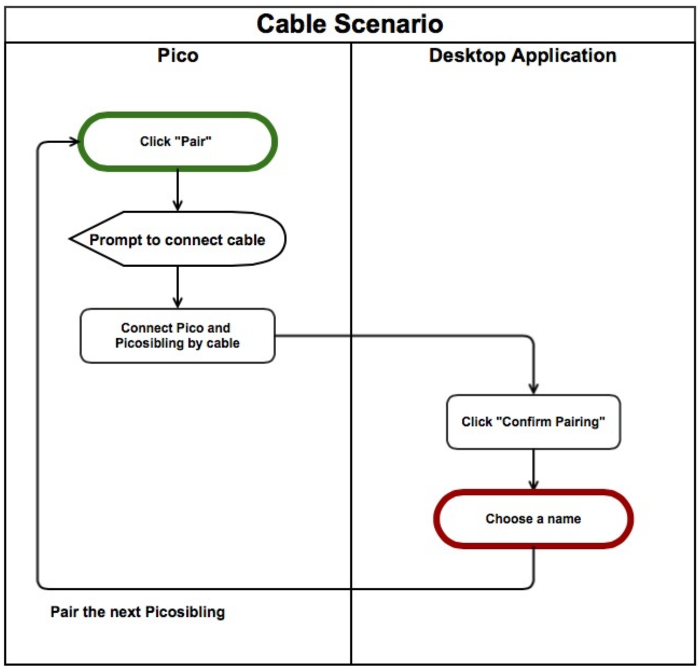


Fig. 23: Flow Chart of the User Study's Cable Scenario

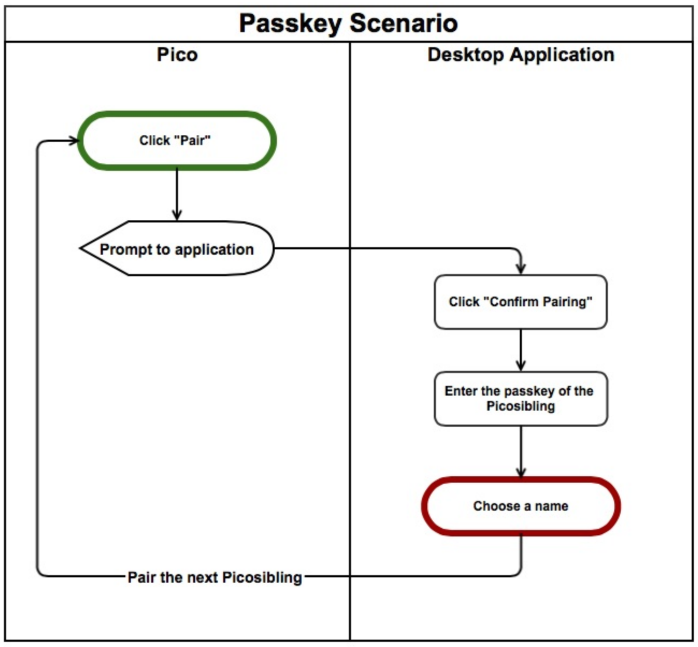


Fig. 24: Flow Chart of the User Study's Passkey Scenario

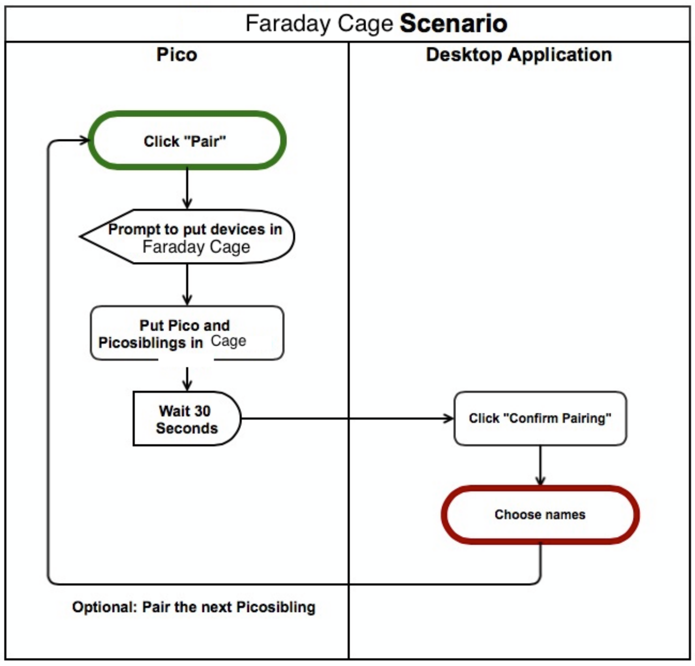


Fig. 25: Flow Chart of the User Study's Faraday Scenario

A.6.5. Pilot Study Flow Charts

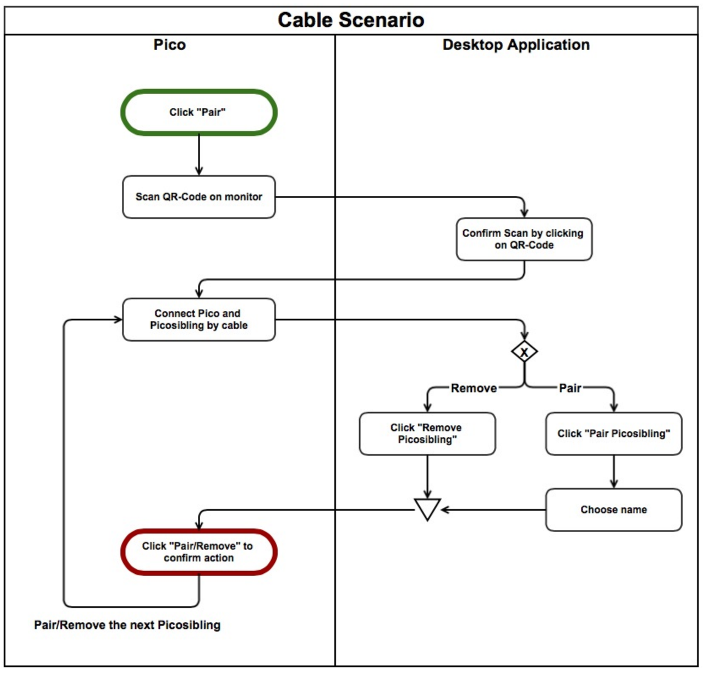

Cable Scenario
Pico
Desktop Application
Click "Pair"
Scan QR-Code on monitor
Confirm Scan by clicking on QR-Code
Connect Pico and Picosibling by cable
X
Remove
Pair
Click "Remove Picosibling"
Click "Pair Picosibling"
Choose name
Click "Pair/Remove" to confirm action
Pair/Remove the next Picosibling

Fig. 26: Flow Chart of the Pilot's Cable Scenario

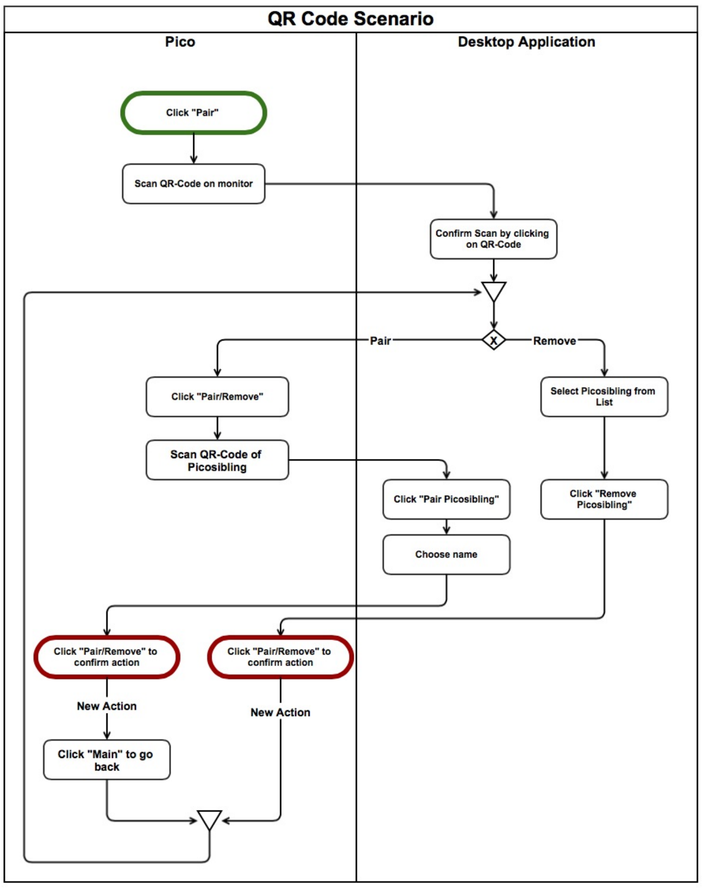

QR Code Scenario
Pico
Desktop Application
Click "Pair"
Scan QR-Code on monitor
Confirm Scan by clicking on QR-Code
Pair
X
Remove
Click "Pair/Remove"
Scan QR-Code of Picosibling
Select Picosibling from List
Click "Pair Picosibling"
Choose name
Click "Remove Picosibling"
Click "Pair/Remove" to confirm action
Click "Pair/Remove" to confirm action
New Action
New Action
Click "Main" to go back

Fig. 27: Flow Chart of the Pilot's QR Code Scenario

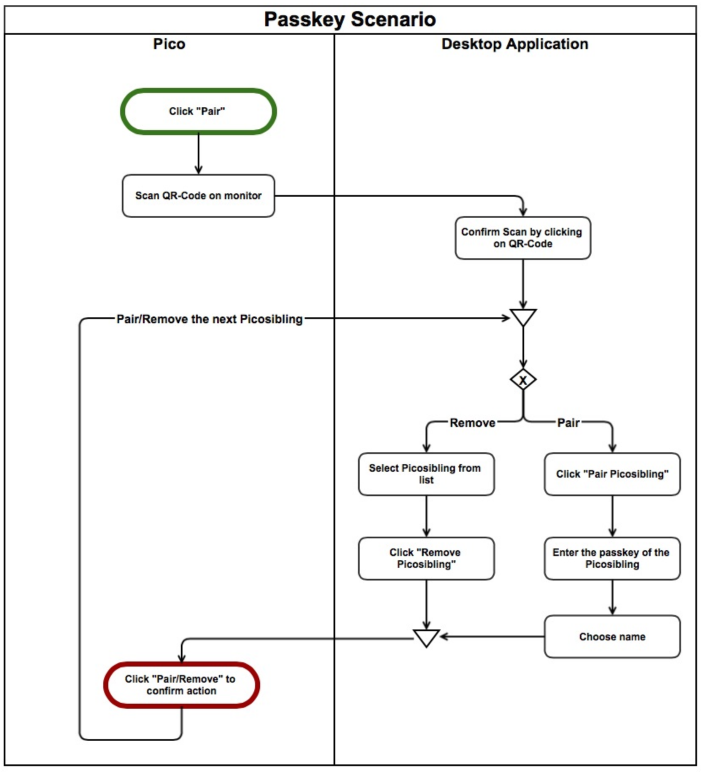

Passkey Scenario
Pico
Desktop Application
Click "Pair"
Scan QR-Code on monitor
Confirm Scan by clicking on QR-Code
Pair/Remove the next Picosibling
Remove
Pair
Select Picosibling from list
Click "Pair Picosibling"
Click "Remove Picosibling"
Enter the passkey of the Picosibling
Choose name
Click "Pair/Remove" to confirm action

Fig. 28: Flow Chart of the Pilot's Passkey Scenario

References

[1] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42 (12), 1999. (Cited on page 1.)

[2] G. T. Amariucai, C. Bergman, and Y. Guan. An automatic, time-based secure pairing protocol for passive RFID. In *RFID. Security and Privacy*, pages 108–126. Springer Berlin Heidelberg, 2012. (Cited on page 11.)

[3] Y. Ayatsuka and J. Rekimoto. tranSticks: physically manupulatable virtual connections. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 251–260, 2005. (Cited on page 12.)

[4] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Network and Distributed System Security Symposium*. NDSS, 2002. (Cited on page 9.)

[5] BBC. Linkedin passwords leaked by hackers. june 2012. Last checked: 26.03.2014. URL http://www.bbc.co.uk/news/technology-18338956. (Cited on page 1.)

[6] A. Beimel. Secret-sharing schemes: a survey. In *Coding and cryptology*, pages 11–46. Springer Berlin Heidelberg, 2011. (Cited on page 5.)

[7] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Symposium on Research in Security and Privacy*, pages 72–84. IEEE, 1992. (Cited on page 14.)

[8] A. Bentzon. Security architecture and implementation for a TPM-based mobile authentication device. Master's thesis, University of Cambridge Computer Laboratory, June 2013. (Cited on pages 1, 3, and 4.)

[9] K. J. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, The Mitre Corporation, April 1977. (Cited on page 7.)

[10] B. Blakley, G. R. Blakley, A. H. Chan, and J. L. Massey. Threshold schemes with disenrollment. In Springer Berlin Heidelberg, editor, *Advances in Cryptology*, CRYPTO'92, pages 540–548, 1993. (Cited on page 46.)

[11] G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, pages 313–317, New York, 1979. AFIPS Press. (Cited on page 5.)

[12] Bluetooth Special Interest Group. Simple pairing whitepaper. 2006. Last checked: 15.04.2014. URL http://mclean-linsky.net/joel/cv/Simple20Pairing_WP_V10r00.pdf. (Cited on pages 11 and 30.)

[13] J. Bonneau and S. Preibusch. The password thicket: technical and market failures in human authentication on the web. In *Proceedings of WEIS*, 2010. (Cited on page 1.)

[14] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. *IEEE Security and Privacy*, 2012. (Cited on pages 1 and 52.)

[15] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Springer Berlin Heidelberg, editor, *Advances in Cryptology - Eurocrypt 2000*, pages 156–171, 2000. (Cited on page 14.)

[16] J. Brooke. SUS - a quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, and A. L. McClelland, editors, *Usability Evaluation in Industry*. Taylor and Francis, London, 1996. (Cited on page 18.)

[17] I. Buhan, J. Doumen, P. Hartel, and R. Veldhuis. Secure ad-hoc pairing with biometric: SAfE. In *Proceedings of the First International Workshop on Security for Spontaneous Interaction (IWSSI 2007)*, pages 450–456, 2007. (Cited on page 9.)

[18] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, volume 426, pages 233–271, 1989. (Cited on page 46.)

[19] M. Cagalj, S. Capkun, and J.-P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, 2006. (Cited on page 11.)

[20] C. Castelluccia and G. Avoine. Noisy tags: A pretty good key exchange protocol for RFID tags. In *Smart Card Research and Advanced Applications*, pages 289–299. Springer Berlin Heidelberg, 2006. (Cited on page 11.)

[21] C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 51–64. ACM, 2005. (Cited on page 10.)

[22] J. Cheng, X. Hu, and P. B. Heidorn. New measures for the evaluation of interactive information retrieval systems: Normalized task completion time and normalized user effectiveness. In *Proceedings of the American Society for Information Science and Technology*, volume 47, pages 1–9, 2010. (Cited on page 19.)

[23] M. K. Chong, R. Mayrhofer, and H. Gellersen. A survey of user interaction for spontaneous device association. *ACM Computing Surveys*, 2014 (to be published). (Cited on page 11.)

[24] E. Dawson and D. Donovan. The breadth of shamir's secret-sharing scheme. *Computers & Security*, 13(1):69–78, 1994. (Cited on page 5.)

[25] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590, 2006. (Cited on page 11.)

[26] W. Diffie and M. E. Hellman. New directions in cryptography. In *Information Theory, IEEE Transactions on*, volume 22, pages 644–654, 1976. (Cited on page 8.)

[27] D. Dolev and A. C. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983. (Cited on page 6.)

[28] C. Ellison and S. Dohrmann. Public-key support for group collaboration. *ACM Transactions on Information and System Security (TISSEC)*, 6(4), 2003. (Cited on page 8.)

[29] D. Florencio and C. Herley. A large-scale study of web password habits. In *WWW 2007 Proceedings of the 16th international conference on World Wide Web*, 2007. (Cited on page 1.)

[30] C. Gehrmann and C. J. Mitchell. Manual authentication for wireless devices. *RSA Cryptobytes*, 7(1):29–37, 2004. (Cited on page 11.)

[31] C. Gehrmann and K. Nyberg. Security in personal area networks. In C. J. Mitchell, editor, *Security for Mobility*, pages 191–229. IEEE TELECOMMUNICATIONS SERIES, 2004. (Cited on page 11.)

[32] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*. IEEE, 2006. (Cited on page 9.)

[33] M. T. Goodrich, M. Sirivianos, J. Solis, C. Soriente, G. Tsudik, and E. Uzun. Using audio in secure device pairing. *International Journal of Security and Networks*, 4(1): 57–68, 2009. (Cited on page 9.)

[34] J. C. Hanna. Configuring security parameters in small devices. july 2002. Last checked: 11.05.2014. URL http://tools.ietf.org/html/draft-hanna-zeroconf-seccfg-00. (Cited on page 9.)

[35] C. Herley and P. C. van Oorschot. A research agenda acknowledging the persistence of passwords. *Security & Privacy, IEEE*, 10(1):28–36, 2012. (Cited on page 52.)

[36] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Springer Berlin Heidelberg, editor, *Advances in Cryptology*, CRYPTO'95, pages 339–352, 1995. (Cited on page 33.)

[37] K. Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 149–158. ACM, 2003. (Cited on page 10.)

[38] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Ubicomp 2001: Ubiquitous Computing*, pages 116–122. Springer Berlin Heidelberg, 2001. (Cited on page 10.)

[39] M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989. (Cited on page 5.)

[40] Y. Iwasaki, N. Kawaguchi, and Y. Inagaki. Touch-and-connect: A connection request framework for ad-hoc networks and the pervasive computing environment. In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 20–29. IEEE, 2003. (Cited on page 10.)

[41] M. Jakobsson. Method and apparatus for immunizing against offline dictionary attacks. Technical report, U.S. Patent Application 60/283,996, Filed on 16th April, 2001. (Cited on page 11.)

[42] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Springer Berlin Heidelberg, editor, *Advances in Cryptology - Eurocrypt 2001*, pages 475–494, 2001. (Cited on page 14.)

[43] T. Kindberg and K. Zhang. Secure spontaneous device association. In A. K. Dey, A. Schmidt, and J. F. McCarthy, editors, *Lecture Notes in Computer Science*, volume 2864 of *UbiComp*, pages 124–131. Springer Berlin Heidelberg, 2003. (Cited on page 9.)

[44] T. Kindberg and K. Zhang. Validating and securing spontaneous associations between wireless devices. In C. Boyd and W. Mao, editors, *Lecture Notes in Computer Science*, volume 2851 of *ISC*, pages 44–53. Springer Berlin Heidelberg, 2003. (Cited on page 9.)

[45] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun. Caveat eptor: A comparative study of secure device pairing methods. In *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, PerCom, pages 1–10. IEEE, 2009. (Cited on pages 8 and 9.)

[46] C. Kuo, M. Luk, R. Negi, and A. Perrig. Message-in-a-bottle: user-friendly and secure key deployment for sensor nodes. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 233–246. ACM, 2007. (Cited on pages 9 and 14.)

[47] S. Laur and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. In *Cryptography and Network Security*, pages 90–107. Springer Berlin Heidelberg, 2006. (Cited on page 11.)

[48] H. J. Lee, H. J. Cho, W. Xu, and A. Fairhurst. The influence of consumer traits and demographics on intention to use retail self-service checkouts. *Marketing Intelligence & Planning*, 28(1):46–58, 2010. (Cited on page 17.)

[49] F. X. Lin, D. Ashbrook, and S. White. RhythmLink: securely pairing I/O-constrained devices by tapping. In *Proceedings of the 24th annual ACM symposium on User Interface software and technology*, pages 263–272. ACM, 2011. (Cited on page 10.)

[50] K. M. Martin. Challenging the adversary model in secret sharing schemes. In *Coding and Cryptography II. Proceedings of the Royal Flemish Academy of Belgium for Science and the Arts*, pages 45–63, 2008. (Cited on page 31.)

[51] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Mandayam. Proximate: proximity-based secure pairing using ambient wireless signals. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 211–224. ACM, 2011. (Cited on page 11.)

[52] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *Pervasive Computing*, pages 144–161. Springer Berlin Heidelberg, 2007. (Cited on page 10.)

[53] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Security and Privacy, IEEE Symposium on*, pages 110–124. IEEE, 2005. (Cited on page 8.)

[54] P. Morilla, C. Padro, G. Saez, and J. L. Villar. Weighted threshold secret sharing schemes. *Information Processing Letters*, 70(5):211–216, 1999. (Cited on page 5.)

[55] R. Morris and K. Thompson. Password security: a case history. *Communications of the ACM*, 22(11), 1979. (Cited on page 1.)

[56] A. Nicholson, I. Smith, J. Hughes, and B. Noble. LoKey: Leveraging the SMS network in decentralized, end-to-end trust establishment. In K. Fishkin, B. Schiele, P. Nixon, and A. Quigley, editors, *Lecture Notes in Computer Science (Pervasive Computing)*, volume 3968, pages 202–219. Springer Berlin Heidelberg, 2006. (Cited on page 9.)

[57] D. G. Park, J. K. Kim, J. B. Sung, J. H. Hwang, C. H. Hyung, and S. W. Kang. TAP: touch-and-play. In *Proceeddings of the SIGCHI conference on Human Factors in computing systems*, pages 677–680. ACM, 2006. (Cited on page 12.)

[58] S. Pasini and S. Vaudenay. SAS-based authenticated key agreement. In *Public Key Cryptography - PKC 2006*, pages 395–409. Springer Berlin Heidelberg, 2006. (Cited on page 11.)

[59] R. Peeters. *Security Architecture for Things That Think.* PhD thesis, Katholieke Universiteit Leuven, June 2012. (Cited on page 34.)

[60] C. Peng, G. Shen, Y. Zhang, and S. Lu. Point&Connect: intention-based device pairing for mobile phone users. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 137–150. ACM, 2009. (Cited on page 10.)

[61] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *International Workshop on Cryptographic Techniques and E-Commerce*, pages 131–138, 1999. (Cited on page 8.)

[62] R. Prasad and N. Saxena. Efficient device pairing using "human-comparable“ synchronized audiovisual patterns. In *Applied Cryptography and Network Security*, pages 328–345. Springer Berlin Heidelberg, 2008. (Cited on page 10.)

[63] J. Rekimoto, Y. Ayatsuka, and M. Kohno. SyncTap: An interaction technique for mobile networking. In *Human-Computer Interaction with Mobile Devices and Services*, pages 104–115. Springer Berlin Heidelberg, 2003. (Cited on page 10.)

[64] J. Rekimoto, Y. Ayatsuka, M. Kohno, and H. Oba. Proximal interactions: A direct manipulation technique for wireless networking. *Interact*, 3:511–518, 2003. (Cited on page 9.)

[65] J. Rekimoto, T. Miyaki, and M. Kohno. ProxNet: secure dynamic wireless connection by proximity sensing. In *Pervasive Computing*, pages 213–218. Springer Berlin Heidelberg, 2004. (Cited on page 10.)

[66] V. Roth, W. Polak, T. Turner, and E. Rieffel. Simple and effective defense against evil twin access points. In *ACM Conference on Wireless Network Security, WISEC*, pages 220–235, 2008. (Cited on page 8.)

[67] N. Saxena and M. B. Uddin. Automated device pairing for asymmetric pairing scenarios. In *Information and Communications Security*, pages 311–327. Springer Berlin Heidelberg, 2008. (Cited on page 9.)

[68] N. Saxena and M. B. Uddin. Secure pairing of “Interface-constrained” devices resistant against rushing user behavior. In *Applied Cryptography and Network Security*, pages 34–52. Springer Berlin Heidelberg, 2009. (Cited on page 11.)

[69] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 307–313. IEEE, 2006. (Cited on page 9.)

[70] N. Saxena, M. B. Uddin, and J. Voris. Treat'em like other devices: user authentication of multiple personal rfid tags. *SOUPS*, 9, 2009. (Cited on page 9.)

[71] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979. (Cited on page 5.)

[72] G. J. Simmons. How to (really) share a secret. In Springer Verlag New York, editor, *Proceedings on Advances in cryptology*, pages 390–448, 1990. (Cited on page 46.)

[73] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-enabled device association. In *Proceedings of the International Workshop on Security for Spontaneous Interaction IWSSI, UbiComp Workshop*, 2007. (Cited on page 10.)

[74] C. Soriente, G. Tsudik, and E. Uzun. HAPADEP: human-assisted pure audio device pairing. In *Information Security*, pages 385–400. Springer Berlin Heidelberg, 2008. (Cited on page 9.)

[75] C. Soriente, G. Tsudik, and E. Uzun. Secure pairing of interface constrained devices. *International Journal of Security and Networks*, 4(1):17–26, 2009. (Cited on page 12.)

[76] F. Stajano. The resurrecting duckling - what next? In Springer Berlin Heidelberg, editor, *Security Protocols*, 2001. (Cited on pages 3 and 7.)

[77] F. Stajano. Pico: No more passwords. In *Security Protocols XIX*, pages 49–81, 2011. (Cited on pages 1, 3, and 8.)

[78] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, J. Malcolm, and M. Roe, editors, *Security Protocols, Lecture Notes in Computer Science*, volume 1796, pages 172–182. Springer Berlin Heidelberg, 2000. (Cited on pages 4, 7, 12, and 32.)

[79] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ubiquitous computing. *Computer*, 35(4):22–26, 2002. (Cited on page 7.)

[80] F. Stajano, G. Jenkinson, J. Payne, M. Spencer, Q. Stafford-Fraser, and C. Warrington. Bootstrapping adoption of the pico password replacement system. In B. Christianson et

al., editor, *To appear: Proceedings of Security Protocols Workshop*. Springer LNCS, 2014. (Cited on pages 17 and 31.)

[81] O. Stannard. Picosiblings. Bachelor's thesis, University of Cambridge Computer Laboratory, May 2012. (Cited on pages 1, 4, 7, 13, 33, 38, and 41.)

[82] O. Stannard and F. Stajano. Am I in good company? A privacy-protecting protocol for cooperating ubiquitous computing devices. In *Security Protocols Workshop XX*, pages 223–230, 2012. (Cited on pages 4, 6, and 33.)

[83] D. R. Stinson. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, 2:357–390, 1992. (Cited on page 5.)

[84] C. Swindells, K. M. Inkpen, J. C. Dill, and M. Tory. That one there! pointing to establish device identity. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 151–160. ACM, 2002. (Cited on page 9.)

[85] Tamir Tassa. Hierarchical threshold secret sharing. *Journal of Cryptology*, 20(2):237–264, 2007. (Cited on page 46.)

[86] M. Tompa and H. Woll. How to share a secret with cheaters. *Journal of Cryptology*, 1 (3):133–138, 1989. (Cited on page 31.)

[87] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *Financial Cryptography and Data Security*, pages 307–324. Springer Berlin Heidelberg, 2007. (Cited on pages 11 and 12.)

[88] A. Varshavsky, A. Scannell, A. LaMarca, and E. De Lara. Amigo: Proximity-based authentication of mobile devices. In *Ubiquitous Computing*, pages 253–270. Springer Berlin Heidelberg, 2007. (Cited on page 10.)

[89] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology - CRYPTO 2005*, pages 309–326. Springer Berlin Heidelberg, 2005. (Cited on page 11.)

[90] T. G. Zimmermann. Personal area networks: near-field intrabody communication. *IBM Systems Journal*, 35(3.4):609–617, 1996. (Cited on page 12.)